ΠΑΝΟΣ ΦΙΤΣΙΛΗΣ
ΚΑΘΗΓΗΤΗΣ

Ευέλικτες μέθοδοι διοίκησης και διαχείρισης έργων

Kanban

Scrum

DevOps

ΚΑΛΛΙΠΟΣ
ανοικτές ακαδημαϊκές εκδόσεις

Ευέλικτες μέθοδοι διοίκησης και διαχείρισης έργων

Συγγραφή

Πάνος Φιτσιλής

ΚΑΛΛΙΠΟΣ

Εθνικό Μετσόβιο Πολυτεχνείο

Ηρώων Πολυτεχνείου 9, 15780 Ζωγράφου

www.kallipos.gr

ISBN:

God grant me the serenity to accept the things I cannot change;

The courage to change the things I can;

And the wisdom to know the difference.

Francis of Assisi

Αφιερωμένο στην αγαπημένη μου κόρη, τη Μαρίνα,

που ως φοιτήτρια προσπαθεί να κατανοήσει πολύπλοκες οργανωτικές δομές

και τη λειτουργία των ομάδων.

Θα ήθελα να μοιραστώ μαζί σου ένα μυστικό….

ο αγώνας αυτός δεν τελειώνει ποτέ.

Περιεχόμενα

Πίνακας Εικόνων

Πίνακας πινάκων

Πίνακας συντομεύσεων-ακρωνυμίων

Ακρώνυμο	Περιγραφή
AHP	Analytical Hierarchical Process
AM	Agile Modelling
API	Application Programming Interface
AWS	Amazon Web Services
CVCS	Centralized Version Control System
CMDB	Configuration Management DB
CD	Continuous Delivery
CD3	Cost of Delay Divided by Duration
CI	Continuous Integration
CIs	Configurable Items
CoD	Cost of Delay
CT	Cycle Time
DAST	Dynamic Application Security Testing
DVCS	Distributed Version Control System
DEEP	Detailed, Emergent, Estimated, Prioritised
DevOps	Development and Operations
DMAIC	Define, Measure, Analyze, Improve, και Control
DoD	Definition of Doing
DoF	Definition of Fun
DRY	Don't Repeat Yourself
EFQM	European Foundation for Quality Management
FDD	Feature-Driven Development
FIFO	First In First Out
HRM	Human Resource Management
IaC	Infrastructure-as-Code
ICT	Information and Communication Technology
INVEST	Independ, Negotiable, Value, Estimable, Small, Testable
JIT	Just-in-Time
LD	Lean Development
LIFO	Last in First Out
LRM	Last Responsible Moment

Ακρώνυμο	Περιγραφή
MoSCoW	AMMust Should Could Would
MTTR	Mean Time to Repair
MUS	Minimum Usable Subset
MVP	Minimum Viable Product
NIST	National Institute of Standards and Technology
OAOO	Once And Only Once
OCI	Open Container Initiative
OKR	Objectives and Key Results
P&L	Profit and Losses
PBI	Product Backlog Items
PDCA	Plan-Do-Check-Act
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
PMLC	Project Management Life Cycle
POM	Project Object Model
QA	Quality Assurance
REST API	Representational State Transfer API
ROI	Return Of Investment
SaaS	Software as a Service
SAST	Static Application Security Testing
SBD	Set Based Development
SDLC	Software Development Life Cycle
SLA	Service level Agreements
SoS	Scrum of Scrums
SVC	Software Version Control
SWBOK	Software Engineering Body of Knowledge
TCO	Total Cost of Ownership
TDD	Test Driven Development
TPS	Toyota Production System
TTM	Time to Market
UAT	User Acceptance Test
VM	Virtual Machines
VMM	Virtual Machine Monitor
VSM	Value Stream Mapping

Ακρώνυμο	Περιγραφή
VUCA	Volatile, Uncertain, Complex, και Ambiguous
WIP	Work In Progress
WPM	Wiegers Prioritization Matrix
XML	eXtended Markup Language
XP	eXtreme Programming
YAGNI	You Aren't Going to Need It
ΔΟΠ	Διοίκησης Ολικής Ποιότητας
ΥΝ	Υπολογιστικό Νέφος

Πρόλογος

Η συγγραφή αυτού του βιβλίου ξεκίνησε πριν από δύο σχεδόν χρόνια υπό την παρακίνηση της ομάδας Κάλλιπος, με επιστημονικό υπεύθυνο τον καθηγητή Νικόλαο Μήτρου. Η ομάδα Κάλλιπος επιτελεί σημαντικό έργο παρουσιάζοντας στο ευρύ κοινό, τους επαγγελματίες, τους επιστήμονες αλλά και στους φοιτητές επιστημονικά βιβλία υψηλής ποιότητας . Πρόκειται για το δεύτερο βιβλίο που εκδίδω με το πρόγραμμα Κάλλιπος και αυτό καταδεικνύει την υποστήριξή μου στα ανοικτά βιβλία. Αρχικά λοιπόν θα ήθελα να τους συγχαρώ και να ενθαρρύνω τη συνέχιση αυτής της προσπάθειας.

Στη συνέχεια θα επιχειρήσω να κάνω μια σειρά παρατηρήσεων, φαινομενικά ασύνδετων, οι οποίες τελικά, κατά την άποψή μου, αιτιολογούν την ανάγκη συγγραφής αυτού του βιβλίου.

- Ίσως να γνωρίζετε μια σειρά βιβλίων, γνωστού διεθνούς εκδοτικού οίκου που εκδίδει βιβλία με τίτλους «Agile Project Management for dummies» ή «DevOps for Dummies» που συνήθως παρουσιάζουν μια απλή, σύντομη και κατανοητή εισαγωγή στο θέμα που πραγματεύονται. Όταν ξεκίνησα να μελετώ τις ευέλικτες μεθόδους, μου φαινόταν ότι όλες αυτές οι έννοιες που θεμελίωναν τις ευέλικτες μεθόδους, ήταν τόσο απλές, που πίστευα ότι ούτε καν ένα βιβλίο «for Dummies» δεν ήταν αναγκαίο. Μετά από δέκα πέντε και περισσότερα χρόνια που μελετώ αυτές τις μεθόδους, συνειδητοποιώ ότι έχουν γραφεί εκατοντάδες βιβλία για τις ευέλικτες μεθόδους, που παρουσιάζουν όλες τις δυνατές οπτικές γωνίες και απόψεις. Ο κοινός παρονομαστής σε όλα αυτά τα βιβλία είναι το άτομο, ο εργαζόμενος, η ομάδα, η συνεργασία και η επικοινωνία. Επιπλέον, συνειδητοποιώ ότι κάθε φορά που ανοίγω ένα νέο βιβλίο μαθαίνω κάτι καινούργιο, όχι τόσο στο τεχνικό ή στο διοικητικό επίπεδο αλλά στον τρόπο που οι ομάδες λειτουργούν και εργάζονται. Τέλος, παρατήρησα ότι ενώ υπάρχουν εκατοντάδες βιβλία στην αγγλική γλώσσα, δεν υπάρχει ούτε ένα στα ελληνικά, τουλάχιστον εγώ δεν γνωρίζω την ύπαρξη κάποιου.

- Εργαζόμενος στον χώρο της πληροφορικής για περισσότερα από τριάντα έτη έχω καταλήξει στο συμπέρασμα ότι τελικά τα τεχνικά προβλήματα δεν είναι η σημαντική πρόκληση. Και για να γίνω πιο σαφής, τα τεχνικά προβλήματα στην πλειοψηφία τους είναι δύο ειδών: αυτά για τα οποία υπάρχει μια τεχνική λύση και αυτά για τα οποία δεν υπάρχει μια δυνατή τεχνική λύση. Συνήθως ασχολούμαστε, εκτός από την περίπτωση των ερευνητικών έργων, με τα προβλήματα που μπορούμε να επιλύσουμε. Και ξανά εδώ στα προβλήματα που μπορούμε να επιλύσουμε υπάρχουν δύο κατηγορίες: τα προβλήματα των οποίων γνωρίζουμε τη λύση καιαυτά των οποίων η λύση δεν είναι γνωστή. Στην περίπτωση που δεν γνωρίζουμε τη λύση του προβλήματος είτε θα πρέπει να μελετήσουμε τη λύση αυτών των προβλημάτων, είτε να προσλάβουμε ένα έμπειρο εργαζόμενο που να γνωρίζει τη λύση τους, είτε τέλος να αναθέσουμε την εργασία αυτή σε μια εξωτερική οντότητα (υπεργολάβο). Τα ουσιαστικά προβλήματα που συναντάμε στην ανάπτυξη λογισμικού, είναι προβλήματα που αφορούν την οργάνωση της επιχείρησης και της ομάδας, την αποτελεσματική επικοινωνία, την εμπιστοσύνη μεταξύ των μελών της ομάδας, στην κατανόηση του πεδίου προβλήματος, στην καταγραφή των αναγκών του πελάτη, κ.λπ.

- Το Hollywood, στις ταινίες, παρουσιάζει τους προγραμματιστές λογισμικού ως εγωκεντρικούς, ατομικιστές, ιδιόρρυθμους κ.λπ. Η άποψή μου, αλλά και αυτή της επιστημονικής κοινότητας που ασχολείται με το θέμα, είναι ότι η ανάπτυξη λογισμικού είναι κυρίως μια κοινωνική δραστηριότητα. Επισημαίνουν ότι, χρειάζεται συνεχής επικοινωνία, ώστε η ομάδα έργου να κατανοήσει τις ανάγκες του προβλήματος, την αναγκαία αρχιτεκτονική, τις διεπαφές και διάφορες άλλες λεπτομέρειες του συστήματος. Για να ενισχύσω το παραπάνω επιχείρημα, θα αναφέρω το νόμο του Conway, ο οποίος παρατήρησε ότι η κοινωνική φύση της ανάπτυξης λογισμικού, απεικονίζεται ακόμη και στον σχεδιασμό του υποκείμενου συστήματος λογισμικού, το οποίο αντανακλά τις κοινωνικές σχέσεις μεταξύ των μελών της ομάδας ανάπτυξης.

- Οι επιχειρήσεις σήμερα έχουν ως προτεραιότητα την παραγωγή αξίας και θέλουν να καινοτομούν. Συνεπώς, αξιολογούν υψηλά εργαζόμενους που διαθέτουν καινοτομική σκέψη και

δημιουργικότητα. Για να προάγουμε τη δημιουργικότητα είναι σημαντικό ο εργαζόμενος να έχει ελεύθερο χρόνο ώστε να μπορεί να μελετά και να αναπτύσσει νέες ιδέες. Για παράδειγμα, οι εργαζόμενοι της εταιρείας Google έχουν στη διάθεσή τους, μια ημέρα την εβδομάδα, για αυτό το σκοπό. Ο εμπειρισμός (empiricism) που βρίσκεται στο κέντρο των ευέλικτων μεθόδων βοηθά προς αυτή την κατεύθυνση καθώς υποστηρίζει την αυτό-οργάνωση της ομάδας, τη γνώση που προέρχεται από την εμπειρία, την εφαρμογή καλών πρακτικών και βρίσκεται σε αντιδιαστολή με τα διαδικασιο-κεντρικά (process-oriented) συστήματα που προσδιορίζουν με ακρίβεια το κάθε βήμα που πρέπει να γίνει και πολλές φορές οδηγούν στη γραφειοκρατία.

- Σήμερα, η λειτουργία της κοινωνίας αλλά και όλη η οικονομική δραστηριότητα βασίζεται στη χρήση ψηφιακών συστημάτων. Η ορθή λειτουργία αυτών είναι προϋπόθεση για τη ομαλή λειτουργία των επιχειρήσεων αλλά και όλων των οργανισμών. Συνεπώς χρειαζόμαστε μηχανισμούς και εργαλεία για να εξασφαλίσουμε την απρόσκοπτη λειτουργία των ψηφιακών υπηρεσιών, αλλά και δυνατότητες να αλλάζουμε και να αποδεσμεύουμε το λογισμικό μέσα σε λίγα λεπτά της ώρας. Επομένως, οι έννοιες της συνεχούς ανάπτυξης λογισμικού αλλά και της συνεχούς παράδοσης αυτού κερδίζουν συνεχώς έδαφος. Το αποτέλεσμα είναι ότι οι χρήστες λογισμικού μετατρέπονται όλο και περισσότερο σε καταναλωτές λογισμικού, αφού το λογισμικό είναι διαθέσιμο αυτόματα, απαιτεί στη χρήση του ελάχιστες τεχνικές γνώσεις επιτρέποντάς τους να εστιαστούν αποκλειστικά στη χρήση του. Μπορούμε λοιπόν με σχετική βεβαιότητα να πούμε ότι ένα από τα βασικά χαρακτηριστικά του 21ου αιώνα θα είναι η συνεχής παράδοση του λογισμικού.Άλλωστε όλοι αρχίζουμε να συνειδητοποιούμε ότι ζούμε σε μια «software enabled society».
- Αν και οι ιδέες της ευελιξίας ξεκίνησαν από την ανάπτυξη λογισμικού, σήμερα, έχουν εφαρμοστεί και στις επιχειρήσεις. Οι επιχειρήσεις χρειάζεται να αναπτύξουν «ευέλικτες» ικανότητες, ώστε να μπορούν να ανταγωνιστούν αποτελεσματικά άλλες επιχειρήσεις, να χρησιμοποιούν την ψηφιακή τεχνολογία για την παραγωγή αξίας και να ανταποκρίνονται στις αλλαγές της αγοράς και στις αναδυόμενες ευκαιρίες. Οι επιθυμίες των πελατών, οι ανταγωνιστικές απειλές, οι τεχνολογικές επιλογές, οι επιχειρηματικές προσδοκίες, οι ευκαιρίες αύξησης των εσόδων/κερδών συμβαίνουν πλέον ταχύτατα και χρειαζόμαστε ευέλικτες μεθόδους οργάνωσης και διοίκησης. Εκτός όμως των επιχειρήσεων, η ευέλικτη προσέγγιση έχει εφαρμοστεί τόσο στο δημόσιο τομέα όσο και στην εκπαίδευση, με αξιοσημείωτα αποτελέσματα.

Εκτιμώ, ότι η γνώση των ευέλικτων ιδεών θα πρέπει να αποτελεί αντικείμενο μελέτης οπωσδήποτε για τους φοιτητές πληροφορικήςαυτούς και της διοίκησης επιχειρήσεων αλλά και πιθανά και για αυτούς άλλων ακαδημαϊκών κατευθύνσεων. Έχουμε διδάξει το συγκεκριμένο αντικείμενο επί σειρά ετών τόσο στο Πανεπιστήμιο Θεσσαλίας, στους φοιτητές του Τμήματος Διοίκησης Επιχειρήσεων, όσο και στους φοιτητές του προγράμματος «Πληροφορική» στο Ελληνικό Ανοικτό Πανεπιστήμιο. Η ανταπόκριση των φοιτητών ήταν εξαιρετικά θετική και άμεση.

Επίσης πιστεύω, χωρίς να είμαι θέση να το αποδείξω, ότι η ευέλικτη προσέγγιση είναι μια προσέγγιση που ταιριάζει στην Ελληνική κουλτούρα. Οι Έλληνες δεν αρέσκονται τις αυστηρές διαδικασίες, την αυστηρή οργάνωση και πάντα ψάχνουν έναν πιο εύκολο και αποδοτικό τρόπο για να κάνουν την εργασία τους. Διαθέτουν φαντασία, δημιουργικότητα και αν ενταχθούν σε μια ομάδα στην οποία εμπιστεύονται τα μέλη έχουν εξαιρετικά αποτελέσματα. Το γεγονός αυτό αποτύπωσε με εξαιρετικό τρόπο και ο δημιουργός Διονύσης Σαββόπουλος, στο τραγούδι του «Ας κρατήσουν οι χοροί»

Μέχρι τα ουράνια σώματα με πομπούς και με κεραίες

φτιάχνουν οι Έλληνες κυκλώματα κι ιστορία οι παρέες

Κλείνοντας, θα ήθελα να ευχαριστήσω τη Ευαγγελία Μπότη, υποψήφια διδάκτωρ του Πανεπιστημίου Θεσσαλίας για τις παρατηρήσεις της στο κείμενο, τον Διονύση Αδαμόπουλου, εξαιρετικό συνάδελφο στο Ελληνικό Ανοικτό Πανεπιστήμιο για τις παρατηρήσεις του επί της δομής του βιβλίου, καθώς και τη σύζυγό μου Σοφία Γερογιάννη για το χρόνο που μου εξασφάλισε για τη συγγραφή αυτού του βιβλίου. Ελπίζω να αποτελέσει ένα χρήσιμο βοήθημα για όσους επιχειρούν να κατανοήσουν και να εφαρμόσουν τις «ευέλικτες αξίες».

Λάρισα, Σεπτέμβριος 2022

Μέρος Α – Εισαγωγή στην έννοια της ευελιξίας

Εισαγωγή στις ευέλικτες μεθόδους

There is a tide in the affairs of men,
Which taken at the flood, leads on to fortune;
Omitted, all the voyage of their life
Is bound in shallows and in miseries.
On such a full sea are we now afloat,
And we must take the current when it serves,
Or lose our ventures.

Ουίλιαμ Σαίξπηρ – Ιούλιος Καίσαρας 4η Πράξη, 3η Σκηνή

Σύνοψη
Στο κεφάλαιο αυτό θα γίνει μια εισαγωγή στην ανάπτυξη λογισμικού και στα έργα λογισμικού. Στη συνέχεια θα δοθούν βασικοί ορισμοί για την ευελιξία στα έργα και γενικότερα στη διοίκηση. Θα παρουσιαστεί μια σύντομη ιστορική αναδρομή των ευέλικτων μεθόδων μαζί με τα βασικά χαρακτηριστικά της κάθε μεθόδου. Ακολουθεί η παρουσίαση του Agile Manifesto καθώς και η σύγκριση των παραδοσιακών μεθόδων με τις ευέλικτες. Το κεφάλαιο κλείνει με την παρουσίαση των πλεονεκτημάτων και μειονεκτημάτων των ευέλικτων μεθόδων.

1 Εισαγωγή στις ευέλικτες μεθόδους

Η ανάπτυξη λογισμικού, επίσης γνωστή ως μηχανική λογισμικού, ορίζεται στο σώμα της γνώσης λογισμικού, γνωστό και ως Software Engineering Body of Knowledge – SWBOK (Bourque & Fairley, 2014) ως τη «συστηματική, πειθαρχημένη, μετρήσιμη προσέγγιση για την ανάπτυξη, λειτουργία και συντήρηση του λογισμικού, και η μελέτη αυτών των προσεγγίσεων, με άλλα λόγια η εφαρμογή των αρχών της μηχανικής στην κατασκευή λογισμικού».

Η ανάπτυξη λογισμικού ως συστηματική διαδικασία μηχανικής εμφανίστηκε στη δεκαετία του 1950 με την ανάπτυξη των πρώτων λειτουργικών συστημάτων και σταδιακά έγινε κάτι πολύ διαδεδομένο. Ο ρόλος του λογισμικού στην σύγχρονη κοινωνία είναι εξαιρετικά σημαντικός. Η πρώτη, και ακόμα πολύ δημοφιλής, διαδικασία ανάπτυξης λογισμικού αναφέρεται ως το μοντέλο καταρράκτη (waterfall model) όπου η ανάπτυξη λογισμικού ακολουθεί γραμμικά τις φάσεις ανάπτυξης του συστήματος, δηλαδή ανάλυση, σχεδιασμός, κωδικοποίηση, έλεγχος και συντήρηση. Το μοντέλο καταρράκτη αποτελεί ένα από τα παλαιότερα μοντέλα ανάπτυξης λογισμικού. Σύμφωνα με το μοντέλο αυτό η ανάπτυξη λογισμικού είναι μια σειριακή διαδικασία, η οποία περνά από τις φάσεις της ανάλυσης των απαιτήσεων, του σχεδιασμού, της υλοποίησης, του ελέγχου, της ολοκλήρωσης και της συντήρησης. Το μοντέλο αυτό είναι από τα παλαιότερα (Royce, 1970) και, αν και είναι ευρέως γνωστό, δεν χρησιμοποιείται πλέον. Μάλιστα, ο Royce το παρουσίασε ως αντιπαράδειγμα και πρότεινε ένα επαναληπτικό μοντέλο ανάπτυξης. Θα πρέπει να σημειωθεί ότι το μοντέλο καταρράκτη δεν αναφέρεται στη διαχείριση ενός έργου γενικότερα, αλλά αποκλειστικά στην ανάπτυξη λογισμικού.

Παράλληλα, από το 1975 κιόλας, στο εμβληματικό του έργο «The Mythical Man-Month» ο Frederick Brooks είχε επισημάνει, με γλαφυρό τρόπο, την τάση των έργων πληροφορικής να αποτυγχάνουν (Brooks, 1995). Η τάση αυτή, που έχει στηριχτεί σε πληθώρα ερευνητικών ευρημάτων, έχει αποτελέσει αντικείμενο

ευρύτατης συζήτησης για αρκετές δεκαετίες και φαίνεται να είναι, σε κάποιο βαθμό, σύμφυτη με τη φύση των έργων πληροφορικής.

Η ίδια αυτή τάση αποτυπώνεται με ιδιαίτερη ένταση στις έρευνες του Standish Group με το όνομα «Chaos report», που από το 1995 και μετά, αξιολογούν την επιτυχία των έργων πληροφορικής. Τα συμπεράσματα στα οποία καταλήγουν είναι αποκαρδιωτικά: Στην πρώτη έρευνα (1994) που πραγματοποιήθηκε βρέθηκε ότι μόλις το 16% των έργων IT κατορθώνουν να ολοκληρωθούν έγκαιρα, χωρίς υπερβάσεις κόστους και να καλύπτουν άρτια το εύρος των προδιαγραφών του έργου (project scope). Τα τελευταία έτη η εικόνα που παρουσιάζουν οι ίδιοι δείκτες της έρευνας είναι βελτιωμένοι (με τα επιτυχημένα έργα να κυμαίνονται από 31% έως 39%), αλλά η γενικότερη εικόνα είναι πως τα έργα IT εξακολουθούν να χαρακτηρίζονται από υψηλά ποσοστά αποτυχίας.

Γενικότερα, τα έργα ανάπτυξης λογισμικού ή τα έργα ανάπτυξης πληροφοριακών συστημάτων έχουν αρκετές ομοιότητες με έργα άλλων ειδών, όπως, για παράδειγμα τα τεχνικά έργα. Στις περισσότερες περιπτώσεις οι στόχοι, οι διαδικασίες και τα εργαλεία που χρησιμοποιούνται είναι παρόμοια. Διαφοροποιείται όμως ο τρόπος εφαρμογής τους, λόγω των διαφορών που προκύπτουν από τη φύση του λογισμικού. Όπως έγραψε και ο Brooks (1995), το λογισμικό έχει ορισμένα χαρακτηριστικά που το διαφοροποιούν από άλλα προϊόντα. Μερικά από τα χαρακτηριστικά αυτά είναι (Bechtold, 1999; Hughes 1999; Κιουντουζής, 1999):

- Άυλο: Μπορούμε πολύ εύκολα να δούμε την πρόοδο που επιτελείται στην κατασκευή μιας γέφυρας, ενώ κάτι τέτοιο είναι αρκετά πιο δύσκολο στην ανάπτυξη λογισμικού.
- Πολύπλοκο: Κατά μέσο όρο το λογισμικό είναι πιο πολύπλοκο από άλλα προϊόντα αντίστοιχης τιμής. Κατ' ουσία, ενώ υπάρχει ένα μέγιστο ύψος που μπορεί να φτάσει μια γέφυρα, δεν υπάρχει μέγιστος αριθμός γραμμών κώδικα που μπορεί να έχει ένα σύστημα λογισμικού. Εδώ θα πρέπει να σημειώσουμε ότι η πολυπλοκότητα αυξάνεται με μη γραμμικό τρόπο σε σχέση με το μέγεθος του συστήματος.
- Εύπλαστο: Το γεγονός ότι το λογισμικό είναι άυλο, σημαίνει ότι έχει τη δυνατότητα να αλλάζει εύκολα και γρήγορα. Ταυτόχρονα μοντελοποιεί τον τρόπο εργασίας των ατόμων, γεγονός που οδηγεί σε πολλές και σύνθετες αλλαγές στο αντικείμενο των έργων. Έτσι, ενώ σε άλλα έργα η διαχείριση αλλαγών αποτελεί μια τετριμμένη διαδικασία, στα έργα ανάπτυξης λογισμικού αποτελεί βασική διαδικασία και πολλές φορές ενέχει κινδύνους.
- Διαθέσιμη τεχνολογία: Κανένας δεν θα διαφωνήσει με το γεγονός ότι η τεχνολογία εξελίσσεται με ραγδαίους ρυθμούς με συνέπεια τη δυσκολία διαχείρισης των τεχνολογικών αλλαγών. Αποτελεί συνηθισμένο φαινόμενο στα έργα ανάπτυξης λογισμικού η ύπαρξη προβλημάτων ολοκλήρωσης μεταξύ διαφορετικών εκδόσεων του λογισμικού, η ταυτόχρονη συνύπαρξη εργαλείων διαφορετικών εκδόσεων κ.λπ. Συνεπώς, η αλλαγή της τεχνολογίας αποτελεί για τα έργα ανάπτυξης λογισμικού έναν μόνιμο παράγοντα αστάθειας που θα πρέπει να λαμβάνουμε πάντα σοβαρά υπόψη.
- Πράγματι, στη βιβλιογραφία έχουν παρουσιαστεί διάφορες απόψεις σχετικά με τους παράγοντες που οδηγούν στο μεγαλύτερο βαθμό αποτυχίας.

Οι κυριότεροι είναι η άυλη φύση των έργων, που δυσχεραίνει την κοινή κατανόηση μεταξύ των εμπλεκομένων σε αυτά, το γεγονός ότι δεν υπάρχουν σωστές και λάθος απαντήσεις σε βασικά τους ζητήματα, το ότι αναπτύσσονται αλληλεξαρτήσεις μεταξύ των μερών τους, οι αποφάσεις που πρέπει να ληφθούν είναι πολλές και εμπεριέχουν μεγάλο ποσοστό ασάφειας και αβεβαιότητας, υπάρχει εγγενής δυσκολία ρεαλιστικών εκτιμήσεων κ.α. (Goatham, 2008; Al-Ahmad et al., 2009; Dwivedi, 2013)

Περισσότερο ενδιαφέρον έχει να παρατηρήσουμε στις ίδιες έρευνες του Standish Group, τους παράγοντες που, συγκριτικά, βοηθούν ένα έργο να είναι περισσότερο επιτυχημένο. Σε αυτούς θα βρούμε, τα τελευταία έτη, τη χρήση ευέλικτων (agile) μεθόδων ανάπτυξης λογισμικού ως έναν από τους κυριότερους παράγοντες βελτίωσης. Σε στοιχεία που προέρχονται από τη βάση του Standish Group από το 2002 έως το

2010, τα έργα που αναπτύσσονται με ευέλικτες μεθόδους θεωρούνται επιτυχημένα κατά 41%, σε αντίθεση με την παραδοσιακή, γραμμική μέθοδο του καταρράκτη (waterfall) με ποσοστό επιτυχίας μόλις 14%.

Βέβαια, η διάκριση αυτή είναι μάλλον απλουστευτική. Το δίπολο agile – waterfall είναι υπαρκτό αλλά όχι απόλυτο, καθώς οι μεθοδολογίες ανάπτυξης στην πραγματικότητα δεν είναι τόσο αυστηρά οριοθετημένες. Ακόμα όμως και αυτοί που έχουν αντιρρήσεις σχετικά με την απαισιόδοξη γενική εικόνα που παρουσιάζει το Standish Group, όπως ο Ambler, παραδέχονται ότι ο ορισμός της επιτυχίας του Standish Group είναι πολύ περιοριστικός (Ambler, 2014).

Στην αναγκαιότητα νέων τεχνικών, νέων πρακτικών, νέου υποδείγματος ανάπτυξης, που να αντιμετωπίζει τις δυσχέρειες και να εγγυάται (ή τουλάχιστον να υπόσχεται) καλύτερα αποτελέσματα, έρχεται να δώσει απάντηση η ευέλικτη αντιμετώπιση του ζητήματος.

Η εμφάνιση των ευέλικτων μεθόδων, ως ένα νέο υπόδειγμα ανάπτυξης λογισμικού, αλλά και ως ένα νέο μοντέλο διοίκησης, που μειώνει τα προβλήματα και την πολυπλοκότητα των διοικητικών διεργασιών και έργων, έγινε με τη διακήρυξη του Agile Manifesto το 2001, και μετρά ήδη σχεδόν 20 χρόνια. Στο χρονικό αυτό διάστημα, οι ευέλικτες μέθοδοι κέρδισαν ένα μεγάλο κοινό στην αγορά, με αποτέλεσμα να θεωρούνται ως ένα από τα κυρίαρχα υποδείγματα (West et al., 2010) και σίγουρα αυτό με τη μεγαλύτερη δυναμική ανάπτυξης. Η επιτυχής υιοθέτηση ευέλικτων μεθόδων από μία πληθώρα εταιρειών και οργανισμών κάθε μεγέθους, βοήθησε να εδραιωθεί η πεποίθηση ότι η ευέλικτη προσέγγιση αποτελεί την οδό προς την επιτυχία (Dybå, & Dingsøyr, 2008).

Ένα από τα κυριότερα διδάγματα του «The Mythical Man-Month» είναι το ότι δεν υπάρχει «μαγική» τεχνική να επιλυθούν τα προβλήματα των έργων πληροφορικής (η αρχή “No Silver Bullet”). Ακολουθώντας αυτό τον κανόνα, έτσι και η ευέλικτη προσέγγιση δεν αποτελεί εξαίρεση. Η επιτυχής εφαρμογή της προϋποθέτει σημαντική διαφοροποίηση σε σχέση με τον τρόπο και τις μεθόδους εργασίας, την οργάνωσή της, τις διαδικασίες που ακολουθούνται και, ίσως το σημαντικότερο, τον τρόπο σκέψης και την κουλτούρα ενός οργανισμού και των μελών του (Sahota, 2012). Το γεγονός αυτό από μόνο του κάνει την ευέλικτη προσέγγιση περισσότερο προσιτή ή μη, ανάλογα με το κατά πόσο οι οργανισμοί είναι έτοιμοι να την υπηρετήσουν.

1.1 Τι σημαίνει ευελιξία

Η λέξη ευελιξία ή ευκινησία (agility) συνήθως μας φέρνει στο μυαλό έναν αθλητή π.χ. ποδοσφαίρου που μπορεί να κινείται με ταχύτητα και να αλλάζει εύκολα κατεύθυνση, να κάνει τρίπλες, ή μια κατσίκα που κινείται με ταχύτητα σε απότομα βραχώδη περιβάλλοντα, ή μια γαζέλα που τρέχει στην άγρια φύση και αλλάζει κατεύθυνση όταν την κυνηγά ένας θηρευτής. Στην καθομιλουμένη κάποιος που είναι ευέλικτος μπορεί να κινηθεί γρήγορα αλλά αποφασιστικά, να αντιδράσει σε μεταβαλλόμενες καταστάσεις με ταχύτητα και πολύ καλό συντονισμό, να αλλάξει κατεύθυνση διατηρώντας την ισορροπία του. Αυτή η εικόνα έρχεται σε πλήρη αντίθεση με τις πιο παραδοσιακές μεθόδους διαχείριση έργων και ανάπτυξης λογισμικού, οι οποίες βασίζονται σε ένα καλά οργανωμένο πλάνο (plan based) όπου όλα ή τα περισσότερα δυνατόν είναι προσδιορισμένα εξαρχής.

Επεκτείνοντας τις παραπάνω παρατηρήσεις, μπορούμε να ορίσουμε ως ευελιξία (agility) την ικανότητα μίας οργανωτικής δομής να ανταποκρίνεται στις αλλαγές και να ανταποκρίνεται σε αυτές με ευρηματικότητα μέσα σε ένα ασταθές και μεταβαλλόμενο περιβάλλον (Lankhorst, 2012). Στην ανάπτυξη λογισμικού, η ευέλικτη προσέγγιση βασίζεται στην παραδοχή ότι οι απαιτήσεις του λογισμικού είναι δυναμικές και σπάνια μένουν οι ίδιες με αυτές που διαγνώστηκαν αρχικά.

Επιπλέον, η ευελιξία είναι ένας γενικός όρος που περιλαμβάνει επαναληπτικές προσεγγίσεις στην ανάπτυξη λογισμικού που υλοποιούν τις αξίες του manifesto για την ευέλικτη ανάπτυξη λογισμικού, το οποίο παρουσιάζεται λεπτομερέστερα αργότερα σε αυτό το κεφάλαιο.

Επιπρόσθετα, η ευέλικτη προσέγγιση, θεωρεί σημαντικό το να λαμβάνονται υπόψη και να συμμετέχουν ενεργά όλοι οι συμμετέχοντες (stakeholders) και το να τροφοδοτούνται διαρκώς και σε

σύντομα χρονικά διαστήματα με λειτουργικές εκδόσεις του λογισμικού για να το αξιολογούν. Αυτή η διαρκής παράδοση λειτουργικότητας επιτυγχάνεται με συνεργατικές προσεγγίσεις, που προκρίνουν την άμεση επικοινωνία και την ετοιμότητα για αλλαγή κατεύθυνσης και προτεραιοτήτων όποτε χρειάζεται. Τα μέσα που χρησιμοποιούνται είναι οι αυτοοργανωμένες ομάδες, που αποτελούνται από άτομα με πολλαπλές ικανότητες και ισχυρά κίνητρα, η διαρκής και ενεργός εμπλοκή των πελατών και των τελικών χρηστών του λογισμικού καθώς και η δημιουργία ενός αυτοματοποιημένου περιβάλλοντος ανάπτυξης και δοκιμών. Οι μικροί, επαναληπτικοί κύκλοι ανάπτυξης βοηθούν στην προσαρμογή στις αλλαγές.

Η ευελιξία (agility) περιλαμβάνει λοιπόν ένα σύνολο μεθόδων και μεθοδολογιών που βοηθούν την ομάδα έργου να σκέφτεται πιο αποτελεσματικά, να εργάζεται πιο αποτελεσματικά και να λαμβάνει καλύτερες αποφάσεις. Αυτές οι μέθοδοι και μεθοδολογίες καλύπτουν όλους τους τομείς της τεχνολογίας λογισμικού, συμπεριλαμβανομένης της διαχείρισης έργων, της σχεδίασης και της ανάπτυξης της αρχιτεκτονικής του λογισμικού, καθώς και τη βελτίωση του τρόπου εργασίας, δηλαδή των ακολουθούμενων διαδικασιών. Κάθε μία από αυτές τις μεθόδους και μεθοδολογίες αποτελείται από πρακτικές, που είναι εξορθολογισμένες και βελτιστοποιημένες για να τις εφαρμόσουμε όσο το δυνατόν ευκολότερα.

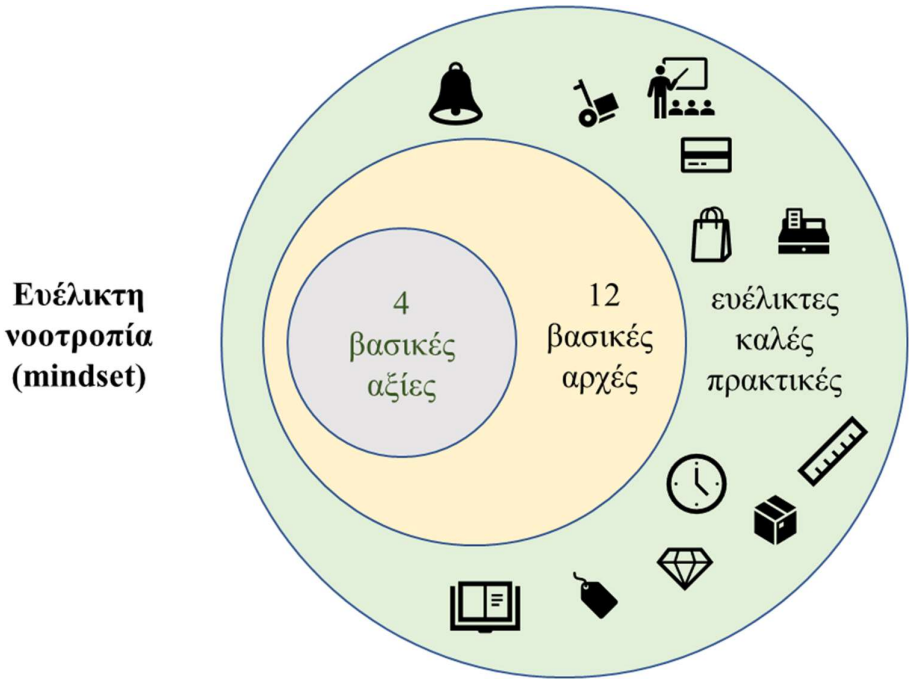
Η ευελιξία στην ανάπτυξη λογισμικού είναι επίσης μια νοοτροπία, διότι η σωστή νοοτροπία μιας ομάδας μπορεί να είναι η ειδοποιός διαφορά στο πόσο αποτελεσματικά μια ομάδα χρησιμοποιεί αυτές τις πρακτικές. Αυτή η νοοτροπία βοηθά τα άτομα μιας ομάδας να μοιράζονται πληροφορίες μεταξύ τους, έτσι ώστε να μπορούν να λαμβάνουν τις σημαντικές αποφάσεις έργου ως ομάδα - αντί να έχουν έναν διευθυντή που λαμβάνει όλες τις αποφάσεις μόνος του, να βελτιώνουν καθημερινά τον τρόπο εργασίας τους, να χρησιμοποιούν τις σωστές πρακτικές με ομοιόμορφο τρόπο, κ.λπ.

Από τα παραπάνω μπορεί να διαπιστωθεί ότι η ευέλικτη προσέγγιση αποτελεί σύστημα αξιών αλλά και πρακτικών και προϋποθέτει μία συμβατή με αυτή νοοτροπία και τρόπο σκέψης. Απαιτεί δε συγκεκριμένα χαρακτηριστικά και δυνατότητες από όλους τους εμπλεκόμενους και από τους οργανισμούς που την επιχειρούν και ιδιαίτερες προϋποθέσεις σε μία σειρά πεδίων (Οργάνωση, Κουλτούρα/Νοοτροπία, Επικοινωνία, Λειτουργίες, Ανθρώπινο δυναμικό κ.α.).

1.2 Η εμφάνιση των ευέλικτων μεθοδολογιών στην ανάπτυξη λογισμικού

Οι ευέλικτες μεθοδολογίες εμφανίστηκαν κατά τη δεκαετία του ’90. Αρχικά θεωρούνταν «ελαφρές» (lightweight) μέθοδοι ανάπτυξης λογισμικού με πειραματικό χαρακτήρα (Balter, 2011). Στα τέλη της δεκαετίας ήλθαν στο προσκήνιο λόγω αυξημένης προσοχής από την IT και κατόπιν από την ακαδημαϊκή κοινότητα, έχοντας επιδείξει αποτελεσματικότητα στην ανάπτυξη λογισμικού, σε αντίθεση με τα υψηλά ποσοστά αποτυχίας που παρουσίαζαν οι παραδοσιακές μέθοδοι. Ο όρος «ευέλικτες μεθοδολογίες» θεωρείται ως γενικός ή περιληπτικός όρος (umbrella term) που καλύπτει μια ποικιλία μεθοδολογιών. Όλες έχουν τα κοινά χαρακτηριστικά που αναφέρθηκαν προηγουμένως, αλλά η στόχευσή τους είναι διαφορετική (π.χ. η Scrum επικεντρώνεται στη διαχείριση έργων λογισμικού, ενώ η XP στην ταχεία ανάπτυξη ποιοτικού λογισμικού). Κάθε μέθοδος έχει τις δικές της κατευθυντήριες αρχές, μία σειρά από πρακτικές και ιδιαίτερες επαναληπτικές φάσεις ανάπτυξης. Στον Πίνακα 1.1 παρουσιάζονται ορισμένες από τις δημοφιλέστερες μεθοδολογίες και η χρονιά που παρουσιάστηκαν, αν και σε ορισμένες περιπτώσεις είναι δύσκολο να προσδιοριστεί μια συγκεκριμένη χρονιά αφού κάποιες από τις βασικές ιδέες στις οποίες βασίστηκαν, προϋπήρχαν.

Οι μεθοδολογίες ονομάστηκαν «ευέλικτες» όταν ορισμένοι από τους κυριότερους ειδικούς και πρωτοπόρους των μεθοδολογιών αυτών συναντήθηκαν προκειμένου να συζητήσουν για το μέλλον της ανάπτυξης λογισμικού. Αποτέλεσμα αυτής της συνάντησης ήταν η σύνταξη και υπογραφή ενός εμβληματικού κειμένου, του «Agile Manifesto» (2001), το οποίο περιείχε τις 4 βασικές αξίες και τις 12 αρχές για την ευέλικτη ανάπτυξη λογισμικού και που παρουσιάζεται στη συνέχεια (βλέπε Εικόνα 1.1). Το διαρκές ενδιαφέρον για την ευέλικτη προσέγγιση οδήγησε επίσης στην ίδρυση ενός επαγγελματικού οργανισμού από τους υπογράφοντες το κείμενο με το όνομα Agile Alliance.



Εικόνα 1.1: *Ευέλικτη νοοτροπία*

Μέθοδος	**Έτος παρουσίασης μεθόδου**
Dynamic Systems Development Method (DSDM) (Stapleton, 1997)	Το 1994 από εταιρίες παραγωγής λογισμικού
Scrum (Schwaber, 2004)	Το 1993 από τον Jeff Sutherland
CrystalClear (Cockburn, 2004)	Το 2004 από τους Scott Ambler και Robert Martin
Extreme Programming (XP) (Beck, 2005)	Το 1999 από τον Kent Beck
Feature-Driven Development (FDD) (Palmer & Felsing, 2001)	Το 1997 από τον Jeff De Luca
Agile Modeling (AM) (Ambler, 2002)	Το 2002 από τους Scott Ambler και Robert Cecil Martin
Lean Development (LD) (Poppendieck & Poppendieck, 2003)	2001 από τους Mary Poppendieck, Tom Poppendieck
Kanban (Anderson, 2012)	Το 2010 από τον David Anderson
Disciplined Agile (Ambler & Lines, 2012)	Το 2012 από τους Scott Ambler and Mark Lines
DevOps (Debois, 2011; Kim at al., 2016)	Το 2009 από τον Patrick Debois

Πίνακας 1.1: *Ευέλικτες μεθοδολογίες και πότε εμφανίστηκαν*

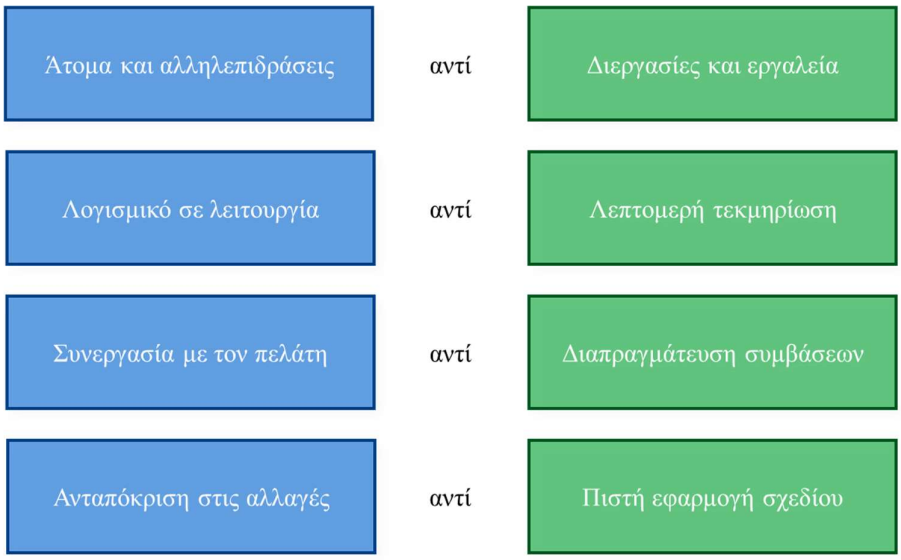
Από τον παραπάνω πίνακα φαίνεται ότι στην κατηγορία των ευέλικτων μεθόδων συμπεριλαμβάνονται μέθοδοι που εστιάζονται στη χρήση «λιτών» (lean) προσεγγίσεων αλλά και στη χρήση kanban.

Σύμφωνα με τη 13η ετήσια μελέτη της VersionOne (2019), η μέθοδος Scrum καθώς και παραλλαγές αυτής, είναι η πιο δημοφιλής μέθοδος αφού μπορεί να χρησιμοποιηθεί σε επαναληπτικά και αυξητικά έργα κάθε είδους. Στη μέθοδο Scrum, ο ιδιοκτήτης του προϊόντος (product owner) συνεργάζεται στενά με την ομάδα έργου για τον εντοπισμό και την ιεράρχηση της λειτουργικότητας του συστήματος, δημιουργώντας ένα κατάλογο χαρακτηριστικών του προϊόντος (product backlog). Το product backlog αποτελείται από όλα τα αναγκαία στοιχεία που χρειάζονται για να παραδώσουμε με επιτυχία ένα σύστημα λογισμικού. Στη συνέχεια εκτελούμε το έργο σε επαναλήψεις των 30 ή και λιγότερων ημερών, τα επονομαζόμενα Sprint.

Στα επόμενα κεφάλαια θα παρουσιαστούν με λεπτομέρεια κάποιες από τις παραπάνω μεθόδους.

1.3 Το ευέλικτο μανιφέστο: Αξίες, αρχές και οι συνέπειές του

Το Agile Software Development Manifesto εκδόθηκε από μια ομάδα προγραμματιστών και συμβούλων επιχειρήσεων το 2001 (Cockburn, 2006, Anderson, 2003, Highsmith, 2002) και εστιάζεται στην αναγνώριση του ανθρώπινου παράγοντα ως πρωταρχικού παράγοντα επιτυχίας ενός έργου λογισμικού, στη συνεχή έμφαση στην αποτελεσματικότητα, στην προσαρμοστικότητα και στη διαχείρισή της.



Πηγή: www.agilemanifesto.org

Εικόνα 1.2: *Οι βασικές αξίες των ευέλικτων μεθόδων*

Οι ευέλικτες μέθοδοι υπόσχονται ανταπόκριση στις αλλαγές, παραγωγικότερες πρακτικές και λιγότερη γραφειοκρατία. Το όνομα ευέλικτες αναφέρεται κυρίως στη συνολική τους ικανότητα να προσαρμόζουν κατάλληλα τη διαδικασία ανάπτυξης όταν προκύπτουν αλλαγές στην πορεία του έργου. Οι τέσσερις βασικές αξίες στις οποίες στοχεύουν οι ευέλικτες μέθοδοι είναι (Beck, 2005):

- Τα άτομα και οι αλληλεπιδράσεις είναι πιο σημαντικά από τις διαδικασίες και τα εργαλεία. Τονίζεται η ανάγκη για συνεργασία μεταξύ των συμμετεχόντων σε ένα έργο λογισμικού, κάτι που αποτελεί τον σημαντικότερο παράγοντα κατά την ανάπτυξη λογισμικού. Οι διαδικασίες και τα εργαλεία δεν είναι σημαντικά, αν η συνεργασία αυτή δεν είναι επιτυχής.
- Το λογισμικό που λειτουργεί είναι πιο σημαντικό από την ύπαρξη εκτενούς τεκμηρίωσης. Το μανιφέστο υπογραμμίζει τον πρωταρχικό στόχο του έργου, ο οποίος είναι η παραγωγή λειτουργικού λογισμικού. Οι προγραμματιστές παρακινούνται να δημιουργούν κώδικα όσο πιο απλό γίνεται, με αποτέλεσμα το ίδιο το λογισμικό να είναι πιο κατανοητό σε σχέση με μακροσκελή έγγραφα και διαγράμματα τεκμηρίωσης. Η ολοκληρωμένη και σωστή τεκμηρίωση είναι σημαντική για να εξηγεί τη λειτουργία του λογισμικού, αλλά ο ρόλος της θα είναι πάντα δευτερεύων και συμπληρωματικός.
- Η συνεργασία με τον πελάτη είναι πιο σημαντική από τις συμβατικές διαπραγματεύσεις. Η συνεργασία και η επικοινωνία της ομάδας ανάπτυξης του λογισμικού με τον πελάτη είναι προτιμητέα σε σύγκριση με την εφαρμογή και την αναδρομή στους όρους των συμβάσεων. Η συνεργασία με τον πελάτη είναι συνήθως δύσκολη, αφού οι πελάτες αλλάζουν συχνά άποψη σχετικά με τις απαιτήσεις της εφαρμογής ή ακόμη δεν γνωρίζουν τι πρέπει να γίνει. Οι δυσκολίες

αυτές θα πρέπει να αντιμετωπίζονται με επικοινωνία και συνεργασία και όχι με την αναφορά στους όρους της σύμβασης.

- Η ανταπόκριση στην αλλαγή είναι πιο σημαντική από την τήρηση ενός προδιαγεγραμμένου σχεδίου. Οι συμμετέχοντες στο έργο πρέπει να είναι καλά πληροφορημένοι, έτοιμοι να εξετάσουν και να αντιμετωπίσουν τις ανάγκες που θα προκύψουν κατά τη διάρκεια του έργου. Οι απαιτήσεις των πελατών, το επιχειρηματικό περιβάλλον, ακόμα και η ίδια η τεχνολογία, μεταβάλλονται συνεχώς και δεν αφήνουν ανεπηρέαστη την ανάπτυξη του πληροφοριακού συστήματος. Γι' αυτό θα πρέπει η ανάπτυξη να ικανοποιεί άμεσα την ανάγκη για αλλαγές. Αυτό δεν σημαίνει ότι απορρίπτεται η ύπαρξη του χρονοδιαγράμματος του έργου, αλλά ότι θα πρέπει να είναι περισσότερο ευέλικτο.

Η διατύπωση των παραπάνω αξιών αναδεικνύει τους τομείς, στους οποίους επικεντρώνει το ενδιαφέρον της η ευέλικτη προσέγγιση: Είναι ανθρωποκεντρική, τονίζοντας τη σημασία του ανθρώπινου παράγοντα και της αμεσότητας της συνεργασίας των ενδιαφερομένων μεταξύ τους. Είναι επίσης πελατοκεντρική, θέτοντας τον πελάτη και τις ανάγκες του στο επίκεντρο της διαδικασίας. Για το λόγο αυτό είναι προσανατολισμένη στην αξία που έχει το τελικό αποτέλεσμα για τον πελάτη και τους χρήστες του. Αυτή δεν είναι άλλη από το ουσιαστικό αποτέλεσμα της διαδικασίας, το λειτουργικό λογισμικό, που αποτελεί και τη μόνο πραγματική αξία για τον πελάτη, πολύ περισσότερο απ' ότι η δαπάνη πόρων για την πλήρη τεκμηρίωση του έργου, προκρίνοντας τόση μόνο τεκμηρίωση, όση χρειάζεται για την παρακολούθησή του. Και η διαρκής επιδίωξη της αξίας, καθιστά απαραίτητη την ετοιμότητα για αλλαγή πλάνου, σε περίπτωση που οι απαιτήσεις αλλάξουν για οποιονδήποτε λόγο. Η επιπλέον σημασία που δίνεται στα πρώτα στοιχεία αξίας έναντι των δεύτερων έχει την έννοια της προτίμησης. Αν τεθεί ανάλογο δίλημμα, η επιλογή πρέπει να είναι υπέρ των πρώτων.

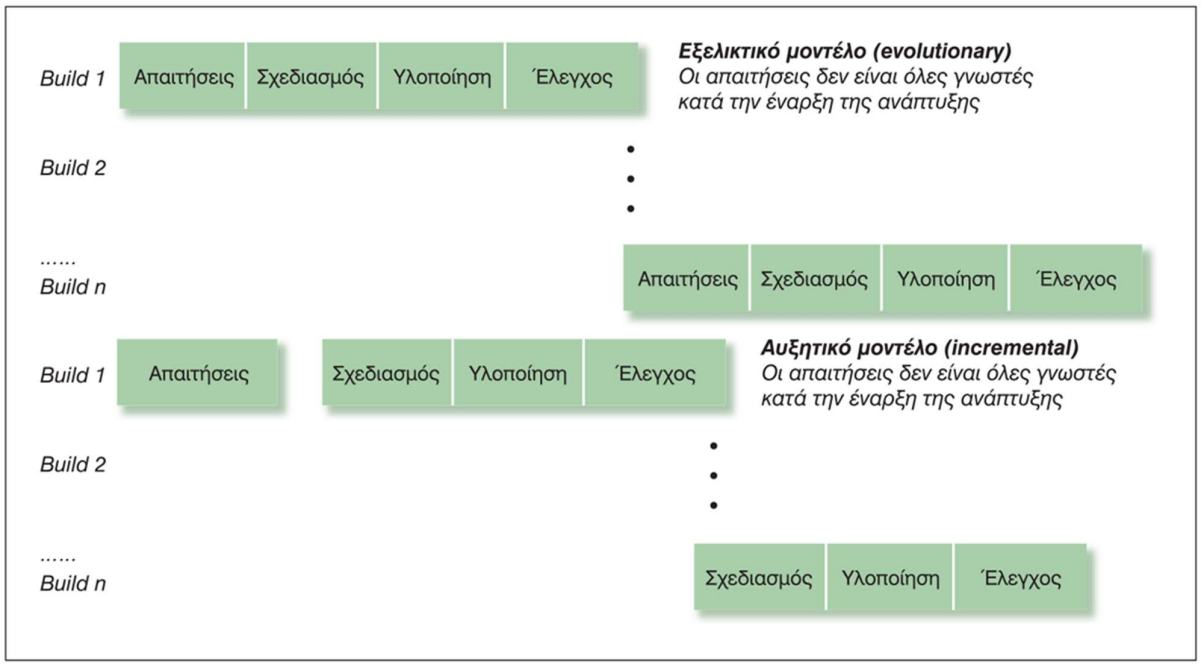
Στο μανιφέστο των ευέλικτων μεθόδων καθορίστηκαν επιπλέον δώδεκα αρχές που καθοδηγούν τον τρόπο ανάπτυξης. Παρουσιάζονται όπως διατυπώθηκαν (Beck, 2005):

1. Βασική προτεραιότητα αποτελεί η ικανοποίηση του πελάτη με τη συνεχή παράδοση σημαντικού τμήματος του λογισμικού από τα πρώτα κιόλας στάδια της παραγωγής (επικέντρωση στον πελάτη).
2. Οι μεταβαλλόμενες απαιτήσεις είναι καλοδεχούμενες ακόμα και σε προχωρημένο στάδιο ανάπτυξης. Οι διαδικασίες στην ανάπτυξη λογισμικού είναι προσαρμοσμένες στην αλλαγή με στόχο την ενίσχυση του ανταγωνιστικού πλεονεκτήματος του πελάτη. (αποδοχή της αλλαγής).
3. Η παράδοση τμημάτων του λογισμικού γίνεται ανά δύο εβδομάδες έως και ανά δύο μήνες. Σημαντική είναι η όσο το δυνατόν συχνότερη παράδοση (συχνή παράδοση λειτουργικότητας).
4. Οι προγραμματιστές και οι ειδικοί της αγοράς πρέπει να συνεργάζονται καθημερινά καθ' όλη τη διάρκεια του έργου (καθημερινή συνεργασία).
5. Θεμελιώνουμε τα έργα γύρω από άτομα με πάθος και ενδιαφέρον. Διαμορφώνουμε το κατάλληλο περιβάλλον, τους παρέχουμε την αναγκαία υποστήριξη και εμπιστευόμαστε την ικανότητά τους να φέρουν σε πέρας την αποστολή τους. (ατομική παρακίνηση).
6. Η πιο αποδοτική και αποτελεσματική μέθοδος για τη μετάδοση πληροφορίας προς και εντός της ομάδας ανάπτυξης λογισμικού είναι η συνομιλία πρόσωπο με πρόσωπο (άμεση επικοινωνία).
7. Το λογισμικό που λειτουργεί είναι το κύριο μέτρο προόδου (επικέντρωση στην αξία).
8. Οι ευέλικτες διαδικασίες προάγουν την αειφόρο ανάπτυξη. Οι χορηγοί, η ομάδα ανάπτυξης λογισμικού και οι χρήστες θα πρέπει να είναι σε θέση να διατηρούν ένα σταθερό ρυθμό επ' αόριστον (σταθερή απόδοση).
9. Η διαρκής έμφαση στην τεχνική αρτιότητα και στην εύρυθμη σχεδίαση ενισχύουν την ευελιξία (επιδίωξη της ποιότητας).
10. Η απλοποίηση, με την έννοια της υλοποίησης στόχων σε σύντομο αλλά και με αποτελεσματικό τρόπο, είναι βασική αρχή (επιδίωξη της απλότητας).

11. Οι καλύτερες αρχιτεκτονικές, απαιτήσεις και σχέδια προκύπτουν από ομάδες που αυτοοργανώνονται (αυτο-οργάνωση).
12. Σε τακτά χρονικά διαστήματα η ομάδα συζητά τρόπους, που την βοηθούν να γίνει περισσότερο αποτελεσματική και επαναπροσδιορίζει τη συμπεριφορά της (διαρκής βελτίωση).

Η τήρηση των βασικών αυτών αρχών του μανιφέστου μειώνουν την πιθανότητα να εμφανιστούν κίνδυνοι και λάθη κατά τη διαδικασία ανάπτυξης του λογισμικού. Συνοψίζοντας, οι ευέλικτες μέθοδοι είναι (Cohen et al., 2004):

- Επαναληπτικές (iterative). Παραδίδεται αρχικά ένα πλήρες σύστημα, το οποίο αποτελείται από επιμέρους υποσυστήματα. Σε κάθε επόμενη έκδοση γίνονται αλλαγές στη λειτουργία κάθε υποσυστήματος.
- Αυξητικές (incremental) και εξελικτικές (evolutionary). Σε κάθε νέα έκδοση προστίθενται νέες λειτουργίες στις ήδη υπάρχουσες.
- Προκύπτουσες (emergent). Οι αποφάσεις σχετικά με τις απαιτήσεις και την τεχνολογία που θα χρησιμοποιηθεί λαμβάνονται κατά τη διάρκεια του κύκλου ανάπτυξης.
- Αυτοοργανωμένες(self organizing). Η ομάδα έχει την ελευθερία της αυτο-οργάνωσης.



Εικόνα 1.3: *Το αυξητικό και εξελικτικό μοντέλο ανάπτυξης λογισμικού*

1.4 Σχέση διαχείρισης έργων και ανάπτυξης λογισμικού

Η ανάπτυξη λογισμικού είναι μια συστηματική και οργανωμένη διεργασία, η οποία τις περισσότερες φορές γίνεται από μια ομάδα ατόμων στο πλαίσιο ενός έργου. Πολλές φορές και από πολλούς, η έννοια της διαχείρισης ενός έργου λογισμικού και του κύκλου ζωής λογισμικού δεν είναι ξεκάθαρες και είναι συγκεχυμένες, αφού πολλοί δεν κατανοούν το στόχο της κάθε διεργασίας. Πριν απαντήσουμε λοιπόν σε αυτό το ερώτημα, δηλαδή ποια είναι η σχέση της διαχείρισης έργων με τον κύκλο ζωής λογισμικού και κατ' επέκταση με την ανάπτυξη λογισμικού, θα πρέπει πρώτα να ορίσουμε τι είναι έργο καθώς και διαχείριση έργων.

Σύμφωνα με τους παραδοσιακούς ορισμούς που έχουν δοθεί έργο είναι ένα προσωρινό εγχείρημα που στοχεύει στη δημιουργία ενός μοναδικού προϊόντος ή υπηρεσίας. Στον ορισμό αυτό:

- Προσωρινό σημαίνει ότι κάθε έργο έχει καθορισμένη έναρξη και λήξη.
- Μοναδικό σημαίνει ότι το προϊόν ή η υπηρεσία διαφέρει κατά διακριτό τρόπο από όλα τα παρόμοια προϊόντα ή υπηρεσίες.

Ο ορισμός αυτός αναφέρεται γενικότερα σε έργα και όχι μόνο σε έργα πληροφορικής. Οι παραπάνω ιδιότητες των έργων, να είναι δηλαδή προσωρινά αλλά και μοναδικά εγχειρήματα, έρχονται σε αντίθεση με τη δομή των περισσότερων επιχειρήσεων που λειτουργούν βάσει διαδικασιών που έχουν σταθερό και μόνιμο χαρακτήρα. Η διαχείριση αυτών των ιδιοτήτων είναι συχνά δύσκολη επειδή απαιτεί συνδυασμό ιδιαίτερων ικανοτήτων που προέρχονται από διαφορετικά γνωστικά πεδία.

Έτσι, η πρώτη πρόκληση που αντιμετωπίζουμε στη διαχείριση έργων είναι να εξασφαλίσουμε ότι το έργο εκτελείται σωστά και παραδίδεται έγκαιρα λαμβάνοντας υπόψη καθορισμένους περιορισμούς. Οι περιορισμοί αυτοί μπορεί να είναι ο ανεπαρκής διαθέσιμος χρόνος, ο περιορισμένος προϋπολογισμός κ.ά. Η δεύτερη πρόκληση, που είναι και η πιο φιλόδοξη, είναι η βελτιστοποίηση που απαιτείται να γίνει σε όλους αυτούς τους παράγοντες που επηρεάζουν την εκτέλεση ενός έργου. Αυτό σημαίνει ότι σε ένα έργο επιλέγουμε τις δραστηριότητες που απαιτούνται με τέτοιον τρόπο, ώστε να γίνεται η βέλτιστη χρήση των πόρων (χρόνος, χρήματα, άνθρωποι, υλικά, μηχανήματα, ενέργεια, χώρος κ.ά.).

Έτσι καταλήγουμε σε έναν δεύτερο ορισμό για το έργο:

Έργο είναι ένα εγχείρημα κατά το οποίο ανθρώπινοι πόροι, μηχανές, οικονομικοί πόροι και πρώτες ύλες οργανώνονται κατά καινοφανή τρόπο, με στόχο την ανάληψη συγκεκριμένου αντικειμένου εργασιών που έχουν συγκεκριμένες προδιαγραφές και υπόκεινται σε δεδομένους κοστολογικούς και χρονικούς περιορισμούς, ώστε να παραχθεί μια επωφελής μεταβολή, η οποία ορίζεται μέσω ποσοτικών και ποιοτικών στόχων (PMI, 2017).

Στην επόμενη λίστα παρουσιάζουμε τα βασικά χαρακτηριστικά ενός έργου:

- Αποτελείται από μη επαναλαμβανόμενες δραστηριότητες οι οποίες, σε μια τυπική περίπτωση, μπορούν να περιγραφούν από τον κύκλο ζωής του λογισμικού.
- Απαιτείται σχεδιασμός ώστε να επιτύχουμε το τελικό αποτέλεσμα.
- Το τελικό αποτέλεσμα είναι μοναδικό.
- Η εκτέλεση του έργου απαιτεί την ύπαρξη ομάδας.
- Έχει έναρξη και λήξη.
- Υπόκειται σε περιορισμούς διαφόρων ειδών (χρόνου, κόστους, ποιότητας κ.ά.).
- Οι διαθέσιμοι πόροι είναι περιορισμένοι.

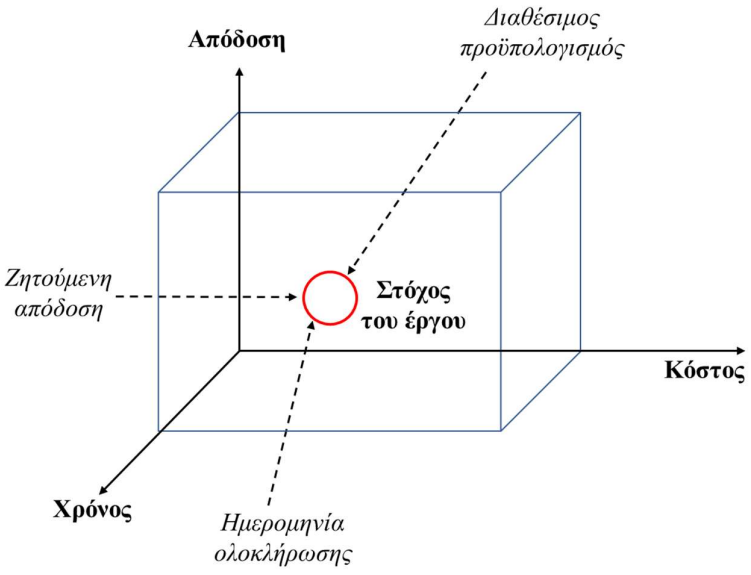
Αντίστοιχα, ο κύκλος ζωής ανάπτυξης λογισμικού (Software Development Life Cycle - SDLC) είναι ένα εννοιολογικό μοντέλο που χρησιμοποιείται στη διαχείριση έργου και περιγράφει τα στάδια που απαιτούνται σε ένα έργο ανάπτυξης λογισμικού, από την αρχική μελέτη σκοπιμότητας, την υλοποίηση του συστήματος, έως και τη λειτουργία αυτού. Ένα SDLC εστιάζεται συνήθως και παρουσιάζει αναλυτικά όλα τα βήματα που πρέπει να εκτελέσει ένας μηχανικός λογισμικού, ώστε να αναπτύξει το λογισμικό, ή γενικότερα το πληροφοριακό σύστημα.

Αυτή η ανάγκη για οργάνωση της γνώσης γύρω από τη διαχείριση έργων πληροφορικής, οδήγησε στη δημιουργία προτύπων και πλαισίων (frameworks) που καθορίζουν το πεδίο αυτό. Πρότυπα ανάπτυξης λογισμικού και πρότυπα διαχείρισης έργου δημιουργούν ένα πλέγμα σχέσεων με σκοπό τον έλεγχο και το συντονισμό όλων των διαθέσιμων πόρων, προκειμένου να επιτευχθούν οι στόχοι του έργου. Συνηθέστερα η μεθοδολογία διαχείρισης έργου (PM) αποτελεί ένα «κέλυφος» για τη μεθοδολογία ανάπτυξης λογισμικού (SD) και αντίστοιχα ο κύκλος ζωής της διαχείρισης έργου (Project Management Life Cycle - PMLC)

περιλαμβάνει, στο μεγαλύτερο τμήμα του, τον κύκλο ζωής ανάπτυξης λογισμικού (Software Development Life Cycle - SDLC). Η διάσταση της διαχείρισης έχει μεγάλη σημασία, καθώς η κακή διαχείριση έργου είναι μία από τις βασικές αιτίες που αποτυγχάνουν τα έργα Πληροφορικής. Παρά την εντύπωση ότι τα παραδοσιακά frameworks για διαχείριση έργων (όπως τα Project Management Body of Knowledge - PMBOK και PRINCE2) δεν μπορούν να υποστηρίξουν τις ευέλικτες μεθοδολογίες, στην πραγματικότητα οι προσεγγίσεις αυτές συχνά είναι συμβατές, με τις κατάλληλες τροποποιήσεις. Για παράδειγμα, στον πρακτικό οδηγό για εφαρμογή ευέλικτων μεθόδων παρουσιάζεται ο συνδυασμός των παραδοσιακών μεθόδων διαχείρισης έργων με τις ευέλικτες μεθόδους (Alliance, A. G. I. L. E., 2017), αναφέρεται στην προσαρμογή του προτύπου για να εξυπηρετεί την ευέλικτη ανάπτυξη, ενώ για την PRINCE 2 υποστηρίζεται ότι μπορεί να συνυπάρξει κάλλιστα με τις ευέλικτες μεθόδους αν προσαρμοστεί ανάλογα, καθώς η προσαρμογή ("tailoring" στην ορολογία της μεθόδου) είναι από τα βασικά της χαρακτηριστικά (Measey, 2013).

Ένα έργο πληροφορικής είναι το αποτέλεσμα δύο διεργασιών, όχι πάντα διακριτών, που βρίσκονται σε μία στενή, αλληλεπίδραση: της ανάπτυξης λογισμικού (software development) και της διαχείρισης του έργου (project management). Έτσι, η ανάπτυξη του λογισμικού ως διεργασία περιλαμβάνει όλες αυτές τις δραστηριότητες που είναι διαφορετικές σε ένα έργο λογισμικού (π.χ. τη συλλογή απαιτήσεων, τη συγγραφή κώδικα) σε σχέση με ένα έργο άλλου τύπου (π.χ. κατασκευαστικό έργο). Από την άλλη πλευρά, η διαχείριση του έργου ως μια επιχειρηματική διεργασία είναι κοινή ή έχει πολλές ομοιότητες σε όλα τα έργα, ανεξάρτητα του αντικειμένου τους και περιλαμβάνει δραστηριότητες όπως διαχείριση των πόρων, των ενδιαφερομένων, την οικονομική διαχείριση κ.λπ. Βέβαια, τόσο η ανάπτυξη λογισμικού όσο και η διαχείριση του έργου, εξετάζονται από κοινού γιατί η προσέγγιση που ακολουθείται (waterfall ή agile) επηρεάζει σημαντικά και τις δύο αυτές οπτικές γωνίες.

Ένας άλλος τρόπος με τον οποίο επιχειρείται να καταδειχθεί η χρησιμότατα της ευέλικτης προσέγγισης, στο πεδίο της διαχείρισης έργων αυτή τη φορά, είναι η αναφορά στο «σιδερένιο τρίγωνο» (iron triangle) των περιορισμών στη διαχείριση έργων, ένα εννοιολογικό σχήμα που χρησιμοποιείται για να εξηγήσει τους περιορισμούς στη διαχείριση έργων, παρουσιάζοντάς την ως μία διαρκή διαπραγμάτευση στο πλαίσιο ενός τρισδιάστατου συστήματος με πλευρές το κόστος, το δεδομένο εύρος (scope) και τον χρόνο που απαιτεί ένα έργο. Οποιαδήποτε παρέμβαση στο ένα σκέλος φέρνει αναπόφευκτα αλλαγές στα άλλα δύο. Δεν μπορεί κανείς π.χ. να αλλάξει το εύρος ενός έργου χωρίς να επιφέρει αλλαγές στο κόστος και το χρονικό πλάνο του. Η ευέλικτη προσέγγιση επιχειρεί να σπάσει αυτό το «σιδερένιο τρίγωνο» αποδεχόμενη ότι η αλλαγή απαιτήσεων, χρονικών περιορισμών κ.λπ. είναι αναπόσπαστα δεμένη με την ανάπτυξη προϊόντων λογισμικού και, συνεπώς, θα πρέπει να βρούμε ευέλικτους τρόπους να τη διαχειριστούμε (βλέπε Εικόνα 1.4).

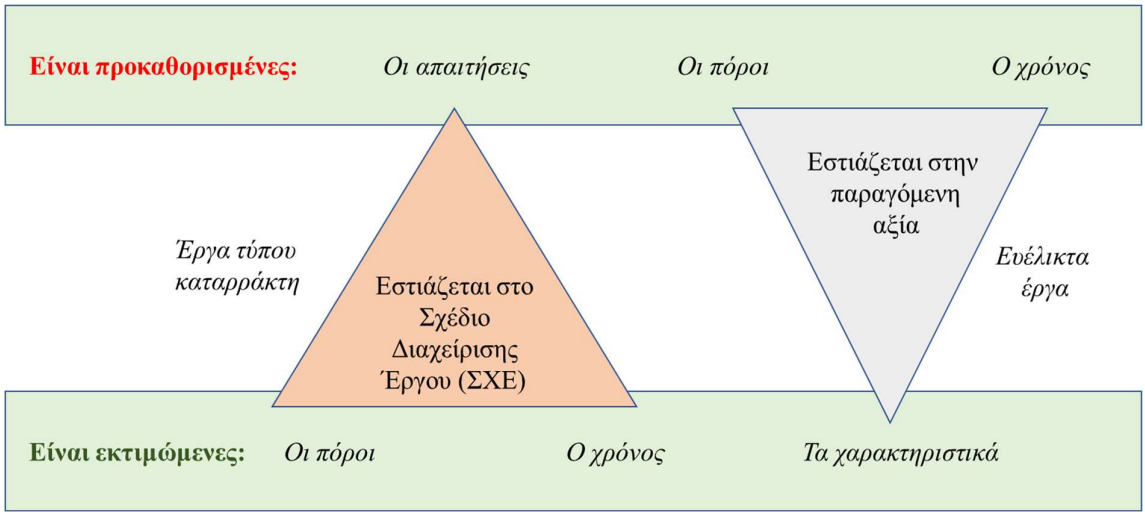


/

Εικόνα 1.4: *Οι τρεις βασικοί περιορισμοί σε ένα έργο*

Οι παραδοσιακές λοιπόν μέθοδοι είναι καθοδηγούμενες από ένα λεπτομερές σχέδιο, το Σχέδιο Διαχείρισης Έργων (ΣΧΕ) – Project Management Plan που έχει προδιαγραφεί στην αρχική φάση του έργου (project initiation phase) και που έχει προκύψει από την ανάλυση των απαιτήσεων. Συνεπώς το έργο εστιάζεται στην ικανοποίηση αυτών των περιορισμών (plan driven), δηλαδή προσπαθεί να ικανοποιήσει τις απαιτήσεις χρόνου και κόστους που θα απαιτηθεί για το δεδομένο εύρους απαιτήσεων.

Η ευέλικτη προσέγγιση, από την άλλη, αντιστρέφει το τρίγωνο, προσπαθώντας να παραδώσει τη μεγαλύτερη δυνατή αξία (π.χ. την περισσότερη λειτουργικότητα) σε δεδομένο χρόνο και με δεδομένο κόστος (value driven). Έτσι, το εύρος του έργου είναι μεταβλητό, για να μπορεί να ανταποκρίνεται καλύτερα στις μεταβολές των συνθηκών. Αυτή η αλλαγή υποδείγματος απεικονίζεται στο σχήμα της Εικόνας 1.5.



Εικόνα 1.5: *Η αλλαγή υποδείγματος στην ευέλικτη διαχείριση έργων*

1.5 Ευέλικτες και παραδοσιακές μεθοδολογίες (“agile vs plan based”)

Οι θιασώτες των ευέλικτων μεθόδων ορίζουν ως παραδοσιακές όλες τις μεθοδολογίες που προϋπήρχαν. Αυτές περιλαμβάνουν το παραδοσιακό μοντέλο καταρράκτη (waterfall model), το V-model, το μοντέλο σπείρας (spiral model), την ενοποιημένη προσέγγιση (Unified Process) και άλλες, λιγότερο διαδεδομένες. Οι μεθοδολογίες αυτές, παρότι όχι όλες τόσο περιοριστικές όσο το μοντέλο καταρράκτη (π.χ. το μοντέλο σπείρας υποστηρίζει μία επαναληπτική και επαυξητική διαδικασία, όπως και οι ευέλικτες μέθοδοι, με σκοπό τη διαρκή εκτίμηση των κινδύνων και του κόστους του έργου), εντούτοις δεν μπορούν να χαρακτηριστούν ευέλικτες, γιατί διαθέτουν ορισμένα βασικά χαρακτηριστικά που δεν συνάδουν με την ευέλικτη προσέγγιση, όπως π.χ. την ανάγκη για εκτενή αρχικό σχεδιασμό, τη δυσκολία αλλαγής των προδιαγραφών ή τη μη συνεργατική ανάπτυξη. Η τάση για επισήμανση των διαφορών ανάμεσα στα δύο υποδείγματα παίρνει συχνά τη μορφή του απλουστευτικού διλήμματος “agile vs waterfall”.

Στον Πίνακα 1.2 συνοψίζονται οι κυριότερες διαφορές παραδοσιακών και ευέλικτων μεθόδων ανάπτυξης λογισμικού, όπως έχουν αποτυπωθεί στη βιβλιογραφία. Διαπιστώνουμε για μία ακόμη φορά ότι οι ευέλικτες μέθοδοι προϋποθέτουν σημαντικές αλλαγές σε αξίες και πρακτικές στο πλαίσιο ενός οργανισμού, πολύ περισσότερο, αν αυτός έχει δημόσιο χαρακτήρα.

Ιδιότητες	**Παραδοσιακές**	**Ευέλικτες**
Κυρίαρχη συμπεριφορά	Τα συστήματα είναι απόλυτα προσδιορισμένα, προβλέψιμα και αναπτύσσονται με λεπτομερή και εκτενή προγραμματισμό. Αντίστοιχα η διαχείριση του έργου γίνεται με	Το λογισμικό είναι υψηλής ποιότητας, προσαρμόσιμο και χρησιμοποιεί τις αρχές τις συνεχούς βελτίωσης του σχεδιασμού, των συχνών δοκιμών, ενώ βασίζεται στις γρήγορες αλλαγές και την ανατροφοδότηση. Αντίστοιχα η διαχείριση του

Ιδιότητες	Παραδοσιακές	Ευέλικτες
	βάση τις διεργασίες (process-oriented)	έργου είναι εστιασμένη στους ανθρώπους (people-centric)
Μέγεθος έργου	Συνήθως μεγάλο	Μικρό
Μέγεθος ομάδας / νοοτροπία	Μεγάλο / πειθαρχία	Μικρό / καινοτομία
Μοντέλο διαχείρισης έργου	Συγκεντρωτικό όπου ο διαχειριστής έργου δίνει εντολές και υπάρχει αυστηρός έλεγχος.	Αποκεντρωμένο, βασισμένο στην ομάδα, στην συνεργασία και στη συλλογικότητα.
Απαιτήσεις	Σταθερές και σαφώς καθορισμένες και τεκμηριωμένες από την αρχή	Ανακαλύπτονται κατά τη διάρκεια του έργου με πολλές πιθανές αλλαγές
Στάση ως προς τις αλλαγές	Αντίσταση στις αλλαγές	Ετοιμότητα για αλλαγές
Τεκμηρίωση	Εκτενής	Περιορισμένη και αφαιρετική
Αρχικός σχεδιασμός	Εκτενής	Περιορισμένος
Κύκλος Ζωής`	Καταρράκτη, σπείρας, ενοποιημένη προσέγγιση	Επαναληπτικός, αυξητικός και βασισμένος σε user stories
Οργανωσιακή κουλτούρα	Ιεραρχικός έλεγχος	Συνεργατική ηγεσία
Ρόλος του πελάτη	Χαμηλή συμμετοχή και παθητικός ρόλος	Μεγάλη συμμετοχή και ενεργός ρόλος
Αντικείμενο προσοχής	Διαδικασίες	Άνθρωποι
Επικοινωνία	Τυπική	Άτυπη
Ποιοτικός έλεγχος	Εκτενής προς το τέλος του έργου	Διαρκής και ενσωματωμένος
Παράδοση αποτελέσματος / παραλαβή αξίας (Return of Investment)	Στο τέλος του έργου	Σταδιακά και από τα αρχικά στάδια

Πίνακας 1.2: *Σύγκριση παραδοσιακών και ευέλικτων μεθοδολογιών*

Λαμβάνοντας υπόψη τα παραπάνω, αλλά και τα πλεονεκτήματα και τους περιορισμούς που παρουσιάζουν τόσο οι παραδοσιακές, όσο και οι ευέλικτες μέθοδοι ανάπτυξης λογισμικού αντιλαμβάνεται κανείς ότι, κάτω από συγκεκριμένες απαιτήσεις, κάθε μεθοδολογία είναι περισσότερο ή λιγότερο κατάλληλη, και για το λόγο αυτό πρέπει να επιλέγονται ιδιαίτερα προσεκτικά.

Στη σύγχρονη εποχή, ωστόσο, οι ευέλικτες μέθοδοι ανάπτυξης λογισμικού βρίσκουν εφαρμογή σε ολοένα και περισσότερα πεδία και κλάδους, όπως είναι ο κλάδος της διαχείρισης ανθρώπινου δυναμικού, η οργάνωση και η διοίκηση επιχειρήσεων, η παραγωγή προϊόντων, αλλά και η δημόσια διοίκηση.

1.6 Προσκλήσεις στην εφαρμογή των ευέλικτων μεθόδων

Η εφαρμογή των ευέλικτων μεθόδων είναι κάτι περισσότερο από την απλή υιοθέτηση μιας δεδομένης μεθόδου για ένα μεμονωμένο έργο. Απαιτεί σχέδιο, δέσμευση της διοίκησης, εκπαίδευση προσωπικού κ.λπ. Σύμφωνα με τη βιβλιογραφία, τα πιο σημαντικά εμπόδια στην υιοθέτηση των ευέλικτων μεθόδων είναι τα ακόλουθα (Hajjdiab & Taleb, 2011; Mahanti, 2006):

- Αντίσταση στην αλλαγή. Η αλλαγή της κουλτούρας ενός οργανισμού και των εργαζομένων αποτελεί πάντα μια σημαντική πρόκληση.
- Η έλλειψη υποστήριξης από τη διοίκηση είναι ένας από τους κύριους λόγους για τους οποίους η ευέλικτη προσέγγιση δεν εφαρμόζεται σε πολλούς οργανισμούς. Για να εφαρμοστούν οι ευέλικτες μέθοδοι πρέπει όλα τα στελέχη, τα μεσαία αλλά και τα ανώτερα, να γνωρίζουν ότι θα υπάρξουν αλλαγές στις πρακτικές διαχείρισης έργων. Πρέπει να κατανοήσουν τα οφέλη του επερχόμενου ευέλικτου μετασχηματισμού, καθώς και τις λεπτομέρειες για το πώς αυτός ο μετασχηματισμός θα

επηρεάσει τις λειτουργικές πτυχές της επιχείρησης. Συνεπώς για να υποστηρίξουν επαρκώς την εισαγωγή των ευέλικτων μεθόδους πρέπει να κατανοήσουν πλήρως τι αναμένεται από αυτές.

- Έλλειψη ιδιοκτησίας από την ομάδα ανάπτυξης. Η αντίσταση ορισμένων ομάδων να υιοθετήσουν τις ευέλικτες πρακτικές αποτελεί ένα μεγάλο εμπόδιο στην επιτυχία της μετάβασης σε ευέλικτες μεθόδους. Για να αλλάξει αυτή η κουλτούρα θα πρέπει να ενδυναμωθεί η ομάδα και τα μέλη της ομάδας να αναλάβουν την πλήρη ευθύνη για την ιδιοκτησία της εργασίας τους. Θα πρέπει να εγκαταλείψουν τη συνήθεια να εξαρτώνται από κέντρα αποφάσεων εξωτερικά της ομάδας. Αυξάνοντας την ιδιοκτησία της ευέλικτης διαδικασίας από τα μέλη της ομάδας, τους δίνεται η ελευθερία να αναλύσουν και να επιλέξουν λύσεις μόνοι τους κάθε φορά που αντιμετωπίζουν προβλήματα, αντί να σπαταλούν πολύτιμο χρόνο περιμένοντας έγκριση από τη διοίκηση του οργανισμού.

- Έλλειψη κατάρτισης και εκπαίδευσης. Η ανεπαρκής εκπαίδευση στις ευέλικτες μεθόδους είναι μεταξύ των κορυφαίων λόγων για τους οποίους αποτυγχάνει η ευέλικτη μετάβαση. Επιπλέον, υπάρχει ανάγκη για εξειδικευμένες δεξιότητες που οδηγεί στην υπερ-εξειδίκευση. Τα μέλη της ομάδας ανάπτυξης πολλές φορές έχουν εξαιρετικά εξειδικευμένες δεξιότητες σε έναν συγκεκριμένο τομέα, ενώ έχουν ελάχιστες ή καθόλου γνώσεις σε άλλους τομείς της ανάπτυξης λογισμικού. Για παράδειγμα, οι διαχειριστές έργων δεν κατανοούν επαρκώς τις υποκείμενες τεχνολογίες που χρησιμοποιούνται από τις ομάδες τους, οι προγραμματιστές δεν έχουν δεξιότητες ανάλυσης και σχεδιασμού, κ.λπ. Αντίθετα οι ευέλικτες ομάδες απαιτούν μέλη με εξειδικευμένες δεξιότητες σε έναν ή περισσότερους τομείς καθώς και βασική κατανόηση των τεχνικών και επιχειρηματικών πτυχών της ανάπτυξης λογισμικού.

- Σειριακή σκέψη: Πολλοί επαγγελματίες πληροφορικής έχουν συνηθίσει σε σειριακές προσεγγίσεις π.χ. μοντέλο ζωής καταρράκτη και είναι μη δεκτικοί σε νέες επαναληπτικές και αυξητικές προσεγγίσεις. Αυτό μπορεί να αποδοθεί στο γεγονός ότι τα τελευταία 50 χρόνια κυριαρχήθηκαν από μεθοδολογίες ανάπτυξης λογισμικού που χρησιμοποιούν σειριακές προσεγγίσεις. Αυτοί οι εργαζόμενοι θέλουν να προσδιορίσουν πρώτα τις πλήρεις απαιτήσεις, μετά να σχεδιάσουν το σύστημα και μόνο στο τέλος να ξεκινήσουν την κωδικοποίηση. Προφανώς πρέπει να υπάρξει κατάλληλη εκπαίδευση, και στοχευμένη καθοδήγηση για να κατανοήσουν τις αρχές της ευέλικτης ανάπτυξης. Ταυτόχρονα, θα πρέπει να είναι κανείς σε εγρήγορση για να βεβαιωθεί ότι η σειριακή νοοτροπία δεν εμποδίζει την εισαγωγή και τη διατήρηση ευέλικτων πρακτικών στην επιχείρηση.

- Κακή επικοινωνία και συνεργασία. Η επικοινωνία παίζει καθοριστικό ρόλο στην ευέλικτη προσέγγιση. Τα μέλη της ομάδας πρέπει να επικοινωνούν συνεχώς και αποτελεσματικά ώστε το κάθε έργο να αναπτυχθεί σωστά. Για να γίνει αυτό, απαιτείται η δημιουργία κατάλληλων καναλιών επικοινωνίας από την επιχείρηση και, κυρίως, για κατανεμημένες ομάδες. Οπωσδήποτε, η παρουσία των μελών της ομάδας στον ίδιο χώρο διευκολύνει την άμεση ροή πληροφοριών και την ανατροφοδότηση. Ωστόσο, στην πραγματικότητα που ζούμε σήμερα, πάνω από το 50% όλων των εργαζομένων στις επιχειρήσεις πληροφορικής παγκοσμίως θα εργάζονται εξ αποστάσεως τα επόμενα χρόνια. Με άλλα λόγια, ο αυξανόμενος αριθμός κατανεμημένων ομάδων και ατόμων που εργάζονται από τα σπίτια τους απαιτεί νέους έξυπνους τρόπους για να εξομαλύνουμε την επικοινωνία. Ο νέος κανόνας δεν είναι πλέον η «πρόσωπο με πρόσωπο» επικοινωνία αλλά η εξασφάλιση της δυνατότητας για «άμεση» επικοινωνία και συνεργασία.

Προφανώς η παραπάνω λίστα είναι ενδεικτική αλλά δείχνει ξεκάθαρα πόσο σημαντική είναι για την εισαγωγή της ευελιξίας η κατάλληλη κουλτούρα και εκπαίδευση του προσωπικού. Η παραπάνω παρατήρηση βρίσκεται σε πλήρη αντιστοιχία με το ευέλικτο μοντέλο ευχέρειας (agile fluency model) (Larsen et al., 2015) που παρουσιάζει τον τρόπο με τον οποίο οι ευέλικτες ομάδες αναπτύσσουν νέες ικανότητες (βλέπε Εικόνα 1.6)

Εικόνα 1.6: *Ευέλικτο μοντέλο ευχέρειας*

Μια επιτυχημένη ομάδα ξεκινά αρχικά ως ένα ασύνδετο σύνολο ατόμων με συμπληρωματικές τεχνικές δεξιότητες. Καθώς η ομάδα υιοθετεί ευέλικτες πρακτικές, λαμβάνει χώρα μια αλλαγή κουλτούρας της ομάδας. Έτσι αντί η ομάδα να εστιάζεται στα τεχνικά ζητήματα, όπως τη δομή και τα αρθρώματα του λογισμικού, η ομάδα εστιάζεται στο πως θα μεγιστοποιήσει το όφελος της επιχείρησης, του πελάτη ή του χρήστη, επιδεικνύοντας ευχέρεια εστίασης (focusing fluency). Οι ομάδες που βρίσκονται στη ζώνη της εστίασης ασχολούνται με θέματα όπως:

- Πως ανταποκρινόμαστε στις επιχειρηματικές ανάγκες;
- Πόσο καλά κατανοούμε την επιχειρηματική αξία στις προδιαγραφές του προϊόντος;
- Πως επικοινωνούμε την εργασία μας με τον πελάτη του προϊόντος μας;
- Πως αλλάζουμε κατεύθυνση όταν αλλάζουν οι επιχειρηματικές ανάγκες;
- Πως γινόμαστε πιο αποτελεσματικοί ως ομάδα;
- Πως ενισχύουμε την ψυχολογία της ομάδας μας;

Το βασικό κέρδος αυτής της κατάστασης είναι η μεγαλύτερη διαφάνεια στην εργασία των ομάδων και η μεγαλύτερη δυνατότητα ανακατεύθυνσης που έχουν.

Μόλις η ομάδα αποκτήσει την ευχέρεια εστίασης, τότε θα πρέπει να δοθεί έμφαση στην ανάπτυξη των τεχνικών δυνατοτήτων της ομάδας. Η ανάπτυξη των τεχνικών πρακτικών απαιτεί μεγαλύτερη επένδυση και συνήθως, περισσότερο χρόνο. Προφανώς κατά τη διάρκεια της ανάπτυξης των τεχνικών ικανοτήτων της ομάδας υπάρχει μειωμένη παραγωγικότητα. Μόλις εξαλειφθούν οι τεχνικοί περιορισμοί στην ανάπτυξη λογισμικού, η ομάδα έχει αποκτήσει ευχέρεια παράδοσης (delivery fluency). Το βασικό σημείο βελτίωσης σε αυτή την κατάσταση είναι η μεγαλύτερη παραγωγικότητα της ομάδας και τα λιγότερα παραγόμενα σφάλματα.

Το επόμενο βήμα είναι η ομάδα να αναπτύξει δυνατότητες βελτιστοποίησης που της επιτρέπουν να προσαρμόζεται στις ανάγκες της αγοράς, να αλλάζει την κατεύθυνσή της, να μαθαίνει γρηγορότερα, κ.λπ. Επιπλέον, όταν η επιχείρηση αλλάζει και μετακινεί βασικές επιχειρηματικές ή τεχνικές ικανότητες εντός της ομάδας ανάπτυξης, και δίνει τη δυνατότητα στον ομάδα ανάπτυξης να αποφασίζει, η ομάδα μπορούμε να πούμε ότι επιδεικνύει ευχέρεια βελτιστοποίησης (optimization fluency). Το κέρδος από την ευχέρεια βελτιστοποίησης είναι οι καλύτερες αποφάσεις που λαμβάνονται σχετικά με το προϊόν και η μεγαλύτερη αξία που παραδίδεται στον πελάτη (Shore & Warden, 2021).

Όταν οι ομάδες αναπτύξουν σημαντικές ικανότητες, είτε τεχνικές, είτε γνώσης της αγοράς, είτε η ομάδα καινοτομεί τότε η επιχείρηση μπορεί να επωφεληθεί από την εμπειρία της ομάδας και η γνώση αυτή να κεφαλοποιηθεί. Η φάση αυτή ονομάζεται φάση της ισχυροποίησης (strengthening) και έχει ως

αποτελέσματα την διατομεακή συνεργασία μέσα στην επιχείρηση, την αριστεία, και την οργανωτική βελτίωση (Fowler, 2018).

Η βελτίωση της ευχέρειας στην ευελιξία δεν είναι κάτι μόνιμο και όπως αποκτάται μπορεί να απολεσθεί. Η πιο κοινή αιτία απώλειας της ευχέρειας ευελιξίας είναι όταν η νέα διοίκηση αποφασίζει ότι οι ευέλικτες προσεγγίσεις δεν ταιριάζουν με το όραμά τους. Χωρίς υποστήριξη της ανώτατης διοίκησης, η ευχέρεια της ομάδας διαβρώνεται γρήγορα. Αυτό συχνά συνοδεύεται από απώλεια τεχνογνωσίας καθώς δυσαρεστημένα μέλη της ομάδας αναζητούν νέες θέσεις εργασίας.

Βιβλιογραφία/Αναφορές

Al-Ahmad, W., Al-Fagih, K., Khanfar, K., Alsamara, K., Abuleil, S., & Abu-Salem, H. (2009). A taxonomy of an IT project failure: root causes. International Management Review, 5(1), 93-104.

Agile Alliance (2017). Agile Practice Guide, 1st ed. Newtown Square, NM, USA:Project Management Institute.

Ambler S. (2002). Agile Modelling: Effective Practices for Extreme Programming and the Unified Process. 2002.

Ambler S.W., Lines M (2012). Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enter- prise (IBM Press).

Ambler, S. (2014). The non-existent software crisis: Debunking the chaos report. Dr. Dobb's.

Anderson, D. J. (2012). Lessons in agile management: on the road to Kanban. Blue Hole Press.

Balter, B. J. (2011). Toward a more agile government: The case for rebooting federal IT procurement. Public Contract Law Journal, 149-171.

Bechtold P. (1999). Essentials of Software Project Management. Management Concepts.

Beck K. (2005). Extreme Programming Explained: Embrace Change (2nd ed.). Addison Wesley.

Bourque, P., & Fairley, R. E. (2014). SWEBOK, version 3.0: Guide to the Software Engineering Body of Knowledge. IEEE Computer Engineering Society.

Brooks Jr, F. P. (1995). The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, (2nd ed.). Pearson Education India.

Chervenkova, M. (2021). Why Agile Doesn't Work in Every Case & What Are the Top Agile Challenges?. Ανακτήθηκε 8η Αυγούστου, 2022 από https://kanbanize.com/blog/agile-challenges/

Clancy, T. (1995). The Standish group report. Chaos report.

Cockburn A. (2006). Agile Software Development: The Cooperative Game (2nd ed.). Addison-Wesley.

Debois, P. (2011). Devops: A software revolution in the making. *Journal of Information Technology Management*, *24*(8), 3-39.

Fowler, M. (2018). The Agile Fluency Model. Ανακτήθηκε 2η Σεπτεμβρίου, 2022 από https://martinfowler.com/articles/agileFluency.html

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations*. IT Revolution.

Dwivedi, Y. K., Ravichandran, K., Williams, M. D., Miller, S., Lal, B., Antony, G. V., & Kartik, M. (2013, June). IS/IT project failures: a review of the extant literature for deriving a taxonomy of failure factors. In International working conference on transfer and diffusion of IT (pp. 73-88). Springer, Berlin, Heidelberg.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and software technology, 50(9-10), 833-859.

Goatham, R. (2009). The story behind the high failure rates in the IT sector. Calleam Consulting.

Hajjdiab, H., & Taleb, A. S. (2011). Adopting agile software development: issues and challenges. *International Journal of Managing Value and Supply Chains (IJMVSC)*, *2*(3), 1-10.

Hastie, S., & Wojewoda, S. (2015). Standish group 2015 Chaos report-Q&A with Jennifer Lynch. Retrieved, 1(15), 2016.

Hughes B. & Cotterell M. (1999). Software Project Management. McGraw Hill.

Lankhorst, M. (Ed.). (2012). Agile service development: combining adaptive methods and flexible solutions. Springer Science & Business Media.

Larsen, D., Holyer, S., Eckstein, J., Kirjavainen, A., & Sorje, O. (2015, May). Practical applications of the agile fluency model. In *Agile Processes in Software Engineering and Extreme Programming: 16th International Conference, XP 2015* (Vol. 212, p. 339).

Mahanti, A. (2006). Challenges in enterprise adoption of agile methods-A survey. *Journal of Computing and Information technology*, *14*(3), 197-206.

Measey, P. (2013). Agile and the Best Management Practice framework within the public sector.

Palmer S.R. & Felsing M. (2001). A practical guide to feature-driven development. Pearson Education.

PMI Institute. (2017). A guide to the Project Management Body of Knowledge, (6th ed.).. PMI Standard Committee

Poppendieck, M., & Poppendieck, T. (2003). Lean Software Development: An Agile Toolkit: An Agile Toolkit. Addison-Wesley.

Royce W.W. (1970). Managing the development of large software systems. In Proceedings of IEEE WESCON (Vol. 26, No. 8).

Sahota, M. (2012). An Agile Adoption and Transformation Survival Guide. Lulu. com.

Schwaber K. (2004). Agile project management with Scrum. Microsoft press.

Shore, J., & Warden, S. (2021). *The art of agile development*. " O'Reilly Media, Inc.".

Stapleton J. (1997). DSDM, dynamic systems development method: the method in practice. Cambridge University Press.

VersionOne, C. 13th Annual State of Agile report (2018).

West, D., Grant, T., Gerush, M., & D'Silva, D. (2010). Agile development: Mainstream adoption has changed agility. Forrester Research, 2(1), 41.

Κιουντουζής Ε. (1999). Διαχείριση Έργων Πληροφορικής. Εκδόσεις Σταμούλη.

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Πως διαφοροποιούνται τα έργα λογισμικού σε σχέση με άλλα έργα.

Απάντηση/Λύση

Η ανάπτυξη λογισμικού έχει ορισμένα χαρακτηριστικά που το διαφοροποιούν από άλλα προϊόντα. Μερικά από τα χαρακτηριστικά αυτά είναι:

- Άυλο: Μπορούμε πολύ εύκολα να δούμε την πρόοδο που επιτελείται στην κατασκευή μιας γέφυρας, ενώ κάτι τέτοιο είναι αρκετά πιο δύσκολο στην ανάπτυξη λογισμικού.
- Πολύπλοκο: Κατά μέσο όρο το λογισμικό είναι πιο πολύπλοκο από άλλα προϊόντα αντίστοιχης τιμής. Κατ' ουσίαν, ενώ υπάρχει ένα μέγιστο ύψος που μπορεί να φτάσει μια γέφυρα, δεν υπάρχει μέγιστος αριθμός γραμμών κώδικα που μπορεί να έχει ένα σύστημα λογισμικού. Εδώ θα πρέπει να σημειώσουμε ότι η πολυπλοκότητα αυξάνεται με μη γραμμικό τρόπο σε σχέση με το μέγεθος του συστήματος.
- Εύπλαστο: Το γεγονός ότι το λογισμικό είναι άυλο, σημαίνει ότι έχει τη δυνατότητα να αλλάζει εύκολα και γρήγορα. Ταυτόχρονα μοντελοποιεί τον τρόπο εργασίας των ατόμων, γεγονός που οδηγεί σε πολλές και σύνθετες αλλαγές στο αντικείμενο των έργων. Έτσι, ενώ σε άλλα έργα η διαχείριση αλλαγών αποτελεί μια τετριμμένη διαδικασία, στα έργα ανάπτυξης λογισμικού αποτελεί βασική διαδικασία και πολλές φορές ενέχει κινδύνους.
- Διαθέσιμη τεχνολογία: Κανένας δεν θα διαφωνήσει με το γεγονός ότι η τεχνολογία εξελίσσεται με ραγδαίους ρυθμούς έχοντας ως συνέπεια τη δυσκολία διαχείρισης των τεχνολογικών αλλαγών. Αποτελεί συνηθισμένο φαινόμενο στα έργα ανάπτυξης λογισμικού η ύπαρξη προβλημάτων ολοκλήρωσης μεταξύ διαφορετικών εκδόσεων του λογισμικού, η ταυτόχρονη συνύπαρξη εργαλείων διαφορετικών εκδόσεων κ.λπ. Συνεπώς, η αλλαγή της τεχνολογίας αποτελεί για τα έργα ανάπτυξης λογισμικού έναν μόνιμο παράγοντα αστάθειας που θα πρέπει να λαμβάνουμε πάντα σοβαρά υπόψη.

Κριτήριο αξιολόγησης 2

Μελετήστε πώς διαφέρει η διαχείριση έργων λογισμικού από τη διαχείριση τεχνικών έργων. Συγκρίνετε την ανάπτυξη ενός λογισμικού οικονομικής διαχείρισης με το κτίσιμο μιας γέφυρας.

Απάντηση/Λύση

Για να δώσουμε την απάντηση θα πρέπει να σκεφτούμε αναλυτικά διάφορους παράγοντες που επηρεάζουν τη διαχείριση ενός έργου. Ο παρακάτω πίνακας παρουσιάζει τον τρόπο με τον οποίο διαφοροποιούνται τα έργα ανάπτυξης λογισμικού από άλλα έργα.

	Έργο ανάπτυξης συστήματος λογισμικού	**Έργο άλλου τύπου (π.χ. κτίσιμο γέφυρας)**
Δομή έργου	Σχετίζεται άμεσα με επιχειρηματικές διαδικασίες και σίγουρα με άλλα συστήματα λογισμικού της επιχείρησης (μισθοδοσία, διαχείριση πελατών, κ.λπ.).	Είναι σχετικά ανεξάρτητο από επιχειρηματικές διαδικασίες. Πιθανόν να συνδέεται με άλλα έργα κατασκευής οδών πρόσβασης.
Αντικείμενο του έργου	Στις περισσότερες περιπτώσεις τέτοιων έργων δεν είναι καλά ορισμένο και υπάρχει	Καλά ορισμένο. Το σχέδιο του αναμενόμενου αποτελέσματος (π.χ. η γέφυρα) χαρακτηρίζεται

	Έργο ανάπτυξης συστήματος λογισμικού	**Έργο άλλου τύπου (π.χ. κτίσιμο γέφυρας)**
	μια γενική ιδέα και ασαφείς απαιτήσεις. Είναι βέβαιο ότι θα υπάρχουν πολλές αλλαγές.	από ικανοποιητικό επίπεδο λεπτομέρειας. Συνήθως υπάρχει προμελέτη και μελέτη εφαρμογής.
Διαχείριση αλλαγών	Στα περισσότερα έργα λογισμικού υπάρχει διαδικασία διαχείρισης αλλαγών, αλλά είναι δύσκολο να συσχετίσουμε τις αλλαγές με το τελικό προϊόν.	Είναι καλά ορισμένη.
Επάνδρωση του έργου	Υπάρχει μεγάλος αριθμός διαφορετικών ειδικοτήτων που έχει ως αποτέλεσμα το προσωπικό του έργου να είναι μερικής απασχόλησης.	Οι εργαζόμενοι στο έργο είναι πλήρους απασχόλησης. Χρησιμοποιούνται εκτενώς υπεργολάβοι.
Διαχείριση κινδύνου	Οι κίνδυνοι δεν προσδιορίζονται εύκολα και συνήθως έχουν μεγάλο αντίκτυπο μια και συνήθως εντοπίζονται αργά.	Οι κίνδυνοι προσδιορίζονται σχετικά εύκολα και η διαχείρισή τους είναι τυποποιημένη. Υπάρχει σχετική νομοθεσία.
Διαχείριση ποιότητας	Τεκμηριωμένη αλλά δύσκολη στην εφαρμογή της.	Είναι πιο εύκολη η εφαρμογή τυποποιημένων ελέγχων.

Κριτήριο αξιολόγησης 3

Αναφέρατε μερικές από τις πιο γνωστές ευέλικτες μεθοδολογίες

Απάντηση/Λύση

Το κείμενο που προσδιορίζει τις ευέλικτες αξίες και αρχές είναι το «Agile Manifesto» ενώ μερικές από τις πιο γνωστές και δημοφιλείς μεθοδολογίες είναι:

- Dynamic Systems Development Method (DSDM) (Stapleton, 1997)
- Scrum (Schwaber, 2004)
- CrystalClear (Cockburn, 2004)
- Extreme Programming (XP) (Beck, 2005)
- Feature-Driven Development (FDD) (Palmer & Felsing, 2001)
- Agile Modeling (AM) (Ambler, 2002)
- Lean Development (LD) (Poppendieck & Poppendieck, 2003)
- Kanban (Anderson, 2012)
- Disciplined Agile (Ambler & Lines, 2012)
- DevOps (Debois, 2011; Kim at al., 2016)

Κριτήριο αξιολόγησης 4

Ποια κατά τη γνώμη σας είναι τα 4 βασικά χαρακτηριστικά των ευέλικτων μεθόδων.

Απάντηση/Λύση

Θα μπορούσε να αναφέρει κανείς ως απάντηση τις τέσσερις βασικές αξίες των ευέλικτων μεθόδων που είναι:

- Τα άτομα και οι αλληλεπιδράσεις είναι πιο σημαντικά από τις διαδικασίες και τα εργαλεία.

- Το λογισμικό που λειτουργεί είναι πιο σημαντικό από την ύπαρξη εκτενούς τεκμηρίωσης.
- Η συνεργασία με τον πελάτη είναι πιο σημαντική από τις συμβατικές διαπραγματεύσεις.
- Η ανταπόκριση στην αλλαγή είναι πιο σημαντική από την τήρηση ενός προδιαγεγραμμένου σχεδίου.

Εναλλακτικά θα μπορούσε να αναφερθεί ότι οι ευέλικτες μέθοδοι έχουν τα παρακάτω χαρακτηριστικά:

- Επαναληπτικές (iterative). Παραδίδεται αρχικά ένα πλήρες σύστημα, το οποίο αποτελείται από επιμέρους υποσυστήματα. Σε κάθε επόμενη έκδοση γίνονται αλλαγές στη λειτουργία κάθε υποσυστήματος.
- Αυξητικές (incremental) και εξελικτικές (evolutionary). Σε κάθε νέα έκδοση προστίθενται νέες λειτουργίες στις ήδη υπάρχουσες.
- Προκύπτουσες (emergent). Οι αποφάσεις σχετικά με τις απαιτήσεις και την τεχνολογία που θα χρησιμοποιηθεί λαμβάνονται κατά τη διάρκεια του κύκλου ανάπτυξης.
- Αυτοοργανωμένες(self organizing). Η ομάδα έργου σε μια ευέλικτη μέθοδο έχει την ελευθερία της αυτο-οργάνωσης.

Κριτήριο αξιολόγησης 5

Στο ευέλικτο μανιφέστο χρησιμοποιείται η λέξη «αντί» (over). Για παράδειγμα:

Individuals and interactions <u>over</u> processes and tools

Ποιο είναι το νόημα της λέξης «αντί» (over);

Απάντηση/Λύση

Η λέξη over υποδηλώνει μια σχέση προτεραιότητας η οποία αποδίδεται ενδεικτικά με εκφράσεις όπως "έρχεται πριν" ή "προτιμάται" ή "περισσότερο από". Αυτό αναφέρεται στο τέλος του κειμένου του μανιφέστου όταν αναφέρεται "εκτιμούμε περισσότερο τα στοιχεία στα αριστερά".

Δηλαδή στο συγκεκριμένο παράδειγμα αναφέρεται ότι δίνουμε έμφαση στα άτομα και τις αλληλεπιδράσεις τους χωρίς όμως να απορρίπτουμε τις διεργασίες ή τα εργαλεία. Πως θα μπορούσαμε άλλωστε να αναπτύξουμε λογισμικό χωρίς εργαλεία.

Κριτήριο αξιολόγησης 6

Ποια είναι τα δύο βασικά χαρακτηριστικά ενός έργου;

Απάντηση/Λύση

Σύμφωνα με τους παραδοσιακούς ορισμούς που έχουν δοθεί, «έργο» είναι ένα προσωρινό εγχείρημα που στοχεύει στη δημιουργία ενός μοναδικού προϊόντος ή υπηρεσίας. Στον ορισμό αυτό:

- Προσωρινό σημαίνει ότι κάθε έργο έχει καθορισμένη έναρξη και λήξη.
- Μοναδικό σημαίνει ότι το προϊόν ή η υπηρεσία διαφέρει κατά διακριτό τρόπο από όλα τα παρόμοια προϊόντα ή υπηρεσίες.

Κριτήριο αξιολόγησης 7

Ποιες είναι οι τρεις πλευρές του τριγώνου των περιορισμών (σιδερένιο τρίγωνο) ενός έργου;

Απάντηση/Λύση

Το τρίγωνο των περιορισμών στη διαχείριση έργων ή «σιδερένιο τρίγωνο» (iron triangle), είναι ένα εννοιολογικό σχήμα που χρησιμοποιείται για να εξηγήσει τους περιορισμούς στη διαχείριση έργων, παρουσιάζοντάς την ως μία διαρκή διαπραγμάτευση στο πλαίσιο ενός τρισδιάστατου συστήματος με πλευρές το κόστος, το δεδομένο εύρος (scope) και τον χρόνο που απαιτεί ένα έργο. Οποιαδήποτε παρέμβαση στο ένα σκέλος φέρνει αναπόφευκτα αλλαγές στα άλλα δύο. Δεν μπορεί κανείς π.χ. να αλλάξει το εύρος ενός έργου χωρίς να επιφέρει αλλαγές στο κόστος και στο χρονικό πλάνο του. Η ευέλικτη προσέγγιση επιχειρεί να σπάσει αυτό το «σιδερένιο τρίγωνο» αποδεχόμενη ότι η αλλαγή απαιτήσεων, χρονικών περιορισμών κ.λπ. είναι αναπόσπαστα δεμένη με την ανάπτυξη προϊόντων λογισμικού και, συνεπώς, θα πρέπει να βρούμε ευέλικτους τρόπους να τη διαχειριστούμε.

Η φιλοσοφία της ευέλικτης διοίκησης

Learn from yesterday,

live for today, hope for tomorrow.

The important thing is not to stop questioning.

Albert Einstein

Σύνοψη

Η ευέλικτη προσέγγιση διαφέρει από οποιαδήποτε άλλη προσέγγιση στην ανάπτυξη λογισμικού, γιατί ξεκίνησε με βάση ένα σύνολο ιδεών, αξιών και αρχών που αν συνδυαστούν αποτελούν μια κουλτούρα και μια νοοτροπία.

Η προσέγγιση αυτή έφερε επανάσταση στον κόσμο της ανάπτυξης λογισμικού. Οι ομάδες που υιοθέτησαν την ευέλικτη προσέγγιση κατάφεραν με συστηματικό τρόπο να δημιουργούν λογισμικό υψηλής ποιότητας. Στο κεφάλαιο αυτό θα παρουσιάσουμε λεπτομερώς αυτή τη φιλοσοφία αναλύοντας τις τέσσερις αξίες και τις δώδεκα αρχές που πρεσβεύει η ευέλικτη φιλοσοφία.

2 Η φιλοσοφία της ευέλικτης προσέγγισης

2.1 Τα άτομα και οι αλληλεπιδράσεις είναι πιο σημαντικά από τις διεργασίες και τα εργαλεία

Στις επιχειρήσεις πληροφορικής και υψηλής τεχνολογίας, υπάρχει μια επικρατούσα κουλτούρα που καθιστά την εφαρμογή αυτή της πρώτης ευέλικτης αξίας πολύ δύσκολη. Σύμφωνα με τον Philip Atkinson, η **εταιρική κουλτούρα** «είναι η υποδομή, η κόλλα που ενώνει τους ανθρώπους και τις διαδικασίες για τη δημιουργία αποτελεσμάτων» (Atkinson, 2012). Η κουλτούρα ενός οργανισμού είναι κάτι που απαιτεί πολλά χρόνια για να δημιουργηθεί, λόγω της συσσώρευσης αλληλεπιδράσεων και εμπειριών που έχουν διαμορφωθεί σε ένα κοινό σύστημα πεποιθήσεων για το πώς διεκπεραιώνεται η εργασία, για το πώς λαμβάνονται οι αποφάσεις, κ.α. Συνεπώς, για να αλλάξει, απαιτείται κριτική εξέταση των παλαιών πρακτικών, η δημιουργία νέων πρακτικών και εμπειριών, καθώς και ανταμοιβή για όσους υιοθετούν τις νέες πρακτικές και αντιλήψεις. Πολλές φορές η υιοθέτηση νέων πρακτικών είναι δύσκολη και άβολη και, για το λόγο αυτό, χρειάζεται δέσμευση του οργανισμού, καθώς και ενεργή συμμετοχή της διοίκησης σε όλα τα επίπεδα του οργανισμού.

Συνεπώς, η βασική δυσκολία την υιοθέτηση αυτής της αξίας, δηλαδή το να δίνουμε έμφαση στους εργαζομένους και όχι στις διεργασίες, έγκειται στην απαιτούμενη αλλαγή κουλτούρας, αφού η λειτουργία των περισσοτέρων επιχειρήσεων βασίζεται στις αρχές της «επιστημονικής διοίκησης» (scientific management). Η επιστημονική διοίκηση στοχεύει στην καταγραφή με λεπτομερή τρόπο όλων των βημάτων που απαιτούνται για την εκτέλεση μιας εργασίας, τον επιμερισμό της εργασίας, τις έννοιες της εξουσίας και της ατομικής υπευθυνότητας, κ.α. Επιπλέον, οι μηχανικοί λογισμικού και οι προγραμματιστές έχουν μάθει να βασίζουν την εργασία τους στη συστηματική χρήση εργαλείων λογισμικού (Berteig, 2015).

Κάθε διοίκηση επιθυμεί να ορίσει διεργασίες που εμπεριέχουν σαφώς περιγραφόμενες δραστηριότητες/βήματα, σαφείς εισόδους και εξόδους, και σαφείς πηγές και παραλήπτες της δραστηριότητας, ώστε να εξασφαλίσει την ποιότητα μέσω της προκαθορισμένης επανάληψης. Αντίστοιχα, οι μηχανικοί λογισμικού δημιουργούν κατάλληλα εργαλεία για την αυτοματοποίηση αυτών των διεργασιών

με σκοπό τη βελτίωση της αποτελεσματικότητας, της ποιότητας και της αξιοπιστίας του παραγόμενου λογισμικού.

Ταυτόχρονα, η κάθε διοίκηση δημιουργεί οργανωτικές δομές, ρόλους με λεπτομερείς περιγραφές, στόχους και υπευθυνότητες με σκοπό την καλύτερη αξιοποίηση των επιχειρηματικών πόρων. Αντίστοιχα, δημιουργούνται και τα εργαλεία που έχουν σκοπό να περιορίσουν και να ελέγξουν τους εργαζόμενους σε αυτούς τους λεπτομερείς ρόλους για τη βελτίωση της αποτελεσματικότητας, της ποιότητας και της αξιοπιστίας.

Συνεχίζοντας αυτό τον ατέρμονα κύκλο η διοίκηση της επιχείρησης αυτοματοποιεί όλο και περισσότερο την παραγωγική διεργασία με σκοπό την αύξηση της παραγωγικότητας και του ελέγχου. Συνεπώς η κυρίαρχη κουλτούρα είναι η επίλυση προβλημάτων με την εισαγωγή διεργασιών και εργαλείων και όχι δίνοντας έμφαση στους εργαζομένους, μια κουλτούρα που είναι (σχεδόν) εγγενώς αντι-ευέλικτη.

Ας δούμε το πρώτο σκέλος αυτής της αξίας, δηλαδή «τα άτομα και οι αλληλεπιδράσεις» σε περισσότερο βάθος. Είναι πολύ συνηθισμένο στην εργασία μας αλλά και στην καθημερινή μας ζωή να συνεργαζόμαστε με άλλα άτομα. Ο καθένας από εμάς σε αυτή τη συνεργασία καταθέτει μοναδικές δεξιότητες, στοιχεία της προσωπικότητάς μας, τα ενδιαφέροντά μας, ενώ ταυτόχρονα και οι συνεργάτες μας κάνουν το ίδιο. Σε ένα περιβάλλον εργασίας βασισμένο σε ένα γραφειοκρατικό σύστημα εργασίας με καλά ορισμένες επιχειρηματικές διεργασίες και ιεραρχική οργάνωση της εργασίας (Meisenbach & Jensen, 2017), είναι εύκολο να ξεχάσουμε την μοναδικότητα του κάθε ατόμου και να μετατρέψουμε τους εργαζομένους σε ανθρώπινους πόρους (human resources), ή με άλλα λόγια να τους αντικειμενοποιήσουμε (objectification). Με αυτή την οπτική γωνία, οι άνθρωποι είναι και αυτοί μια μορφή επιχειρηματικών πόρων, όπως και το κεφάλαιο, ο εξοπλισμός, το απόθεμα που ανήκουν στην επιχείρηση. Η αντικειμενοποίηση των εργαζομένων (Nussbaum, 1995) έχει τα ακόλουθα, συνήθως αρνητικά, χαρακτηριστικά για το άτομο:

- Αντιμετώπιση του ατόμου ως εργαλείου κατάλληλου για την επίτευξη των σκοπών του οργανισμού
- Έλλειψη αυτονομίας ή αυτοδιάθεσης ή άρνηση αυτής
- Μεταχείριση του ατόμου ως άβουλου όντος
- Μεταχείριση του ατόμου ως αναλώσιμου πόρου
- Θεώρηση του ατόμου και της εργασίας αυτού ως εμπορεύσιμου προϊόντος
- Αδιαφορία για τις επιδιώξεις και τα συναισθήματα του ατόμου

Προφανώς, η ύπαρξη των παραπάνω χαρακτηριστικών, που δυστυχώς δεν είναι σπάνια ακόμη και στις σύγχρονες επιχειρήσεις, δημιουργεί αρνητικά συναισθήματα στους εργαζόμενους μιας επιχείρησης, διότι οι περισσότεροι από εμάς, θέλουμε να μας αντιμετωπίζουν ως ελεύθερα όντα, με μοναδικό χαρακτήρα, ικανότητες, αλλά και συναισθήματα. Το ευέλικτο μανιφέστο αναγνωρίζει αυτή την ιδέα, των μοναδικών ατόμων και ζητά από τις επιχειρήσεις να ενσωματώσουν αυτή την αρχή στην κουλτούρα τους.

Η έννοια της «ανθρώπινης εργασίας» (humanizing work) (Renesch, 2006) έχει πολλές πτυχές, όπως για παράδειγμα να ενθαρρύνουμε τους εργαζόμενους για:

- δημιουργικότητα και καινοτομία,
- συνεχή μάθηση και επίλυση προβλημάτων,
- να νοιάζονται για τα άλλα μέλη της ομάδας,
- να νιώθουν υπερήφανοι για τη δουλειά τους,
- να είναι υπεύθυνοι, και
- να προάγουν το ομαδικό πνεύμα.

Από την άλλη πλευρά αυτής της αξίας έχουμε τις διεργασίες και τα εργαλεία (processes and tools), πλευρά που είναι εξίσου ενδιαφέρουσα.

Για την πλευρά αυτή, μια πρώτη βασική παρατήρηση είναι ότι οι διεργασίες και τα εργαλεία δεν κατασκευάζονται, αλλά ούτε και αλλάζουν από μόνα τους. Συνήθως μια διεργασία ορίζεται από εργαζομένους, εφαρμόζεται, παραμένει ίδια, αναβαθμίζεται ή υποβαθμίζεται μέσα στον χρόνο και κάποια στιγμή καταργείται.

Μια δεύτερη παρατήρηση είναι ότι οι διεργασίες και τα εργαλεία είναι φορείς για τη διατήρηση της υπάρχουσας κατάστασης (status quo). Υπό αυτή την οπτική γωνία, μια επιχείρηση είναι ένα σύμπλεγμα κανόνων και αξιών που καθιερώνονται και φέρουν ομαδική συναίνεση. Ή, ακόμη γενικότερα για να γίνουμε μέλη οποιασδήποτε κοινωνικής ομάδας, πρέπει να εσωτερικεύσουμε και να ενεργήσουμε με τους κανόνες που διέπουν τις συμπεριφορές της ομάδας ή να αντιμετωπίσουμε τις αρνητικές συνέπειες που διέπουν την παραβίαση αυτών των κανόνων. Η ύπαρξη όλων των οργανισμών βασίζεται σε αυτή τη συμπεριφορική νόρμα (Lach et al., 2004).

Μια τρίτη παρατήρηση είναι ότι οι μηχανικοί (π.χ. μηχανικοί λογισμικού) ζουν σε ένα φιλοσοφικό δίπολο: ορίζουν διεργασίες και αναπτύσσουν εργαλεία για χρήση από άλλους που πολλές φορές δεν θα ήθελαν να τα χρησιμοποιήσουμε ούτε και οι ίδιοι.

Το ευέλικτο μανιφέστο δίνει έμφαση στους ανθρώπους αλλά δεν αρνείται την ύπαρξη των διεργασιών και των εργαλείων τα οποία τις περισσότερες φορές είναι αναγκαία και χρήσιμα. Για παράδειγμα διεργασίες και εργαλεία που:

- Διευκολύνουν τη συνεργασία της ομάδας
- Επιλύουν σύνθετα τεχνικά προβλήματα
- κ.α.

	Έμφαση στα άτομα και τις αλληλεπιδράσεις	**Έμφαση στις διεργασίες και στα εργαλεία**
Θετικά	Η επικοινωνία είναι σαφής, γρήγορη και αποτελεσματική. Η ομαδική εργασία γίνεται αποδοτική Η ομάδα μπορεί να αυτο-οργανωθεί Η ομάδα μπορεί να καινοτομήσει Η ομάδα μπορεί να προσαρμόσει τις διεργασίες ανάλογα με τις ανάγκες Τα μέλη της ομάδας νιώθουν ικανοποίηση από την εργασία τους	Οι διαδικασίες είναι σαφείς και μπορούν εύκολα να ακολουθηθούν Υπάρχουν γραπτά τεκμήρια της επικοινωνίας.
Αρνητικά	Για να εφαρμοστεί, τα μέλη της ομάδας πρέπει να είναι υπεύθυνα και να καινοτομούν, ικανότητες που δεν είναι πάντα διαθέσιμες σε όλα τα μέλη της ομάδας έργου. Για να εφαρμοστεί, τα μέλη της ομάδας πρέπει να έχουν ομαδικό πνεύμα	Οι άνθρωποι που βασίζονται σε διεργασίες αμελούν να βρουν καλύτερους τρόπους δημιουργίας προϊόντων. Η ίδια διεργασία δεν ταιριάζει το ίδιο σε όλες τις ομάδες Η ίδια διεργασία δεν ταιριάζει το ίδιο σε όλα τα έργα. Η επικοινωνία μπορεί να είναι διφορούμενη και χρονοβόρα.

Πίνακας 2.1: *Άνθρωποι και αλληλεπιδράσεις σε σχέση με τις διεργασίες και τα εργαλεία*

2.2 Έμφαση στο λογισμικό που λειτουργεί παρά στην εκτενή τεκμηρίωση

Σε ένα παραδοσιακό κύκλο ζωής ανάπτυξης λογισμικού ένα τυπικό σενάριο είναι η καταγραφή όλων των απαιτήσεων των πελατών κατά την έναρξη του έργου και στη συνέχεια η ανάπτυξη του λογισμικού χωρίς τη συμμετοχή πελατών. Σε ένα τέτοιο έργο που ακολουθεί για παράδειγμα το μοντέλο του κύκλου ζωής καταρράκτη, η ομάδα ανάπτυξης θα πρέπει να δημιουργήσει μια εκτενή σειρά εγγράφων η οποία θα πρέπει να περιλαμβάνει έγγραφα με σκοπό την καταγραφή των:

- Απαιτήσεων του λογισμικού
- Το σχεδιασμό και την αρχιτεκτονική του συστήματος
- Το σχεδιασμό του ελέγχου που περιλαμβάνουν μοναδιαίους ελέγχους (unit tests), περιπτώσεις λειτουργικού ελέγχου (test cases), ελέγχους ολοκλήρωσης (integration tests), κ.λπ.

Στην παραπάνω τεκμηρίωση θα πρέπει να προστεθεί η τεκμηρίωσή που αναφέρεται στη χρήση του λογισμικού όπως εγχειρίδια χρήσης, εγκατάστασης, λειτουργίας, κ.λπ.

Συνήθως η έννοια του εγγράφου μέσα σε ένα έργο λογισμικού είναι ταυτόσημη πολλές φορές με την έννοια του παραδοτέου (deliverable). Στο σημείο αυτό καλό θα ήταν να κάνουμε κάποιες παρατηρήσεις και να δώσουμε κάποιους ορισμούς:

Ποιοι είναι οι βασικοί λόγοι που τεκμηριώνουμε ένα σύστημα λογισμικού;

Ο βασικός λόγος που δημιουργούμε παραδοτέα και γενικότερα είναι για να επικοινωνήσουμε με τους συμμετέχοντες (stakeholders) του έργου. Πιο συγκεκριμένα μπορούμε να διακρίνουμε τους παρακάτω δυο βασικούς λόγους:

- Ο πρώτος λόγος είναι η παροχή πληροφοριών που είναι απαραίτητες για την αλληλεπίδραση των συμμετεχόντων του έργου. Αυτές οι πληροφορίες μπορεί να είναι ιστορικές, τεχνικές ή προτάσεις θεμάτων συζήτησης, και έχουν ως στόχο να υποβοηθήσουν τους συμμετέχοντες στο να επιτύχουν μια πιο παραγωγική συζήτηση από ό, τι θα συνέβαινε αν δεν είχαν την πληροφορία διαθέσιμη. Όταν τα έγγραφα χρησιμοποιούνται για αυτόν τον σκοπό, τυχόν λάθη ή παραλήψεις είναι εύκολο να διορθωθούν κατά τη διάρκεια της αλληλεπίδρασης.
- Ο δεύτερος σκοπός είναι η καταγραφή των αποτελεσμάτων των συζητήσεων των συμμετεχόντων του έργου. Όταν χρησιμοποιούνται για αυτόν τον σκοπό, τα έγγραφα λειτουργούν ως συλλογικής μνήμης, ώστε τα αποτελέσματα της αλληλεπίδρασης των συμμετεχόντων να είναι λιγότερο πιθανό να αμφισβητηθούν στο μέλλον, όταν οι μνήμες των γεγονότων έχουν εξασθενίσει. Για παράδειγμα, μια σχεδιαστική απόφαση αποτέλεσμα της διαβούλευσης της ομάδας του έργου συνήθως λησμονείται με το πέρας του έργου.

Τι είναι ένα παραδοτέο;

Ορίζουμε ως παραδοτέο οποιοδήποτε μοναδικό και επαληθεύσιμο προϊόν ή αποτέλεσμα που πρέπει να παραχθεί προκειμένου να ολοκληρωθεί μία δραστηριότητα, μία φάση ή ένα έργο λογισμικού. Ο ορισμός των παραδοτέων συνδέεται στενά με τη διαχείριση του αντικειμένου των εργασιών σε ένα έργο, μια και η λίστα των παραδοτέων είναι αυτή που προσδιορίζει με ακρίβεια το αποτέλεσμα του έργου.

Ποια είναι τα είδη των παραδοτέων;

Τα παραδοτέα μπορεί να είναι τελικά ή ενδιάμεσα. Τελικά είναι τα παραδοτέα που αποτελούν τμήμα του τελικού προϊόντος του έργου, ενώ ενδιάμεσα χαρακτηρίζονται τα παραδοτέα που δεν αποτελούν μέρος του τελικού προϊόντος, αλλά είναι απαραίτητα για την παραγωγή του τελικού προϊόντος. Για παράδειγμα,

τελικό παραδοτέο είναι ο κώδικας του συστήματος ή η τεκμηρίωση του συστήματος, ενώ ενδιάμεσο παραδοτέο είναι το έγγραφο που περιέχει την ανάλυση του συστήματος.

Πώς οργανώνουμε τα παραδοτέα;

Για την οργάνωση των παραδοτέων δημιουργούμε μια βάση δεδομένων στην οποία αποθηκεύονται όλες οι παραπάνω πληροφορίες, ενώ ταυτόχρονα όλα τα έγγραφα υπόκεινται σε διαχείριση σχηματισμών (configuration management).

Είναι προφανές ότι σε ένα τυπικό έργο λογισμικού υπάρχουν πάρα πολλά που μπορούν να τεκμηριωθούν, και είναι συχνά δύσκολο κατά τη διάρκεια του έργου να προβλέψουμε τι θα είναι χρήσιμο στο μέλλον και τι όχι. Για το λόγο αυτό πολλές ομάδες - και ειδικά οι διευθυντές τους - αποφασίζουν να ακολουθήσουν μια «ολοκληρωμένη προσέγγιση», όπου τεκμηριώνονται όλα τα αποτελέσματα και η εργασία του έργου, ανεξάρτητα από τη μελλοντική τους χρησιμότητα.

Στην ευέλικτη προσέγγιση σημαντικό είναι το λογισμικό να λειτουργεί (working software) και να προσθέτει αξία στον οργανισμό που το χρησιμοποιεί.

Αυτό σημαίνει ότι το έργο λογισμικού θα πρέπει να κοστίζει λιγότερο σε οικονομικούς όρους, από το όφελος που φέρνει η χρήση του λογισμικού από τον τελικό χρήστη, ώστε η διαφορά αυτή να είναι θετική. Αυτό είναι προφανές στην περίπτωση που κατασκευάζουμε ένα προϊόν λογισμικού. Τα έσοδα από την πώληση του λογισμικού θα πρέπει να είναι μεγαλύτερα από το κόστος ανάπτυξης του προϊόντος.

Είναι βέβαιο λοιπόν ότι το κόστος σύνταξης και συντήρησης παραδοτέων/εγγράφων είναι σημαντικό, και κάθε φορά πριν προχωρήσουμε στην ανάπτυξη ενός εγγράφου μπορούμε να εφαρμόσουμε κάποια απλά κριτήρια για να αποφασίσουμε, όπως (Koch, 2005) :

- Ποιος είναι ο σκοπός του εγγράφου; Εάν δεν υπάρχει σαφής σκοπός ή χρήση για το έγγραφο, τότε η παραγωγή του είναι πιθανό να μην είναι αναγκαία.
- Σε ποιο κοινό απευθύνεται το έγγραφο; Εάν το κοινό αποτελείται από πολλούς διαφορετικούς συμμετέχοντες, τότε θα πρέπει να αναρωτηθούμε εάν το έγγραφο μπορεί να ανταποκριθεί στις ανάγκες όλων αυτών. Συνεπώς, αν το κοινό που απευθύνεται είναι ανομοιογενές και έχει διαφορετικές ανάγκες για να καλύψει τις ανάγκες, τότε η παραγωγή του είναι πιθανό να αποτελεί σπατάλη.
- Το έγγραφο καταγράφει τα αποτελέσματα της διαπροσωπικής επικοινωνίας (ή προετοιμάζει τη διαπροσωπική επικοινωνία); Εάν όχι, τότε είναι πιθανό να μην είναι αναγκαίο.
- Είναι το έγγραφο, στη δεδομένη χρονική στιγμή που βρίσκεται το έργο, ακόμη χρήσιμο; Η διατήρηση ενός εγγράφου πέρα από το ωφέλιμο χρόνο του αποτελεί σπατάλη.
- Ποιο είναι το ελάχιστο ποσό προσπάθειας που απαιτείται για να διασφαλιστεί ότι το παραδοτέο μπορεί να εκπληρώσει το σκοπό του για το κοινό στο οποίο απευθύνεται κατά τη διάρκεια του χρόνου στον οποίο είναι απαραίτητο;

Επίσης, το γεγονός ότι δίνουμε έμφαση στο «λειτουργούν λογισμικό» δεν σημαίνει ότι θα πρέπει να εγκαταλείψουμε ολοκληρωτικά την τεκμηρίωση αυτού. Άλλωστε η ευέλικτη αξία στην οποία αναφερόμαστε μιλά για «εκτενή» τεκμηρίωση. Έτσι, υπάρχουν πολλά είδη εγγράφων που είναι πολύ χρήσιμα τόσο για την ομάδα, όσο και για τον τελικό χρήστη/πελάτη. Στο σημείο αυτό θα κάνουμε τρεις παρατηρήσεις:

- Πολλές φορές αυτοί που τεκμηριώνουν ένα σύστημα λογισμικού είναι οι ίδιοι που το υλοποιούν. Στην περίπτωση αυτή η χρησιμότητα της τεκμηρίωσης είναι ελάχιστη.
- Σε άλλες περιπτώσεις η τεκμηρίωση και τα έγγραφα που παράγουμε μας βοηθούν να κατανοήσουμε το πρόβλημα, να επικοινωνήσουμε με τους πελάτες/χρήστες κάνοντάς τα ιδιαίτερα χρήσιμα και γι’ αυτό θα πρέπει να αναπτύσσονται.

- Οι μηχανικοί λογισμικού και ιδιαίτερα οι πιο έμπειροι γνωρίζουν το είδος της τεκμηρίωσης που είναι αναγκαίο και χρήσιμο και επενδύουν χρόνο στην παραγωγή τέτοιων εγγράφων (π.χ. δυναμική συμπεριφορά του συστήματος με δημιουργία διαγραμμάτων ακολουθίας)

Η σύγχρονη προσέγγιση συνδυάζει την προσπάθεια ανάπτυξης «λειτουργούντος λογισμικού», με την τεκμηρίωση, αφού στόχος είναι η τεκμηρίωση να είναι ενσωματωμένη στο λογισμικό (self-documented). Για παράδειγμα, όταν ακολουθούμε την πρακτική της ανάπτυξης με βάση τους ελέγχους (test driven development), αναπτύσσουμε τους μοναδιαίους ελέγχους (unit tests) πριν την ανάπτυξη του λογισμικού και οι μοναδιαίοι αυτοί έλεγχοι ενσωματώνονται στον κώδικα του συστήματος λογισμικού αποτελώντας τεκμήρια ελέγχου, δίνοντας στους προγραμματιστές καταγεγραμμένες οδηγίες για τι πρέπει να κάνει ο κώδικας και ποια είναι η αναμενόμενη συμπεριφορά των συστατικών του λογισμικού. Έτσι όταν η ομάδα ανάπτυξης εργάζεται τεκμηριώνοντας τους μοναδιαίους ελέγχους του «λειτουργούντος λογισμικού», συμβάλλει θετικά στην πρόοδο του έργου (Stellman & Greene, 2014).

2.3 Η συνεργασία με τον πελάτη αντί για τη διαπραγμάτευση της σύμβασης

Οι σύγχρονες επιχειρήσεις σήμερα, έχοντας στη διάθεσή τους τις νέες τεχνολογίες, την αυξανόμενη χρήση του διαδικτύου από την πλευρά των καταναλωτών, τις ευέλικτες τεχνικές κατασκευής προϊόντων καθώς και τις εφοδιαστικές αλυσίδες που λειτουργούν σε παγκόσμιο επίπεδο, καλούνται να λειτουργήσουν σε ένα πολύ διαφορετικό, σε σχέση με το παρελθόν, ανταγωνιστικό περιβάλλον, όπου η βασική οντότητα είναι ο πελάτης.

Ο πελάτης είναι επομένως η βασική έννοια γύρω από την οποία περιστρέφονται όλες οι ενέργειες της σύγχρονης επιχείρησης, ενέργειες είτε παραγωγικές, είτε μάρκετινγκ, αφού είναι άμεση η σχέση μεταξύ της προσέλκυσης και ικανοποίησης των πελατών και του ανταγωνιστικού πλεονεκτήματος της επιχείρησης.

Στο πλαίσιο αυτό, αλλά και με στόχο την ανάπτυξη ποιοτικού λογισμικού μια από τις βασικές προτεραιότητες των ευέλικτων μεθόδων είναι η εδραίωση της σχέσης με τον πελάτη και της συνεργασίας με αυτόν. Μια τέτοια σχέση που βασίζεται στη συνεργασία αντί στην αντιπαράθεση, παράγει καλύτερα, πιο χρήσιμα και, πιο λειτουργικά προϊόντα. Ως αποτέλεσμα αυτής της προτεραιότητας οι ευέλικτες μεθοδολογίες καθιστούν τον πελάτη μέρος του έργου σε συνεχή βάση.

Συνεπώς είναι πιο σημαντικό να επικεντρωθούμε στη συνεισφορά του κάθε ατόμου στην ομάδα, στην επικοινωνία μέσα στην ομάδα του έργου και στην ανάπτυξη μιας σχέσης εμπιστοσύνης, παρά στην τήρηση των διαδικασιών, στην συνεχή διαπραγμάτευση των όρων της σύμβασης ή στη χρήση περίπλοκων εργαλείων.

Όπως είναι συνηθισμένο στις επιχειρήσεις, υπάρχει ένας ευρύς ορισμός για τη λέξη «πελάτης». Ο πελάτης μπορεί να είναι άτομο ή εταιρεία που είναι διαφορετικό από τον οργανισμό που παράγει το λογισμικό, ή μπορεί να είναι ένα άλλο τμήμα της επιχείρησης. Επομένως μπορούμε να διαχωρίσουμε δύο κατηγορίες πελατών:

- Οι εξωτερικοί πελάτες που ανήκουν σε μια νομική οντότητα που διαφέρει από τον οργανισμό που υλοποιεί το έργο. Οι σχέσεις με τους εξωτερικούς πελάτες διέπονται σχεδόν πάντα από μια σύμβαση, που προσδιορίζει τη σχέση μεταξύ των δύο νομικών οντοτήτων και επίσης υπάρχει πάντα μια οικονομική συναλλαγή.
- Ένας εσωτερικός πελάτης ανήκει στο ίδιο νομικό πρόσωπο με τον οργανισμό που υλοποιεί το έργο. Επομένως, ένας τέτοιος πελάτης μπορεί να είναι ένα άλλο τμήμα εντός της επιχείρησης ή του οργανισμού. Στην περίπτωση αυτή δεν είναι απαραίτητη η ύπαρξη μιας σύμβασης, αφού οι όροι της συνεργασίας ορίζονται εσωτερικά στην επιχείρηση. Αυτό όμως πολλές φορές είναι λάθος, διότι ακόμη και στις πιο απλές περιπτώσεις, είναι απαραίτητο να καθορίσουμε επακριβώς τι περιμένει ο κάθε συμμετέχων από τα άλλα μέρη της σχέσης. Άλλωστε αυτή είναι και η βασική λειτουργία μιας σύμβασης, να περιγράψει με ακρίβεια τις υποχρεώσεις και τα δικαιώματα του κάθε συμβαλλόμενου. Συνεπώς, οι λεπτομέρειες των συμβάσεων με τους εσωτερικούς πελάτες δεν διαφέρουν σημαντικά από εκείνες με εξωτερικούς πελάτες.

Ανεξάρτητα από το είδος του πελάτη υπάρχουν τρία βασικά συστατικά στη σχέση με τους πελάτες που επηρεάζονται από την υιοθέτηση ευέλικτων μεθόδων:

- η ύπαρξη σύμβασης και η μορφή αυτής,
- η διαχείριση των απαιτήσεων λογισμικού και ειδικά των αλλαγών που προκύπτουν, και
- η μορφή και η ένταση των αλληλεπιδράσεων που έχουν οι πελάτες με την ομάδα ανάπτυξης.

Τα τρία αυτά συστατικά θα παρουσιαστούν στις επόμενες τρεις παραγράφους.

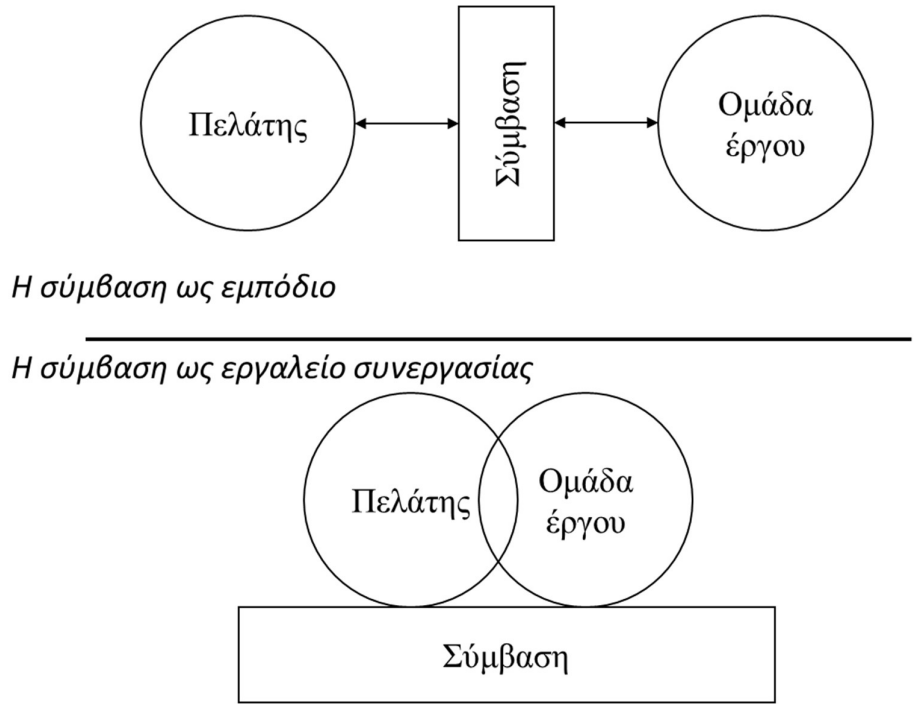
2.3.1 Οι συμβάσεις

Αν και οι ευέλικτες μέθοδοι αναγνωρίζουν την αναγκαιότητα ύπαρξης συμβάσεων και άλλων μορφών συμφωνίας, δεν προσδιορίζουν ρητά το περιεχόμενο αυτών των συμφωνιών. Αν και για πολλούς, αυτό αποτελεί μια σημαντική παράλειψη, η φιλοσοφία των ευέλικτων μεθόδων είναι τέτοια που δεν επιτρέπει τον ακριβή προσδιορισμό του περιεχομένου της σύμβασης, ούτε και την με λεπτομέρεια καταγραφή των ορόσημων ή απαιτήσεων του έργου, κ.λπ.. Αυτό προκύπτει από το γεγονός ότι στις ευέλικτες μεθόδους, οι πελάτες και η ομάδα έργου είναι συνεργάτες και άτομα που μαθαίνουν μαζί και προσπαθούν να κατανοήσουν ποιο είναι το καλύτερο αποτέλεσμα για τον οργανισμό που αντιπροσωπεύουν ή για το πρόβλημα που καλούνται να επιλύσουν.

Στην πραγματικότητα όμως οι πελάτες αλλά και οι προμηθευτές δεν έχουν ως μόνο στόχο ένα επιτυχημένο έργο αλλά την επίτευξη του οφέλους και του ανταγωνιστικού πλεονεκτήματος. Όμως και οι δύο αυτές πλευρές αντιμετωπίζουν το ίδιο πρόβλημα, δηλαδή την αδυναμία να γνωρίζουν τι πραγματικά χρειάζεται και πώς να αλλάξουν οι λεπτομέρειες του πεδίου εφαρμογής κατά τη διάρκεια του έργου για τους δικούς τους λόγους. Το βασικό πρόβλημα είναι επομένως, για όλα τα έργα ανάπτυξης λογισμικού και όλους τους παρόχους υπηρεσιών, η μεταβλητότητα του τι πρόκειται και τι τελικά θα παραδοθεί. Οι «παραδοσιακοί» διαχειριστές έργων, οι διαχειριστές συμβάσεων αλλά και οι νομικοί σύμβουλοι των οργανισμών γενικά επιθυμούν συμφωνίες/συμβάσεις όπου καθορίζονται πλήρως όλες οι παράμετροι ενός έργου, όπως οι ρόλοι των συμμετεχόντων στο έργο ανάπτυξης λογισμικού, καθώς και οι υποχρεώσεις αυτών, ο προϋπολογισμός, το χρονοδιάγραμμα που πρέπει να ισχύει, όπως επίσης επιθυμούν να προσδιορίζεται η λειτουργικότητα του συστήματος προς παράδοση, η επιθυμητή ποιότητα καθώς και οι έλεγχοι αποδοχής που πρέπει να γίνουν. Επομένως, νιώθουν άβολα με συμφωνίες χωρίς σαφή ορόσημα και, συμφωνίες αορίστου χρόνου, επειδή βλέπουν μέσα σε αυτές κινδύνους καθυστερημένης παράδοσης του τελικού προϊόντος.

Σε αντιδιαστολή, οι ευέλικτες μέθοδοι αναγνωρίζουν τους κινδύνους που περιλαμβάνονται σε πιο ανοικτές συμβάσεις, αλλά αντί να προσπαθούν να τους αποφύγουν μέσω προσεκτικά καταγεγραμμένων προδιαγραφών και απαιτήσεων, προσπαθούν να μετριάσουν τους κινδύνους μέσω στενής συνεργασίας με τον πελάτη και εδραιώνοντας την εμπιστοσύνη. Αποδέχονται την πραγματικότητα η οποία είναι: α) δεν μπορούν να γνωρίζουν εξ αρχής όλες τις παραμέτρους του έργου, και β) θέτουν βραχυπρόθεσμους στόχους οι οποίοι στη συνέχεια μπορούν να αλλάζουν με την πρόοδο του έργου. Έτσι για παράδειγμα, με τη χρήση ευέλικτων μεθόδων, ο προϋπολογισμός και το χρονοδιάγραμμα διατηρούνται σταθερά, ενώ η λειτουργικότητα του συστήματος μπορεί να ποικίλει. Συνεπώς, μια σύμβαση σε ένα ευέλικτο έργο είναι διαφορετική από μια τυπική, επειδή λειτουργεί ως πλατφόρμα συνεργασίας και όχι ως κείμενο διαπραγμάτευσης σε περίπτωση προβλημάτων ή αλλαγών. Θα πρέπει επομένως να προσδιορίζει στόχους και όχι απαιτήσεις, να καθορίζει σαφώς τον τρόπο διαχείρισης των απαιτήσεων, καθώς και τη φύση και την έκταση της απαιτούμενης εμπλοκής των πελατών.

Στην Εικόνα 2.1 παρουσιάζεται η «παραδοσιακή» φιλοσοφία στη διαχείριση έργων σε αντιδιαστολή με την ευέλικτη φιλοσοφία.

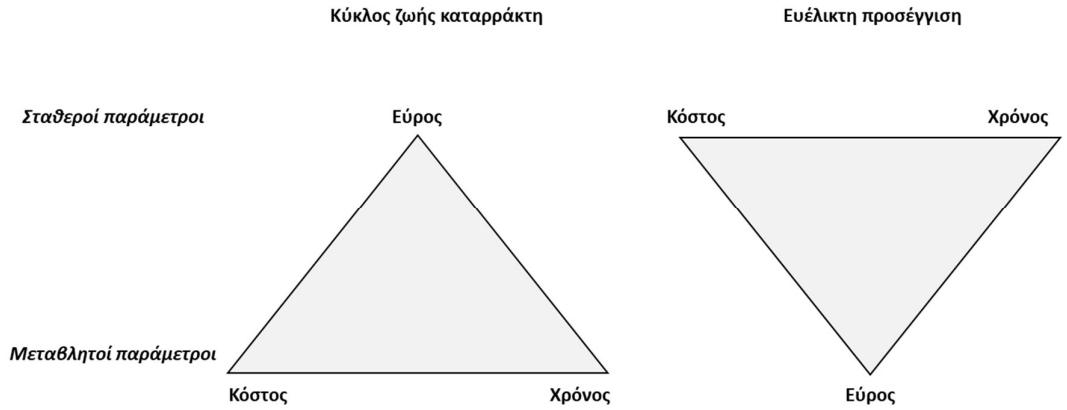


Εικόνα 2.1: *Η σύμβαση ως εμπόδιο και ως εργαλείο συνεργασίας*

Ορισμένες τεχνικές συμβασιοποίησης που μπορούν να υποστηρίξουν την ευέλικτη φιλοσοφία είναι οι ακόλουθες (Pfarl, et al., 2013):

- **Συμβάσεις με πολυεπίπεδη δομή (Multi-tiered structure).** Αντί η συμβατική σχέση να περιγράφεται σε ένα μόνο έγγραφο, μπορούμε να επιτύχουμε μεγαλύτερη ευελιξία περιγράφοντας διαφορετικές πτυχές του έργου σε διαφορετικά έγγραφα. Σταθερές προβλέψεις όπως για παράδειγμα οι εγγυήσεις, η διαιτησία ή γενικοί όροι και αρχές της συνεργασίας μπορούν να περιγραφούν και να οριστικοποιηθούν σε μια κύρια συμφωνία. Θέματα που μεταβάλλονται (π.χ. τιμές παρεχόμενων υπηρεσιών, προδιαγραφές/περιγραφές προϊόντων μπορούν να περιγράφονται σε άλλα έγγραφα που είναι εφικτό να αλλάζουν πιο συχνά. Τέλος, δυναμικά στοιχεία που είναι αναμενόμενα να αλλάζουν τακτικά όπως το εύρος εργασιών, το χρονοδιάγραμμα και ο προϋπολογισμός του έργου, να περιλαμβάνονται σε ένα άλλο έγγραφο που να μπορεί να αλλάζει κι αυτό τακτικά, π.χ. κατά τη διάρκεια μιας συνάντησης εργασίας. Επομένως η απομόνωση των μεταβλητών στοιχείων μιας σύμβασης σε ένα ενιαίο έγγραφο απλοποιεί τις αλλαγές και συνεπώς αυξάνει την ευελιξία.
- **Να δίδεται έμφαση στην αξία (του λογισμικού) που παραδόθηκε.** Πολλές φορές οι σχέσεις με τους προμηθευτές θα πρέπει να ορίζονται με ορόσημα ή «πύλες φάσης» (phase-gates), δηλαδή να επικεντρωνόμαστε στο ενδιάμεσο παραδοτέο (το λογισμικό να είναι χρήσιμο και λειτουργικό σε κάθε φάση και σε κάθε παράδοση) και όχι σε ένα πλήρες παραδοτέο (που ποτέ δεν έρχεται). Στην περίπτωση αυτή τα ορόσημα και οι όροι πληρωμής συνδέονται με γνώμονα πάντα την αξία του παραδοτέου.
- **Προσαυξήσεις προκαθορισμένης τιμής (fixed price increments).** Αντί να συμφωνήσουμε εξ' αρχής για όλο το έργο και όλες τις απαιτήσεις, αποδομούμε τις απαιτήσεις του έργου σε μικρότερα τμήματα σταθερής τιμής, που περιλαμβάνουν έναν συγκεκριμένο αριθμό ιστοριών χρηστών (user stories). Έτσι ο πελάτης, έχει περισσότερο έλεγχο στο πώς ξοδεύονται τα χρήματα, ενώ για τον προμηθευτή, περιορίζεται ο οικονομικός κίνδυνος που μπορεί να προκύψει από την υποεκτίμηση του εύρους ή της πολυπλοκότητας του έργου.

- **Το έργο να μην υπερβαίνει το διαθέσιμο χρόνο και προϋπολογισμό.** Στην περίπτωση αυτή ο προϋπολογισμός και ο χρόνος είναι σταθερά αλλά μπορεί να αλλάξει το εύρος. Αυτό σημαίνει ότι όταν μια νέα απαίτηση κρίνεται αναγκαία, σημαντική και με ιδιαίτερη αξία για την επιχείρηση θα πρέπει για να ενσωματωθεί να αφαιρεθεί μια άλλη εργασία/απαίτηση. Για να εφαρμοστεί αυτή η προσέγγιση θα πρέπει να παρακολουθούμε το έργο λεπτομερώς.
- Τέλος, ο Pfarl et al. (2013) στο σχετικό βιβλίο τους Agile Contracts όρισαν την έννοια της ευέλικτης σύμβασης σταθερής τιμής (agile fixed-price). Το κύριο χαρακτηριστικό αυτών των συμβάσεων είναι ότι το εύρος του έργου πληροφορικής δεν είναι καθορισμένο λεπτομερώς εξαρχής. Αντίθετα, το κόστος και ο χρόνος ορίζονται με βάση αρχές που συμφωνούνται στην έναρξη του έργου ενώ το εύρος των υπηρεσιών αναπτύσσεται και υλοποιείται βήμα προς βήμα σε σύντομους κύκλους επανάληψης. Αυτό σημαίνει ότι παραλείπεται η υπόθεση λεπτομερούς πρόβλεψης του εύρους του έργου. Η λογική αυτή απεικονίζεται στην Εικόνα 2.2.



Εικόνα 2.2: *Η λογική της ευέλικτης σύμβασης σταθερής τιμής*

Οι παραπάνω μέθοδοι είναι μόνο μερικές από τις προσεγγίσεις που μπορούν να εφαρμοστούν. Δεν θα πρέπει να ξεχνά κανείς ότι προϋπόθεση για την εφαρμογή της ευελιξίας είναι η εμπιστοσύνη και η συνεργασία (Agile Alliance, 2017).

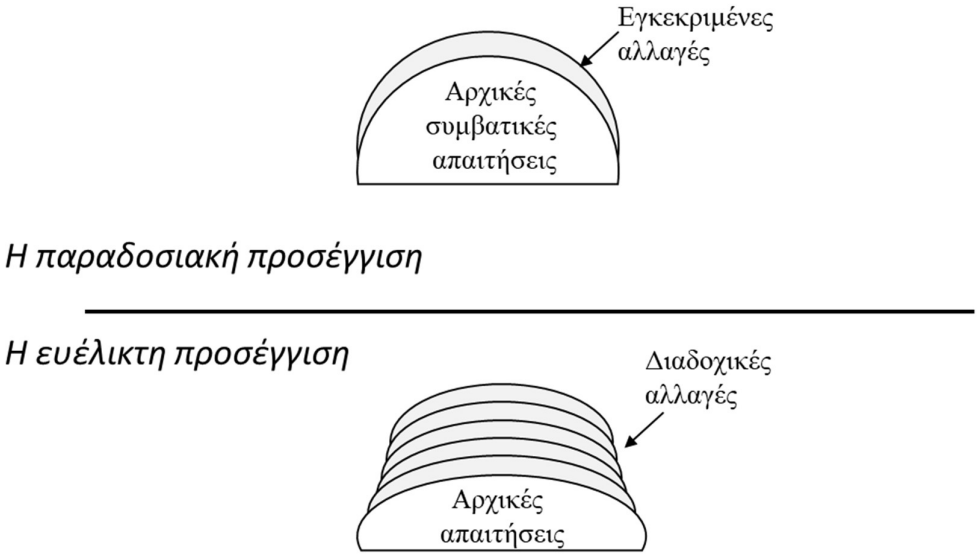
2.3.2 Η διαχείριση των απαιτήσεων

Η παραδοσιακή διαχείριση έργων καθορίζει ότι οι απαιτήσεις για το υπό ανάπτυξη σύστημα πρέπει να καταγράφονται και να οριστικοποιούνται σε σχετικά πρώιμο στάδιο του έργου. Δηλαδή, ο πελάτης και η ομάδα ανάπτυξης θα πρέπει να συμφωνήσουν ότι οι απαιτήσεις όπως τεκμηριώνονται εκείνη την χρονική στιγμή είναι μια ακριβής περιγραφή του τι θα παραδοθεί.

Αυτή η συμφωνία αποτελεί τη βάση για το έργο της ανάπτυξης, καθώς και για τις δραστηριότητες αποδοχής του έργου από τον πελάτη. Μόλις επιτευχθεί αυτή η συμφωνία, οι βασικές απαιτήσεις αποτελούν μέρος μιας σύμβασης, και οποιαδήποτε αλλαγή απαιτεί διαπραγμάτευση και συμφωνία προτού συμπεριληφθεί στα παραδοτέα του έργου.

Οι ευέλικτες μέθοδοι αντιμετωπίζουν τις απαιτήσεις του έργου, ως μια πτυχή του έργου, η οποία μπορεί να αλλάξει, αφού δεν είναι γνωστή πλήρως κατά την έναρξη του έργου. Συνεπώς οι απαιτήσεις θα εξελιχθούν και θα οριστικοποιηθούν κατά τη διάρκεια του έργου. Σε ένα ευέλικτο έργο λοιπόν είναι αρχικά αναγκαίο να γνωρίζουμε τις απαιτήσεις σε ένα σχετικά υψηλό επίπεδο, αφήνοντας πολλά περιθώρια ερμηνείας και προσαρμογής κατά τη διάρκεια του έργου. Θεωρείται βέβαιο ότι τόσο ο πελάτης όσο και η ομάδα έργου θα προτείνουν αλλαγές καθ' όλη τη διάρκεια του έργου. Ωστόσο, ο πελάτης είναι ο μόνος αρμόδιος να εγκρίνει, να απορρίψει και να δώσει προτεραιότητα στις συνεχώς μεταβαλλόμενες απαιτήσεις (έχοντας την τεχνική βοήθεια της ομάδας έργου). Επομένως, οι αρχικές απαιτήσεις αποτελούν μια βάση

πάνω στην οποία οι απαιτήσεις μεταβάλλονται και συσσωρεύονται καθ' όλη τη διάρκεια ζωής του έργου. Ο βασικός μας στόχος είναι να διασφαλίσουμε ότι η λειτουργικότητα που παραδίδεται τελικά, ικανοποιεί τις ανάγκες του πελάτη. Στην Εικόνα 2.3 παρουσιάζεται αυτή η λογική τόσο για τις παραδοσιακές όσο και για τις ευέλικτες μεθόδους.



Εικόνα 2.3: *Η διαχείριση των αλλαγών στην παραδοσιακή και στην ευέλικτη προσέγγιση*

2.3.3 Οι αλληλεπιδράσεις και η επικοινωνία

Επίσης, οι άνθρωποι είναι πιο αποτελεσματικοί όταν επικοινωνούν απευθείας, βρίσκονται στο ίδιο χώρο, αφού μειώνονται τα εμπόδια επικοινωνίας μεταξύ των ατόμων της ομάδας. Τα εμπόδια επικοινωνίας σε ομάδες έργου και ειδικά όταν τα μέλη της ομάδας είναι πολλά έχουν μελετηθεί εκτενώς στη βιβλιογραφία. Για παράδειγμα τα πιο βασικά είναι:

- Έλλειψη εμπιστοσύνης μεταξύ των μελών της ομάδας
- Η έλλειψη χρόνου και τα πιεστικά ορόσημα του έργου
- Η τεχνογνωσία που είναι διαθέσιμη σε κάθε μέλος της ομάδας του έργου σε σχέση με αυτή που είναι απαιτούμενη για το έργο συνολικά
- Διαφορετικές εμπειρίες και κουλτούρα μεταξύ των μελών της ομάδας έργου
- Το στυλ ηγεσίας της διοίκησης του έργου

Επομένως η συμμετοχή των πελατών στην ομάδα του έργου και η καθημερινή συνεργασία βοηθά σημαντικά στην υπερπήδηση των παραπάνω εμποδίων. Για παράδειγμα, η καθημερινή επαφή δημιουργεί σχέσεις εμπιστοσύνης, μειώνει τον χρόνο επίλυσης προβλημάτων, βοηθά στην εξομάλυνση των διαφορών που οφείλονται στις διαφορετικές εταιρικές κουλτούρες, κ.λπ.

Οι παραδοσιακές προσεγγίσεις διαχείρισης έργων προτείνουν την αλληλεπίδραση με τους πελάτες σε τρία βασικά σημεία ενός έργου (Layton, & Ostermiller, 2017).

- **Στην έναρξη έργου:** Όταν ο πελάτης και ο διαχειριστής του έργου (project manager) - ή άλλος εκπρόσωπος της ομάδας έργου - διαπραγματεύονται τις λεπτομέρειες της σύμβασης.
- **Στη διαχείριση των αλλαγών (change management)** κατά τη διάρκεια του έργου: Όταν ο πελάτης και ο διαχειριστής του έργου διαπραγματεύονται τις αλλαγές της σύμβασης.

- **Στη λήξη έργου:** Όταν η ομάδα έργου παραδίδει το ολοκληρωμένο προϊόν στον πελάτη. Στην περίπτωση που το προϊόν δεν ανταποκρίνεται στις προσδοκίες των πελατών, ο διαχειριστής του έργου και ο πελάτης διαπραγματεύονται επιπλέον αλλαγές στη σύμβαση.

Αυτή η ιστορική εστίαση στη διαπραγμάτευση αποθαρρύνει δυνητικά πολύτιμη συμβολή των πελατών και μπορεί ακόμη και να δημιουργήσει μια εχθρική σχέση μεταξύ πελατών και ομάδων έργων.

2.4 Η ανταπόκριση στην αλλαγή

Η υλοποίηση κάθε έργου είναι δυνατόν να επιφέρει οργανωτικές αλλαγές στην επιχείρηση. Αυτό βέβαια δεν ισχύει για όλα τα είδη των έργων. Ο βαθμός του κινδύνου είναι συνάρτηση του μεγέθους της αλλαγής την οποία επιφέρει το έργο. Το βασικό στοιχείο είναι ο τρόπος με τον οποίο η αλλαγή γίνεται αποδεκτή. Αν η αλλαγή γίνει δεκτή με ενθουσιασμό, τότε ο κίνδυνος μη αποδοχής είναι μικρός, ενώ στην αντίθετη περίπτωση που η αλλαγή γίνεται δεκτή με επιφυλάξεις ο κίνδυνος μη αποδοχής είναι μεγάλος.

Για να γίνει θετικά δεκτή η αλλαγή που επιφέρει ένα έργο πρέπει να υπάρξει η κατάλληλη προ- εργασία κατά τη διάρκεια του έργου. Πιο συγκεκριμένα πρέπει:

- η αλλαγή που επιφέρει το έργο να καλύπτει μια γενικώς αποδεκτή ανάγκη,
- οι απαιτήσεις του έργου να είναι καλά προσδιορισμένες και να καλύπτουν αποδεκτές ανάγκες,
- η αλλαγή που επιφέρει το έργο να μην θίγει τις εργασιακές συνθήκες των χρηστών,
- η επικοινωνία με τους συμμετέχοντες στο έργο καθ’ όλη τη διάρκεια του έργου να είναι διαρκής και ουσιαστική.

Βιβλιογραφία/Αναφορές

Atkinson, P. E. (2012). Selling yourself magically-Persuasion strategies for personal and organisational change. Management Services, 56(4), 28.

Agile Alliance (2017). Agile Practice Guide, 1st ed. Newtown Square, NM, USA:Project Management Institute.

Berteig, M. (2015). The agile manifesto – essay 2: individuals and interactions over processes and tools, http://www.agileadvice.com/2015/01/13/agilemanagement/the-agile-manifesto-essay-2-individuals-and-interactions-over-processes-and-tools/.

Lach, D., Ingram, H., & Rayner, S. (2004). Maintaining the status quo: how institutional norms and practices create conservative water organizations. Tex L. Rev., 83, 2027.

Layton, M. C., & Ostermiller, S. J. (2017). Agile project management for dummies, 2nd edition. John Wiley & Sons.

Meisenbach, R. J., & Jensen, P. R. (2017). Bureaucratic theory. The International Encyclopedia of Organizational Communication, 1-13.

Nussbaum, M. C. (1995). Objectification. Philosophy & Public Affairs, 24(4), 249-291.

Renesch, J. (2006). Humanizing work: surviving in the culture of technology. Foresight.

Stellman, A., & Greene, J. (2014). Learning agile: Understanding scrum, XP, lean, and Kanban. " O'Reilly Media, Inc.".

Pfarl, W., Opelt, A., Mittermayr, R., & Gloger, B. (2013). *Agile Contracts: Creating and Managing Successful Projects with Scrum*. John Wiley & Sons, Incorporated.

Rosen, B., Furst, S., & Blackburn, R. (2007). Overcoming barriers to knowledge sharing in virtual teams. Organizational Dynamics, 36(3), 259-273.

Φιτσιλής Π., Σταμέλος Ι. & Ξένος Μ. (2009), Προγραμματισμός Έργων Πληροφορικής – Αντικειμενοστρεφείς Μεθοδολογίες, Ελληνικό Ανοικτό Πανεπιστήμιο.

Koch, A. S. (2005). Agile Software Development: Evaluating the Methods for Your Organization. Artech House. Inc, London.

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Ποια είναι τα αρνητικά χαρακτηριστικά της αντικειμενοποίησης των εργαζομένων ;

Απάντηση/Λύση

Η αντικειμενοποίηση των εργαζομένων (Nussbaum, 1995) έχει τα ακόλουθα, συνήθως αρνητικά χαρακτηριστικά για το άτομο:

- Αντιμετώπιση του ατόμου ως εργαλείου κατάλληλου για την επίτευξη των σκοπών του οργανισμού
- Έλλειψη αυτονομίας ή αυτοδιάθεσης ή άρνηση αυτής
- Μεταχείριση του ατόμου ως άβουλου όντος
- Μεταχείριση του ατόμου ως αναλώσιμου πόρου
- Θεώρηση του ατόμου και της εργασίας του ως εμπορεύσιμου προϊόντος
- Αδιαφορίας για τις επιδιώξεις και τα συναισθήματα του ατόμου

Κριτήριο αξιολόγησης 2

Ποια είναι η έννοια της «ανθρώπινης εργασίας» καθώς και τα χαρακτηριστικά της;

Απάντηση/Λύση

Η έννοια της «ανθρώπινης εργασίας» (humanizing work) (Renesch, 2006) έχει πολλές πτυχές, όπως για παράδειγμα να ενθαρρύνουμε τους εργαζόμενους για:

- δημιουργικότητα και καινοτομία,
- συνεχή μάθηση και επίλυση προβλημάτων,
- να νοιάζονται για τα άλλα μέλη της ομάδας,
- να νιώθουν υπερήφανοι για τη δουλειά τους,
- να είναι υπεύθυνοι, και
- να προάγουν το ομαδικό πνεύμα.

Κριτήριο αξιολόγησης 3

Δώστε παραδείγματα όπου ακόμη και στην ευέλικτη φιλοσοφία τα εργαλεία διευκολύνουν την εργασία

Απάντηση/Λύση

Το ευέλικτο μανιφέστο δίνει έμφαση στους ανθρώπους αλλά δεν αρνείται την ύπαρξη των διεργασιών και των εργαλείων τα οποία τις περισσότερες φορές είναι αναγκαία και χρήσιμα. Για παράδειγμα διεργασίες και εργαλεία που:

- Διευκολύνουν τη συνεργασία της ομάδας

- Επιλύουν σύνθετα τεχνικά προβλήματα

Κριτήριο αξιολόγησης 4

Ποιοι είναι κατά τη γνώμη σας οι βασικοί λόγοι που τεκμηριώνουμε ένα σύστημα λογισμικού;

Απάντηση/Λύση

Ο βασικός λόγος που δημιουργούμε παραδοτέα και γενικότερα είναι για να επικοινωνήσουμε με τους συμμετέχοντες (stakeholders) του έργου. Πιο συγκεκριμένα μπορούμε να διακρίνουμε τους παρακάτω δυο βασικούς λόγους:

- Ο πρώτος λόγος είναι η παροχή πληροφοριών που είναι απαραίτητες για την αλληλεπίδραση των συμμετεχόντων του έργου. Αυτές οι πληροφορίες μπορεί να είναι ιστορικές, τεχνικές ή προτάσεις θεμάτων συζήτησης, και έχουν ως στόχο να υποβοηθήσουν τους συμμετέχοντες στο να επιτύχουν μια πιο παραγωγική συζήτηση από ό, τι θα συνέβαινε αν δεν είχαν την πληροφορία διαθέσιμη. Όταν τα έγγραφα χρησιμοποιούνται για αυτόν τον σκοπό, τυχόν λάθη ή παραλήψεις είναι εύκολο να διορθωθούν κατά τη διάρκεια της αλληλεπίδρασης.
- Ο δεύτερος σκοπός είναι η καταγραφή των αποτελεσμάτων των συζητήσεων των συμμετεχόντων του έργου. Όταν χρησιμοποιούνται για αυτόν τον σκοπό, τα έγγραφα λειτουργούν ως συλλογικής μνήμης, ώστε τα αποτελέσματα της αλληλεπίδρασης των συμμετεχόντων να είναι λιγότερο πιθανό να αμφισβητηθούν στο μέλλον, όταν οι μνήμες των γεγονότων έχουν εξασθενίσει. Για παράδειγμα, μια σχεδιαστική απόφαση ως αποτέλεσμα της διαβούλευσης της ομάδας του έργου συνήθως λησμονείται με το πέρας του έργου.

Κριτήριο αξιολόγησης 5

Τι είναι ένα παραδοτέο;

Απάντηση/Λύση

Ορίζουμε ως παραδοτέο οποιοδήποτε μοναδικό και επαληθεύσιμο προϊόν ή αποτέλεσμα που πρέπει να παραχθεί προκειμένου να ολοκληρωθεί μία δραστηριότητα, μία φάση ή ένα έργο λογισμικού. Ο ορισμός των παραδοτέων συνδέεται στενά με τη διαχείριση του αντικειμένου των εργασιών σε ένα έργο, μια και η λίστα των παραδοτέων είναι αυτή που προσδιορίζει με ακρίβεια το αποτέλεσμα του έργου.

Κριτήριο αξιολόγησης 6

Ποια είναι τα είδη των παραδοτέων;

Απάντηση/Λύση

Τα παραδοτέα μπορεί να είναι τελικά ή ενδιάμεσα. Τελικά είναι τα παραδοτέα που αποτελούν τμήμα του τελικού προϊόντος του έργου, ενώ ενδιάμεσα χαρακτηρίζονται τα παραδοτέα που δεν αποτελούν μέρος του τελικού προϊόντος, αλλά είναι απαραίτητα για την παραγωγή του τελικού προϊόντος. Για παράδειγμα, τελικό παραδοτέο είναι ο κώδικας του συστήματος ή η τεκμηρίωση του συστήματος, ενώ ενδιάμεσο παραδοτέο είναι το έγγραφο που περιέχει την ανάλυση του συστήματος.

Κριτήριο αξιολόγησης 7

Ποια είναι οι τεχνικές συμβασιοποίησης που υποστηρίζουν την ευέλικτη φιλοσοφία;

Απάντηση/Λύση

Ορισμένες τεχνικές συμβασιοποίησης που μπορούν να υποστηρίξουν την ευέλικτη φιλοσοφία είναι οι ακόλουθες:

- **Συμβάσεις με πολυεπίπεδη δομή (Multi-tiered structure).** Αντί η συμβατική σχέση να περιγράφεται σε ένα μόνο έγγραφο, μπορούμε να επιτύχουμε μεγαλύτερη ευελιξία περιγράφοντας διαφορετικές πτυχές του έργου σε διαφορετικά έγγραφα. Σταθερές προβλέψεις όπως για παράδειγμα οι εγγυήσεις, η διαιτησία ή γενικοί όροι και αρχές της συνεργασίας μπορούν να περιγραφούν και να οριστικοποιηθούν σε μια κύρια συμφωνία. Θέματα που μεταβάλλονται (π.χ. τιμές παρεχόμενων υπηρεσιών, προδιαγραφές/περιγραφές προϊόντων μπορούν να περιγράφονται σε άλλα έγγραφα που είναι εφικτό να αλλάζουν πιο συχνά.
- **Να δίνεται έμφαση στην αξία (του λογισμικού) που παραδόθηκε.** Πολλές φορές οι σχέσεις με τους προμηθευτές θα πρέπει να ορίζονται με ορόσημα ή «πύλες φάσης» (phase-gates), δηλαδή να επικεντρωνόμαστε στο ενδιάμεσο παραδοτέο (το λογισμικό να είναι χρήσιμο και λειτουργικό σε κάθε φάση και κάθε παράδοση) και όχι σε ένα πλήρες παραδοτέο (που ποτέ δεν έρχεται). Στην περίπτωση αυτή τα ορόσημα και όροι πληρωμής συνδέονται με γνώμονα πάντα την αξία του παραδοτέου.
- **Προσαυξήσεις προκαθορισμένης τιμής (fixed price increments).** Αντί να συμφωνήσουμε εξ' αρχής για όλο το έργο και όλες τις απαιτήσεις, αποδομούμε τις απαιτήσεις του έργου σε μικρότερα τμήματα σταθερής τιμής, που περιλαμβάνουν ένα συγκεκριμένο αριθμό ιστοριών χρηστών (user stories).
- **Το έργο να μην υπερβαίνει το διαθέσιμο χρόνο και προϋπολογισμό.** Στην περίπτωση αυτή ο προϋπολογισμός και ο χρόνος είναι σταθερά αλλά μπορεί να αλλάξει το εύρος. Αυτό σημαίνει ότι όταν μια νέα απαίτηση κρίνεται αναγκαία, σημαντική και με ιδιαίτερη αξία για την επιχείρηση θα πρέπει για να ενσωματωθεί να αφαιρεθεί μια άλλη εργασία/απαίτηση.

Κριτήριο αξιολόγησης 8

Ποια είναι τα βασικά εμπόδια επικοινωνίας σε μια ομάδα έργου;

Απάντηση/Λύση

Τα εμπόδια επικοινωνίας σε ομάδες έργου και ειδικά όταν τα μέλη της ομάδας είναι πολλά και έχουν μελετηθεί εκτενώς στη βιβλιογραφία. Για παράδειγμα τα πιο βασικά είναι:

- Έλλειψη εμπιστοσύνης μεταξύ των μελών της ομάδας
- Η έλλειψη χρόνου και τα πιεστικά ορόσημα του έργου
- Η τεχνογνωσία που είναι διαθέσιμη σε κάθε μέλος της ομάδας του έργου σε σχέση με αυτή που είναι απαιτούμενη για το έργο συνολικά
- Διαφορετικές εμπειρίες και κουλτούρα μεταξύ των μελών της ομάδας έργου
- Το στυλ ηγεσίας της διοίκησης του έργου

Κριτήριο αξιολόγησης 9

Πότε οι αλλαγές στις απαιτήσεις ενός έργου γίνονται αποδεκτές πιο εύκολα;

Απάντηση/Λύση

Για να γίνει θετικά δεκτή η αλλαγή που επιφέρει ένα έργο πρέπει να υπάρξει η κατάλληλη προ- εργασία κατά τη διάρκεια του έργου. Πιο συγκεκριμένα πρέπει:

- η αλλαγή που επιφέρει το έργο να καλύπτει μια γενικώς αποδεκτή ανάγκη,
- οι απαιτήσεις του έργου να είναι καλά προσδιορισμένες και να καλύπτουν αποδεκτές ανάγκες,
- η αλλαγή που επιφέρει το έργο να μην θίγει τις εργασιακές συνθήκες των χρηστών,
- η επικοινωνία με τους συμμετέχοντες στο έργο καθ’ όλη τη διάρκεια του έργου να είναι διαρκής και ουσιαστική.

Οι ευέλικτες αρχές

There is a way to do it better — find it!

Thomas A. Edison

Σύνοψη
Οι δώδεκα αρχές της ευέλικτης φιλοσοφίας είναι οι κατευθυντήριες αρχές και περιγράφουν μια διαφορετική κουλτούρα στην οποία η αλλαγή είναι ευπρόσδεκτη και ο πελάτης είναι το επίκεντρο της κάθε εργασίας. Δείχνουν επίσης τη βασική πρόθεση της ευέλικτης φιλοσοφίας που είναι το να ευθυγραμμίσει την ανάπτυξη του λογισμικού με τις επιχειρηματικές ανάγκες στο μέγιστο βαθμό. Οι δώδεκα ευέλικτες αρχές μπορούν να υποστηρίξουν τις επιχειρήσεις στη βελτιστοποίηση του κύκλου ανάπτυξης προϊόντων και να επιτύχουν υψηλότερη ικανοποίηση πελατών, αφού οι αλλαγές είναι ευπρόσδεκτες, παραδίδεται λογισμικό που έχει αξία για τον πελάτη και μάλιστα σε σύντομο χρονικό διάστημα. Οι ευέλικτες αρχές δίνουν έμφαση στην προσφορά των εργαζομένων, δείχνοντας εμπιστοσύνη σε αυτούς, επιτρέποντάς τους να αυτο-οργανωθούν, να αναλάβουν πρωτοβουλίες και να έχουν ένα ανθρώπινο ρυθμό εργασίας.

3 Οι ευέλικτες αρχές

Σύμφωνα με το ευέλικτο μανιφέστο οι αρχές που διέπουν την ευέλικτη φιλοσοφία είναι δώδεκα και θα αναλυθούν στις επόμενες παραγράφους. Στη σχετική ιστοσελίδα (https://agilemanifesto.org/principles.html) παρουσιάζονται με περιεκτικότητα και συντομία ώστε να είναι κατανοητές αλλά και εύκολα προσπελάσιμες σε όλους τους αναγνώστες. Πιο συγκεκριμένα αναφέρεται:

> Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

> Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

> Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

> Business people and developers must work together daily throughout the project.

> Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Η ελληνική μετάφραση του παραπάνω κειμένου όπως αναφέραμε και στην εισαγωγή είναι η ακόλουθη:

Βασική προτεραιότητα αποτελεί η ικανοποίηση του πελάτη με τη συνεχή παράδοση σημαντικού τμήματος του λογισμικού από τα πρώτα κιόλας στάδια της παραγωγής (επικέντρωση στον πελάτη).

Οι μεταβαλλόμενες απαιτήσεις είναι καλοδεχούμενες ακόμα και σε προχωρημένο στάδιο ανάπτυξης. Οι διαδικασίες στην ανάπτυξη λογισμικού είναι προσαρμοσμένες στην αλλαγή με στόχο την ενίσχυση του ανταγωνιστικού πλεονεκτήματος του πελάτη. (αποδοχή της αλλαγής)

Η παράδοση τμημάτων του λογισμικού γίνεται ανά δύο εβδομάδες έως και ανά δύο μήνες. Σημαντική είναι η όσο το δυνατόν συχνότερη παράδοση (συχνή παράδοση λειτουργικότητας).

Οι προγραμματιστές και οι ειδικοί της αγοράς πρέπει να συνεργάζονται καθημερινά καθ' όλη τη διάρκεια του έργου (καθημερινή συνεργασία).

Θεμελιώνουμε τα έργα γύρω από άτομα με πάθος και ενδιαφέρον. Διαμορφώνουμε το κατάλληλο περιβάλλον, τους παρέχουμε την αναγκαία υποστήριξη και εμπιστευόμαστε την ικανότητά τους να φέρουν σε πέρας την αποστολή τους (ατομική παρακίνηση).

Η πιο αποδοτική και αποτελεσματική μέθοδος για τη μετάδοση πληροφορίας προς και εντός της ομάδας ανάπτυξης λογισμικού είναι η συνομιλία πρόσωπο με πρόσωπο (άμεση επικοινωνία).

Το λογισμικό που λειτουργεί είναι το κύριο μέτρο προόδου (επικέντρωση στην αξία).

Οι ευέλικτες διαδικασίες προάγουν την αειφόρο ανάπτυξη. Οι χορηγοί, η ομάδα ανάπτυξης λογισμικού και οι χρήστες θα πρέπει να είναι σε θέση να διατηρούν ένα σταθερό ρυθμό επ' αόριστον (σταθερή απόδοση).

Η διαρκής έμφαση στην τεχνική αρτιότητα και στην εύρυθμη σχεδίαση ενισχύουν την ευελιξία (επιδίωξη της ποιότητας).

Η απλοποίηση, με την έννοια της υλοποίησης στόχων με σύντομο αλλά και με αποτελεσματικό τρόπο, είναι βασική αρχή (επιδίωξη της απλότητας).

Οι καλύτερες αρχιτεκτονικές, απαιτήσεις και σχέδια προκύπτουν από ομάδες που αυτο-οργανώνονται (αυτο-οργάνωση).

Σε τακτά χρονικά διαστήματα η ομάδα συζητά τρόπους, που την βοηθούν να γίνει περισσότερο αποτελεσματική και επαναπροσδιορίζει τη συμπεριφορά της (διαρκής βελτίωση).

Στις επόμενες παραγράφους θα αναλύσουμε τις παραπάνω αρχές παρουσιάζοντας το θεωρητικό τους υπόβαθρο.

3.1 Επικέντρωση στον πελάτη και η συνεχής παράδοση λογισμικού

3.1.1 Η ικανοποίηση του πελάτη

Η ικανοποίηση και η επικέντρωση στον πελάτη είναι ένα θέμα το οποίο έχει μελετηθεί εκτενώς, διότι όλα τα συστήματα διαχείρισης ποιότητας δίνουν ιδιαίτερη έμφαση σε αυτή τη διάσταση. Στη βιβλιογραφία συναντάμε την ικανοποίηση του πελάτη με τους όρους, ικανοποίηση καταναλωτή (Consumer Satisfaction), ικανοποίηση πελάτη (Customer Satisfaction) ή απλά ικανοποίηση (Satisfaction).

Ο όρος πελάτης είναι ένας γενικός όρος και με τον όρο αυτό αναφερόμαστε γενικότερα σε αυτούς που επωφελούνται από το αποτέλεσμα του έργου ή θα χρησιμοποιούν το προϊόν που παράγει το έργο. Σε γενικές γραμμές, θα πρέπει να διαφοροποιήσουμε τον όρο πελάτη από τον όρο χρήστη. Με τον όρο πελάτη (customer) αναφερόμαστε στην οντότητα που παραγγέλνει το έργο, ενώ με τον όρο χρήστης (user)

αναφερόμαστε σε αυτόν που χρησιμοποιεί αυτό που παράγεται από το έργο. Επίσης με τον όρο πελάτης αναφερόμαστε σε αυτούς που χρηματοδοτούν ή διοικούν το έργο γενικότερα. Αυτοί είναι, ο χρηματοδότης του έργου που ονομάζεται και σπόνσορας (sponsor), καθώς και η ομάδα διοίκησης του έργου (project management team).

Η ικανοποίηση πελατών συνδέεται άμεσα με την ποιότητα και είναι αναπόσπαστο κομμάτι της Διοίκησης Ολικής Ποιότητας (ΔΟΠ). Η ΔΟΠ είναι μια φιλοσοφία διοίκησης η οποία θεωρεί ότι η ικανοποίηση των πελατών, και η εξασφάλιση της επιχειρηματικής επιτυχίας είναι άρρηκτα συνδεδεμένες με τους στόχους της επιχείρησης. Βασικός στόχος της ΔΟΠ είναι η ικανοποίηση τόσο των εσωτερικών όσο και των εξωτερικών πελατών, δημιουργώντας έτσι την αλυσίδα αξίας των υπηρεσιών (Δερβιτσιώτης, 1993).

Είναι δύσκολο να δοθεί ο ακριβής ορισμός της έννοιας της ικανοποίησης, καθότι ορίζεται διαφορετικά για τον κάθε άνθρωπο αλλά και σε κάθε πεδίο εφαρμογής. Επίσης η ικανοποίηση του πελάτη είναι μία πολύπλευρη έννοια που έχει προσδιοριστεί με πολλούς και διαφορετικούς τρόπους.

Ένας γενικός, αλλά και χρήσιμος ορισμός είναι αυτός των (Farris et al, 2010), όπου η ικανοποίηση του πελάτη ορίζεται ως ο αριθμός των πελατών ή το επί τοις εκατό ποσοστό των συνολικών πελατών, οι οποίοι εξέφρασαν ότι είχαν εμπειρία από την επιχείρηση και τα προϊόντα της ή τις υπηρεσίες της και η βαθμολογία τους αναφορικά με το βαθμό που καλύπτουν τις απαιτήσεις τους είναι ο μέγιστος προσδοκώμενος ή ακόμη και πάνω από αυτόν.

Αντίστοιχα, ο (Juran, 1993) αναφέρει ότι ικανοποίηση του πελάτη σημαίνει ότι το προϊόν ή η υπηρεσία ανταποκρίνεται σε δυο βασικά στοιχεία:

- στα χαρακτηριστικά του προϊόντος – υπηρεσίας, και
- στην απουσία ελαττωμάτων από το προϊόν – υπηρεσία, που αναφέρεται στην ποιότητα συμμόρφωσης ως προς τις προδιαγραφές.
- Η ικανοποίηση του πελάτη συνδέεται με τα συναισθήματα ευχαρίστησης ή δυσαρέσκειας ενός ατόμου που προκύπτουν από την υποκειμενική σύγκριση της απόδοσης (αποτέλεσμα) ενός προϊόντος σε σχέση με τις προσδοκίες του (Kotler et al, 2001). Είναι η αίσθηση που προκύπτει από τη χρήση ενός προϊόντος ή μιας υπηρεσίας ως αποτέλεσμα της αξιολόγησης αυτής.

Οι πιο σημαντικοί λόγοι που μια επιχείρηση καταφεύγει στη μέτρηση της ικανοποίησης των πελατών της, είναι οι εξής (Dutka, 1995):

- Η ικανοποίηση του πελάτη επειδή αποτελεί την πιο αντικειμενική πληροφορία που προέρχεται από την αγορά, δίνει τη δυνατότητα στην επιχείρηση να εκτιμήσει την τρέχουσα κατάσταση της αγοράς και να διαμορφώσει ανάλογα την μελλοντική της πολιτική.
- Η ικανοποίηση των πελατών και τα συμπεράσματα που προκύπτουν από τις σχετικές μελέτες επιτρέπουν τον προσδιορισμό των προτεραιοτήτων της επιχείρησης, και δίνουν τη δυνατότητα στην επιχείρηση να προβεί σε διορθωτικές ενέργειες όπου κρίνεται αναγκαίο.
- Η μέτρηση της ικανοποίησης των πελατών επιτρέπει τον προσδιορισμό «ευκαιριών» στη συγκεκριμένη αγορά.
- Η εφαρμογή των αρχών της συνεχούς βελτίωσης που επιβάλλονται από πρότυπα και πολιτικές ποιότητας, απαιτεί την ύπαρξη συγκεκριμένης διαδικασίας μέτρησης της ικανοποίησης των πελατών. Με αυτό τον τρόπο οι ενέργειες βελτίωσης βασίζονται σε πραγματικά δεδομένα, που είναι σύμφωνα με τις ανάγκες και τις επιθυμίες των πελατών.
- Η μέτρηση της ικανοποίησης μπορεί να βοηθήσει στην κατανόηση των γενικότερων αντιλήψεων του πελάτη και πιο συγκεκριμένα στον προσδιορισμό και την ανάλυση των αναγκών, των προσδοκιών και των επιθυμιών του.
- Το πρόβλημα της ύπαρξης διαφορετικής αντίληψης της ικανοποίησης ανάμεσα στον πελάτη και τη διοίκηση της εταιρείας μπορεί να προσδιοριστεί από την υλοποίηση ενός προγράμματος

μέτρησης της ικανοποίησης. Με αυτόν τον τρόπο δίνεται η δυνατότητα να αμβλυνθούν αυτές οι διαφορές αντίληψης.

Είναι εμφανές, ότι η ικανοποίηση του πελάτη είναι μία πολυδιάστατη έννοια που περιλαμβάνει την αξιολόγηση πολλαπλών παραμέτρων ως προς τον σκοπό, τον προϋπολογισμό, το χρονοδιάγραμμα και άλλες παραμέτρους του πληροφοριακού συστήματος ή των υπηρεσιών.

Οι Jones and Sasser (1995) προσδιορίζουν τα τέσσερα βασικά στοιχεία που έχουν επιπτώσεις στην ικανοποίηση του πελάτη, τα οποία είναι:

- Τα βασικά χαρακτηριστικά του προϊόντος ή της υπηρεσίας
- Οι βασικές υπηρεσίες υποστήριξης
- Οι διαδικασίες ανάκαμψης στην κακή εμπειρία που τυχόν έχει ο πελάτης
- Η παροχή άριστης υπηρεσίας.

3.1.2 Η ικανοποίηση πελατών στα ευέλικτα έργα

Τη σημασία της ικανοποίησης του πελάτη στα έργα αναφέρουν πλήθος μελετών. Οι ερευνητές επισημαίνουν ότι σε ένα δυναμικό ανταγωνιστικό περιβάλλον οργανισμών που αναλαμβάνουν να υλοποιήσουν έργα (projects) ιδιαίτερα σημαντικό ρόλο παίζει η ικανοποίηση του πελάτη. Ξεκινώντας από τη βασικότερη ομάδα σε ένα έργο, που είναι οι πελάτες του έργου, ο πελάτης παίζει καθοριστικό ρόλο σε όλη τη διαδικασία υλοποίησης του έργου. Η ικανοποίησή του αποτελεί βασικό στόχο όλων των οργανισμών και των έργων σε όλες τις βιομηχανίες, εξαιτίας του μεγάλου ανταγωνισμού και των αυξανόμενων απαιτήσεών του (Karna et al, 2009). Για τον πελάτη δεν έχει καμία αξία το τελικό παραδοτέο έργο, εάν οι απαιτήσεις του δεν ικανοποιηθούν. Ουσιαστικά η μεγιστοποίηση της ικανοποίησης του πελάτη έχει προϋπόθεση την μεγιστοποίηση της ποιότητας του έργου, εντός των οικονομικών και χρονικών πλαισίων που καθορίστηκαν από κοινού. Στα έργα γενικά η ικανοποίηση του πελάτη αυξάνεται εφόσον ενισχύεται η επικοινωνία του με τους διαχειριστές του έργου, αλλά και όταν υπάρχει αποτελεσματική διοίκηση του έργου (Karna et al, 2009).

Προϋποθέσεις για να ληφθούν υπόψη γενικότερα οι απαιτήσεις, αλλά και οι παρατηρήσεις των πελατών για την προσαρμογή των έργων σε αυτά, είναι η βελτίωση της επικοινωνίας με τους πελάτες, η κατανόηση των αναγκών/απαιτήσεών τους για παρόμοια μελλοντικά έργα, ο εντοπισμός διαφορών της αρχικής ιδέας και της εικόνας που έχει ο πελάτης και το πώς διαμορφώθηκε μετά την παραλαβή του έργου, ο προσδιορισμός των σημείων προς βελτίωση, η ανάδειξη των πλεονεκτημάτων και μειονεκτημάτων της εταιρείας σε σχέση με τις άλλες και τους στόχους της για βελτίωση στο μέλλον, κ.α. (Karna et al, 2009).

Σύμφωνα με τους (Kotler et al, 2013), η ικανοποίηση του πελάτη στα έργα δύναται να οριστεί ως η αντίληψή του για την ποιότητα του έργου και του κατά πόσο καλύπτει τις προκαθορισμένες προσδοκίες – απαιτήσεις του. Οι πελάτες συγκρίνουν την απόδοση του έργου σε σχέση με προκαθορισμένα εσωτερικά πρότυπα (standards). Είναι ικανοποιημένοι όταν το έργο υπερέχει σε τεχνικά και λειτουργικά χαρακτηριστικά σε σχέση με τα πρότυπα αυτά και αντίστοιχα δυσαρεστημένοι εφόσον υπολείπονται αυτών. Επιπλέον όταν το έργο υλοποιηθεί εντός προκαθορισμένου χρονοδιαγράμματος και προϋπολογισμού, αυτό μεγιστοποιεί την ικανοποίηση των πελατών.

Η ικανοποίηση των πελατών δεν είναι έννοια στατική αλλά δυναμική. Μεταβάλλεται και επηρεάζεται από τις συνθήκες/παραμέτρους λειτουργίας του έργου, σε όλη τη διάρκεια της ζωής του, κατά την υλοποίησή του αλλά και μετά την παραλαβή του έργου (Shenhar et al., 2001). Ο πελάτης συνεχώς κρίνει με βάση τα δικά του κριτήρια και αναλόγως νιώθει σε μικρότερο ή μεγαλύτερο βαθμό ικανοποίηση από τα συμπεράσματα που προκύπτουν.

Γνωρίζοντας το επίπεδο ικανοποίησης των πελατών στα έργα που παραλαμβάνουν, είναι δυνατό να ενισχυθεί η επιτυχία των οργανισμών που τα υλοποιούν. Αντίστοιχα με την περίπτωση της ικανοποίησης πελατών από τις υπηρεσίες, γνωρίζοντας τις παρατηρήσεις τους, μέσω ανατροφοδότησης και διόρθωσης είναι δυνατό να μεγιστοποιηθεί η ικανοποίηση του πελάτη στο μέλλον, σε νέα έργα.

Σύμφωνα με τη μελέτη της Σαρμανιώτη (2020), έγινε αντιληπτή η άρρηκτη σχέση μεταξύ ικανοποίησης του χρήστη-πελάτη και της ενεργού συμμετοχής του στην ανάπτυξη του συστήματος και της παράδοσης του συστήματος. Σύμφωνα με τη μελέτη αυτή οι παράγοντες που επηρεάζουν την ικανοποίηση του πελάτη, η οποία με την σειρά της συντελεί στην επιτυχία ενός έργου ανάπτυξης πληροφοριακού συστήματος, είναι:

- Η ικανοποίηση του πελάτη επηρεάζεται θετικά από τη συμμετοχή του στην ανάπτυξη του έργου πληροφοριακών συστημάτων και συμβάλλει στην επιτυχία του. Η εμπλοκή του πελάτη στα περισσότερα κομβικά σημεία υλοποίησης του έργου μεγιστοποιεί την ικανοποίησή του, οδηγώντας στην αποδοχή και επιτυχία του έργου.
- Η ικανοποίηση του πελάτη δεν είναι ένα φαινόμενο στατικό αλλά μεταβάλλεται κατά τη διάρκεια υλοποίησης του έργου και διαφέρει ανάλογα με τη φάση στην οποία βρίσκεται το κάθε έργο.
- Υψηλότερη ικανοποίηση του πελάτη επιτυγχάνεται με τη χρήση ευέλικτων μεθόδων. Επιπλέον, η έγκριση και αποδοχή του πελάτη σε κάθε ενδιάμεσο στάδιο υλοποίησης του έργου, οδηγεί στην μέγιστη ικανοποίηση.
- Ένα ευέλικτο έργο μπορεί να είναι επιτυχημένο ακόμη και σε περίπτωση που δεν καλυφθούν όλες οι απαιτήσεις του πελάτη. Η συμμετοχή του πελάτη στα ενδιάμεσα στάδια του έργου οδηγεί σε ανάληψη από τον πελάτη της ευθύνης για το τελικό αποτέλεσμα, κάνοντάς τον να αισθάνεται σημαντικός και δημιουργικός με αποτέλεσμα αυτό να δρα σαν ενδιάμεσος μεσολαβητής για την αποδοχή του έργου.
- Η ποιότητα του έργου επηρεάζει θετικά την ικανοποίηση του πελάτη.

Για την επιτυχημένη ενεργοποίηση και ικανοποίηση των πελατών μας θα πρέπει να εστιάσουμε και να ακολουθήσουμε ένα σύνολο τεχνικών. Οι βασικότερες τεχνικές που χρησιμοποιούνται είναι οι ακόλουθες:

- Σε όλες τις ευέλικτες ομάδες έργου θα πρέπει να υπάρχει ένας ιδιοκτήτης προϊόντος (product owner). Ο βασικός του ρόλος είναι να διασφαλίσει ότι οι απαιτήσεις είναι κατανοητές από την ομάδα ανάπτυξης του έργου.
- Ο ιδιοκτήτης του προϊόντος κάνει ιεράρχηση των απαιτήσεων είτε βάση της αξίας που έχουν αυτές για την επιχείρηση είτε με άλλα κριτήρια (π.χ. κίνδυνος)
- Ο ιδιοκτήτης του προϊόντος έχει συνεχή επικοινωνία με την ομάδα ανάπτυξης, σε καθημερινή βάση, ώστε να επεξηγήσει λειτουργίες, να διευκρινίσει χαρακτηριστικά του συστήματος, να αποσαφηνίζει απαιτήσεις και να κάνει ιεράρχηση αυτών, να λαμβάνει αποφάσεις, να δίνει ανατροφοδότηση και να απαντά γρήγορα στις πολλές ερωτήσεις που προκύπτουν κατά τη διάρκεια ενός έργου.
- Η συχνή παράδοση λειτουργικών απαιτήσεων επιτρέπει στον ιδιοκτήτη του προϊόντος και στον πελάτη να έχουν πλήρη εικόνα του τρόπου ανάπτυξης του προϊόντος.
- Καθώς η ομάδα ανάπτυξης συνεχίζει να παρέχει σε τακτά χρονικά διαστήματα πλήρεις, λειτουργικές, δυνητικά αποσπώμενες λειτουργίες, η αξία για την επιχείρηση του συνολικού προϊόντος αυξάνεται σταδιακά, όπως και οι λειτουργικές του δυνατότητες.
- Αντίστοιχα, ο πελάτης συσσωρεύει αξία λαμβάνοντας νέες, έτοιμες προς χρήση λειτουργίες καθ' όλη τη διάρκεια του έργου, αντί να περιμένει μέχρι το τέλος του έργου για την παράδοση του προϊόντος.

3.1.3 Η δημιουργία επιχειρηματικής αξίας

Ο όρος «επιχειρηματική αξία» (Business Value – BV) χρησιμοποιείται στη διοίκηση επιχειρήσεων ως ένας άτυπος όρος που περιλαμβάνει όλες τις μορφές αξίας που καθορίζουν την κατάσταση και την ευημερία της επιχείρησης μακροπρόθεσμα. Στο πλαίσιο της ευέλικτης ανάπτυξης, ο όρος «επιχειρηματική αξία» εμφανίζεται πολύ συχνά ως ένας από τους βασικούς και διαρκής στόχους του κάθε έργου. Η «επιχειρηματική

αξία» δημιουργείται μέσω της ανάπτυξης του λογισμικού και πολύ συχνά κάνουμε την προσπάθεια να μεταφράσουμε την επιχειρηματική αξία σε οικονομική αξία (ευρώ), για παράδειγμα το νέο σύστημα διαχείρισης πελατών μας επέτρεψε να αυξήσουμε τις πωλήσεις κατά 20%. Από την άλλη πλευρά, είναι βέβαιο ότι ο όρος «επιχειρηματική αξία» δεν μπορεί να περιοριστεί και να μεταφραστεί μόνο σε οικονομική αξία, αφού είναι βέβαιο ότι είναι μια πολυδιάστατη έννοια.

Μια σημαντική παρατήρηση είναι ότι η «επιχειρηματική αξία» δεν μπορεί να προσδιοριστεί αντικειμενικά αφού διαφέρει μεταξύ των συμμετεχόντων (stakeholders) και συνεπώς προσδιορίζεται υποκειμενικά. Η αξία που προσδίδουν οι πελάτες στα χαρακτηριστικά του λογισμικού είναι αυτή που προσδιορίζει και την προτεραιότητα υλοποίησης (Racheva et al., 2009; Bakalova, 2014).

3.2 Η αλλαγή στα ευέλικτα έργα

3.2.1 Η διαχείριση της αλλαγής

Σύμφωνα με τον Έλληνα φιλόσοφο Ηράκλειτο, «τα πάντα ρει», μια ιδέα που περιγράφει τη συνεχή αλλαγή που διέπει ως νόμος το σύμπαν. Δική του είναι και η φράση «Κανείς δεν μπορεί να μπει στο ίδιο ποτάμι δύο φορές», εικόνα που υποδηλώνει τον ποταμό που παραμένει ίδιος, ενώ το νερό που κυλάει μέσα του αλλάζει διαρκώς. Επίσης έχει ειπωθεί, από τον Kent Beck στο δημοφιλές βιβλίο του Extreme Programming (Beck, 2005) ότι, «Η επιχείρηση αλλάζει. Η τεχνολογία αλλάζει. Η ομάδα αλλάζει. Τα μέλη της ομάδας αλλάζουν. Το πρόβλημα δεν είναι η αλλαγή, από μόνη της, γιατί η αλλαγή είναι κάτι που θα συμβεί είτε το θέλουμε είτε όχι. Το πρόβλημα είναι η αδυναμία μας να αντιμετωπίσουμε την αλλαγή όταν αυτή έρχεται».

Αυτή η αρχή τονίζει τη σημασία της υιοθέτησης της αλλαγής στα ευέλικτα έργα. Η υιοθέτηση της αλλαγής μας βοηθά να διασφαλίσουμε ότι ο πελάτης είναι ικανοποιημένος από το τελικό προϊόν του έργου. Κάθε έργο υπάρχει στα πλαίσιο ενός τουλάχιστον οργανισμού, δηλαδή μέσα σε ένα οργανωτικό πλαίσιο. Η κουλτούρα του οργανισμού, οι οργανωτικές δομές και οι εφαρμοζόμενες πολιτικές μπορούν να επηρεάσουν τόσο την κατεύθυνση όσο και το αποτέλεσμα οποιουδήποτε έργου. Αυτός ο συνδυασμός παραγόντων, αυτή η δυναμική αποτελεί πρόκληση για όλους όσους διαχειρίζονται ένα έργο.

Η διαχείριση της οργανωτικής αλλαγής απαιτεί τις κατάλληλες δεξιότητες και τις τεχνικές ώστε τελικά να υποστηριχθεί η ευέλικτη φιλοσοφία. Σύμφωνα με τον οδηγό πρακτικής για τη διαχείριση των αλλαγών του Project Management Institute – PMI (2013), περιγράφεται μια ολοκληρωμένη και ολιστική προσέγγιση για την επιτυχή εισαγωγή των αλλαγών. Η προσέγγιση που παρουσιάζεται εκεί περιλαμβάνει:

1. Μοντέλα για την περιγραφή της δυναμικής των αλλαγών.
2. Ένα πλαίσιο για την υλοποίηση των αλλαγών.
3. Εφαρμογή πρακτικών διαχείρισης αλλαγών σε επίπεδο έργου, προγράμματος και χαρτοφυλακίου.

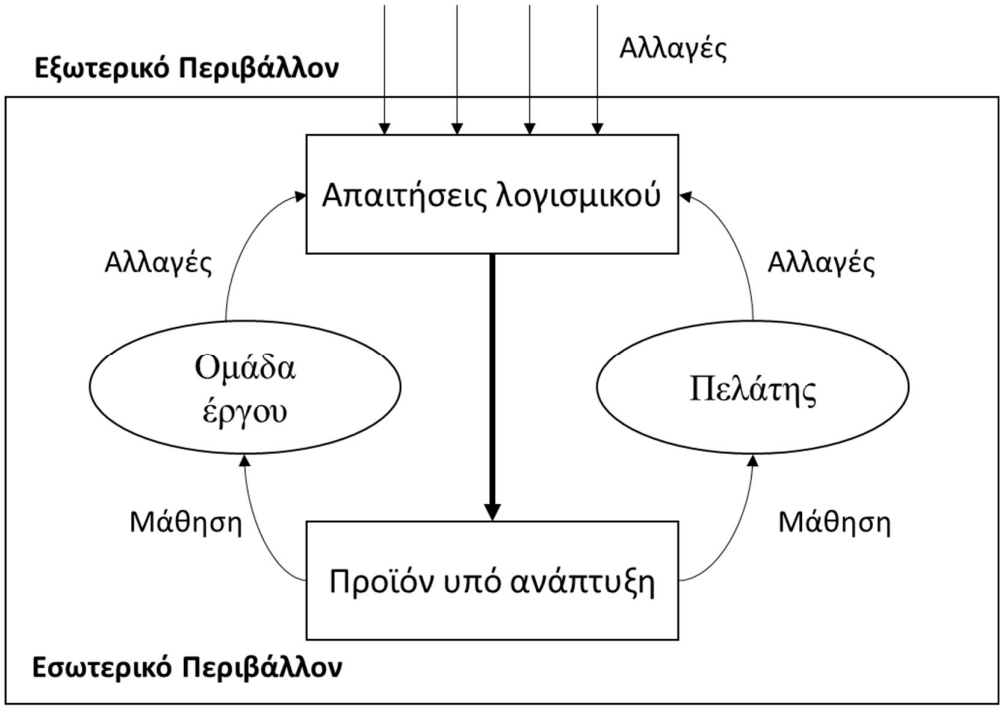
Όλα τα έργα αφορούν αλλαγές. Ωστόσο, υπάρχουν δύο βασικοί παράγοντες που κάνουν τη διαχείριση των αλλαγών αναγκαία:

- Αλλαγές που σχετίζονται με την επιταχυνόμενη παράδοση. Οι ευέλικτες προσεγγίσεις δίνουν έμφαση στην παράδοση αποτελεσμάτων του έργου νωρίς και συχνά. Ωστόσο, ο οργανισμός του πελάτη ενδέχεται να μην είναι πλήρως προετοιμασμένος να ενσωματώσει στη λειτουργία του, το νέο σύστημα/λογισμικό και μάλιστα λογισμικό που παραδίδεται με σταδιακό τρόπο και αυξητικά. Η συνεχής παράδοση είναι μόνο εφικτή αν είμαστε σε θέση να διαχειριστούμε αποτελεσματικά τις αλλαγές. Θα πρέπει να σημειωθεί ότι κάθε καθυστέρηση στην αποδοχή και στη χρήση του συστήματος λογισμικού μειώνει την απόδοση της επένδυσης, με τη λογική ότι η ανάπτυξη λογισμικού είναι μια σημαντική επένδυση για την επιχείρηση, καθυστερημένη ευθυγράμμιση από τις απαιτήσεις των πελατών, απώλεια μεριδίου αγοράς, κ.λπ..
- Αλλαγές που σχετίζονται με την ευέλικτη φιλοσοφία. Οι οργανισμοί που αρχίζουν να χρησιμοποιούν ευέλικτες προσεγγίσεις βιώνουν επίσης πολλές αλλαγές. Η μεγαλύτερη

απαιτούμενη συνεργασία ενδέχεται να απαιτεί συχνότερες μεταθέσεις ατόμων μεταξύ ομάδων, ή τμημάτων ή μεταξύ του οργανισμού και των προμηθευτών του. Επίσης, ο τρόπος εργασίας σε ευέλικτα έργα είναι διαφορετικός αφού το αντικείμενο του έργου αποδομείται και υλοποιείται μέσω μιας σειράς πρωτότυπων, γεγονός που απαιτεί διαχείριση της οργανωτικής αλλαγής.

Όπως έχουμε ήδη πει ένα έργο είναι συνυφασμένο με την έννοια της αλλαγής, αφού στη γενική περίπτωση το έργο υλοποιεί την ανάγκη της επιχείρησης να εφαρμόζει τους στρατηγικούς της στόχους και συνεπώς να αλλάξει. Οι αλλαγές σε ένα έργο μπορεί να προέρχονται είτε από το εσωτερικό, είτε από το εξωτερικό περιβάλλον (βλέπε Εικόνα 3.1):

- Εσωτερικές αλλαγές, οι οποίες προέρχονται από το γεγονός ότι στην ευέλικτη φιλοσοφία η διαδικασία ανάπτυξης ενός προϊόντος είναι μια διαδικασία μάθησης, όπου τόσο ο πελάτης όσο και η ομάδα έργου μαθαίνουν. Καθ' όλη την πορεία του έργου μαθαίνουμε τι είναι εφικτό/ανέφικτο, απλοποιεί την εργασία, είναι σημαντικό, έχει προτεραιότητα, κ.λπ.
- Εξωτερικές αλλαγές, οι οποίες προέρχονται από το εξωτερικό περιβάλλον της επιχείρησης και μπορεί να προέρχονται από αλλαγές στη αγορά, στις ανάγκες των πελατών, το νομικό πλαίσιο, κ.λπ. Βέβαια η σχέση αυτή είναι δυναμική, διότι οι αλλαγές στο περιβάλλον επηρεάζουν τον σύστημα αλλά ταυτόχρονα η εισαγωγή του συστήματος επηρεάζει το περιβάλλον και το μεταβάλει (Apello, 2011).



Εικόνα 3.1: *Εσωτερικές και εξωτερικές αλλαγές*

Συμπερασματικά, με την εφαρμογή αυτής της αρχής στα ευέλικτα έργα:

- Η διαδικασία αλλαγής απλοποιείται: Η αλλαγή είναι μια πολύ σημαντική πτυχή της ευέλικτης φιλοσοφίας και είναι σαφώς διαφορετική από την παραδοσιακή προσέγγιση, η οποία αρνείται την αλλαγή. Ο όρος «ερπυσμός πεδίου» (scope creep) αντικατοπτρίζει την αρνητική χροιά των αλλαγών στο εύρος του έργου (scope) στην παραδοσιακή προσέγγιση. Αγκαλιάζοντας τις αλλαγές και έχοντας ένα απλό σύστημα διαχείρισης αυτών σε ένα ευέλικτο έργο, η ομάδα έργου μπορεί να αφιερώσει λιγότερο χρόνο στην απαιτούμενη «γραφειοκρατία» για να εγκριθούν και να

οριστικοποιηθούν αυτές οι αλλαγές και περισσότερο χρόνο στην ανάπτυξη των χαρακτηριστικών και των λειτουργιών του συστήματος, αφού αυτές είναι που παρέχουν αξία στους πελάτες.

- Οι αλλαγές είναι ευπρόσδεκτες ανά πάσα στιγμή στο έργο και όχι μόνο στην αρχή. Σκοπός μας είναι να προσφέρουμε στον πελάτη ένα σύστημα με αξία.
- Ο πελάτης κερδίζει ανταγωνιστικό πλεονέκτημα, αφού η γρήγορη ανταπόκριση στις αλλαγές, που παρουσιάζουν οι ευέλικτες ομάδες και η άμεση προσαρμογή σε αυτές επιτρέπει στους πελάτες να αξιοποιήσουν τις ανταγωνιστικές ευκαιρίες που προκύπτουν.

3.2.2 Συχνές παραδόσεις

Η πιο σημαντική ιδιότητα οποιουδήποτε έργου, μεγάλου ή μικρού, ευέλικτου ή μη, είναι η δημιουργία και παράδοση ελεγμένου κώδικα στους πελάτες σε μικρά και τακτά χρονικά διαστήματα. Τα πλεονεκτήματα είναι πάρα πολλά:

- Οι χορηγοί (sponsors) λαμβάνουν πληροφορία σχετικά με την πρόοδο του έργου.
- Οι χρήστες (users) έχουν την ευκαιρία να δοκιμάσουν τη λειτουργεία του συστήματος και να ανακαλύψουν εάν το αποτέλεσμα είναι αυτό που πραγματικά χρειάζονται.
- Οι μηχανικοί λογισμικού έχουν τη δυνατότητα να επιβεβαιώσουν τις παραδοχές που έκαναν και να επιλύσουν αδιέξοδα.
- Η ομάδα έργου μπορεί να διορθώσει τις διαδικασίες ανάπτυξης και παράδοσης του συστήματος και εμψυχώνεται κάθε φορά που παραδίδει μια νέα έκδοση με αξία για τον τελικό χρήστη.

Όλα αυτά τα πλεονεκτήματα προέρχονται από τη συχνή παράδοση του λογισμικού (Cockburn, 2010). Η λήψη ανατροφοδότησης είναι ζωτικής σημασίας για την ικανότητα ενός οργανισμού να διορθώνει την πορεία του, δηλαδή εάν κάτι δεν είναι απολύτως σωστό ή δεν ικανοποιεί τις ανάγκες του πελάτη.

Η έννοια των συχνών παραδόσεων είναι άμεσα συνυφασμένη με την έννοια της συνεχούς παράδοσης (Continuous Delivery - CD) που υλοποιείται στην προσέγγιση DevOps. Η λέξη DevOps προέρχεται από τα αρχικά των λέξεων (Development και Operations) και εν συντομία αποτελεί μια φιλοσοφία αλλά και ένα σύνολο αρχών μηχανικής λογισμικού που εστιάζονται στην αυτοματοποίηση της ανάπτυξης και λειτουργίας του λογισμικού στοχεύοντας σε συνεχείς παραδόσεις του λογισμικού στον πελάτη. Η προσέγγιση DevOps θα παρουσιαστεί αναλυτικά στο 6^{ο} κεφάλαιο.

Η συνεχής παράδοση (CD) είναι ένας βασικός στόχος της προσέγγισης DevOps και αποτελεί και αυτή με τη σειρά της ένα σύνολο γενικών αρχών μηχανικής λογισμικού που επιτρέπουν τη συχνή κυκλοφορία νέου λογισμικού μέσω της χρήσης αυτοματοποιημένων δοκιμών και συνεχούς ενσωμάτωσης (Continuous Integration – CI).

Επομένως, η συνεχής παράδοση είναι η δυνατότητα να βάζουμε σε λειτουργία νέες εκδόσεις του λογισμικού καθημερινά ή ακόμη και πολλές φορές την ίδια ημέρα. Οι εκδόσεις αυτές περιλαμβάνουν αλλαγές όλων των τύπων - συμπεριλαμβανομένων νέων λειτουργιών, αλλαγών διαμόρφωσης, επιδιόρθωση σφαλμάτων κ.λπ.

3.3 Η ομάδα στην ευέλικτη προσέγγιση

3.3.1 Η αυτοοργάνωση της ομάδας

Ενώ η ιδέα της ομάδας αυτοοργάνωσης εισήχθη στην τεχνολογία λογισμικού μέσω των ευέλικτων αρχών, στο παρελθόν έχει μελετηθεί εκτενώς σε διάφορους άλλους κλάδους. Χαρακτηριστικό παράδειγμα η μελέτη του φαινομένου της αυτο-οργάνωσης σε ομάδες Άγγλων ανθρακωρύχων, που περιγράφονται ως αυτοδιαχειριζόμενα, συστήματα μάθησης αποτελούμενα από 10-15 άτομα που μοιράζονται ευθύνες

διοίκησης (Trist , 1981). Οι αρχές της αυτο-οργάνωσης έχουν περιγραφεί στη βιβλιογραφία ως ομάδες που λειτουργούν (Morgan, 1998; Hoda et al., 2016):

- έχοντας ελάχιστη καθοδήγηση από την ανώτερη διοίκηση ή ανυπαρξία προδιαγραφών για την εκτέλεση των διεργασιών σε αντίθεση με τα διεργασιοκεντρικά (process-oriented) συστήματα ποιότητας που περιγράφουν με λεπτομέρεια το κάθε βήμα,
- οι καθημερινές αποφάσεις λαμβάνονται αποκλειστικά από την ομάδα,
- υπάρχει επαρκής ποικιλία στις διαθέσιμες δεξιότητες εντός της ομάδας, ώστε η ομάδα να μπορεί να καλύψει μια ποικιλία επιχειρηματικών απαιτήσεων,
- υπάρχει πλεονασμός λειτουργιών (redundancy) ως προς την ικανότητα της ομάδας να συμπληρώνει ή και να αντικαθιστά μέλη της σε συγκεκριμένες λειτουργίες όταν αυτό απαιτείται, και
- η ομάδα έχει την ικανότητα να μαθαίνει νέες δεξιότητες αλλά και να ανακαλύπτει νέους καλύτερους τρόπους για την εκτέλεση δραστηριοτήτων.

Η ομαδική εργασία είναι ζωτικής σημασίας για ευέλικτα έργα. Η δημιουργία σωστού λογισμικού και γενικότερα καλών προϊόντων απαιτεί συνεργασία όλων των μελών της ομάδας έργου, συμπεριλαμβανομένων των πελατών και των συμμετεχόντων.

Μερικά από τα χαρακτηριστικά της «ομάδας» με την ευέλικτη έννοια:

- είναι μια **μικρή ομάδα ανθρώπων**, στην οποία έχει ανατεθεί στο ίδιο έργο ή δραστηριότητα, και σχεδόν όλα τα μέλη της ομάδας έχουν πλήρη απασχόληση εντός της ομάδας. Ένα μικρό τμήμα των μελών της ομάδας μπορεί να είναι μερικής απασχόλησης,
- έχει **κοινή ευθύνη** αφού τα αποτελέσματα, είτε θετικά είτε αρνητικά θα αποδοθούν σε ολόκληρη την ομάδα και όχι σε κάποιο συγκεκριμένο μέλος της ομάδας,
- αναμένεται να **διαθέτει όλες τις απαραίτητες ικανότητες και γνώσεις** (cross functional team), είτε πρόκειται για τεχνικές γνώσεις (προγραμματισμός, σχεδιασμός, έλεγχος), είτε γνώσεις διοίκησης (ικανότητα λήψης αποφάσεων), είτε γνώση του επιχειρηματικού περιβάλλοντος,
- έχει τη **δυνατότητα να λαμβάνει αποφάσεις** (empowered) συνήθως με ομοφωνία των μελών (consensus) και
- είναι αυτοδιοικούμενη (self-directed) και αυτοοργανούμενη (self-managed).

Συνεπώς, οι ευέλικτες ομάδες είναι ομάδες που αυτοοργανώνονται και αποτελούνται από «άτομα που διαχειρίζονται μόνα τους το φόρτο εργασίας, μεταθέτουν την εργασία μεταξύ τους με βάση τις ανάγκες και συμμετέχουν στη λήψη αποφάσεων της ομάδας». Οι αυτοοργανωμένες ομάδες πρέπει να έχουν κοινή εστίαση, αμοιβαία εμπιστοσύνη, σεβασμό και την ικανότητα να οργανώνονται συνεχώς ώστε να αντιμετωπίσουν τις νέες προκλήσεις. Οι ομάδες που αυτοοργανώνονται δεν είναι ομάδες χωρίς ηγέτες, ή ανεξέλεγκτες. Η ηγεσία σε αυτές τις ομάδες αυτοοργάνωσης είναι προσαρμοστική, κατευθυντική και παρέχει ανατροφοδότηση ως προς την απόδοση των μελών. Οι ηγέτες των ευέλικτων ομάδων είναι υπεύθυνοι για τον καθορισμό της κατεύθυνσης του έργου σε συνεργασία με τον πελάτη, την ευθυγράμμιση των ατόμων, την απόκτηση πόρων και την παρακίνηση των ατόμων (Hoda & Noble, 2011).

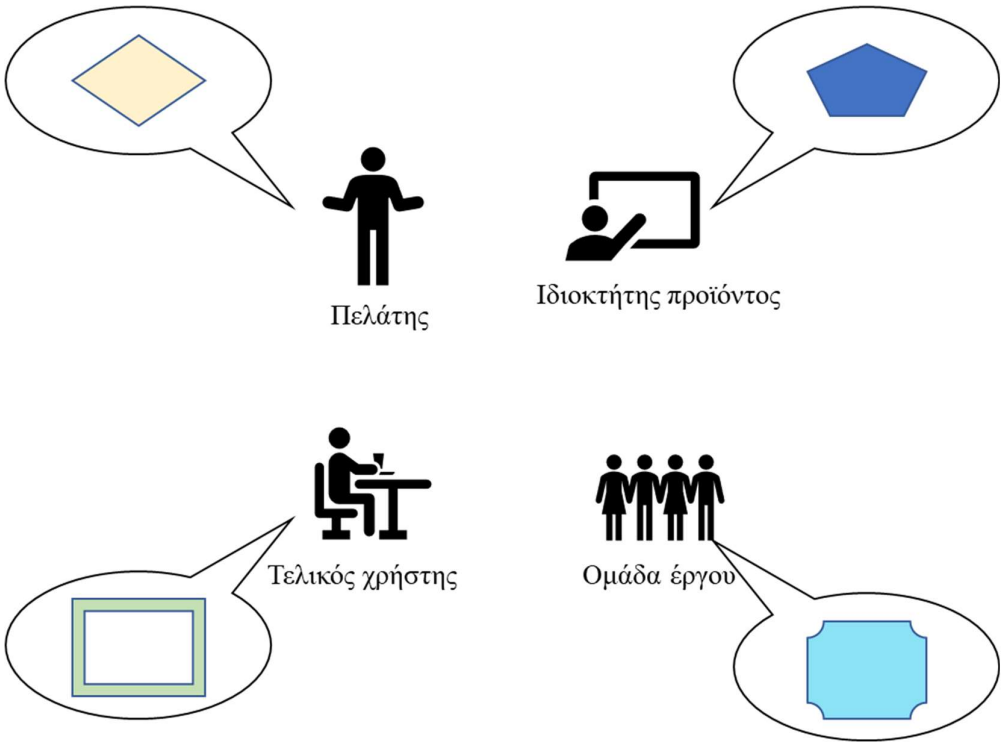
Τα μέλη των αυτοοργανούμενων ομάδων αναλαμβάνουν άτυπους, παροδικούς και αυθόρμητους ρόλους για να ικανοποιήσουν διάφορες οργανωτικές ανάγκες της ομάδας (Hoda et al., 2013) και εκτελούν μια σειρά «πράξεων εξισορρόπησης» μεταξύ ελευθερίας και ευθύνης, διαλειτουργικότητας και εξειδίκευσης, κ.λπ.. με σκοπό την επίτευξη των αρχών και των συνθηκών αυτοοργάνωσης. Οι άτυποι αυτοί ρόλοι που υπάρχουν σε όλες τις ομάδες είναι αυτές που σε μεγάλο βαθμό καθορίζουν την απόδοση της ομάδας (Belbin, 2012).

Οι αυτοοργανωμένες ομάδες θεωρούνται ως ένας από τους κρίσιμους παράγοντες επιτυχίας των ευέλικτων έργων. Η αυτοοργάνωση επηρεάζει άμεσα την αποτελεσματικότητα της ομάδας καθώς και τον τρόπο λήψης αποφάσεων και την ταχύτητα και την ακρίβεια της επίλυσης προβλημάτων (Chow & Cao, 2008).

3.3.2 Η άμεση επικοινωνία και η καθημερινή συνεργασία

Όπως αναφέρθηκε προηγουμένως, ο στόχος μιας ευέλικτης ομάδας είναι να δημιουργήσει ένα προϊόν που παρέχει αξία στον πελάτη. Για να συμβεί αυτό, η ομάδα ανάπτυξης πρέπει να κατανοήσει τι συνιστά «επιχειρηματική αξία» για τον συγκεκριμένο πελάτη. Είναι αναγκαία λοιπόν, η καθημερινή συνεργασία και η άμεση επικοινωνία με τον πελάτη. Με αυτόν τον τρόπο ο πελάτης/ο ιδιοκτήτης του προϊόντος αποτελεί πλήρες και ισότιμο μέλος της ομάδας έργου.

Πολλές φορές η συνεργασία αυτή δεν είναι απρόσκοπτη διότι ο πελάτης έχει διαφορετικούς στόχους ή/και γνωστικό υπόβαθρο σε σχέση με αυτά της ομάδας έργου. Οι τεχνικοί της ομάδας έργου δεν διαθέτουν τις απαραίτητες γνώσεις που αφορούν το πεδίο εφαρμογής (problem domain) ώστε να κατανοήσουν τις απαιτήσεις και να τις μεταφράσουν σε ένα προϊόν, ενώ οι πελάτες πολλές φορές αγνοούν τις τεχνικές προκλήσεις που προκύπτουν σε ένα έργο ανάπτυξης λογισμικού. Επομένως, αυτή η διαφορετική οπτική γωνία ή η διαφορετική κατανόηση του προβλήματος από τους συμμετέχοντες δημιουργεί σημαντικά προβλήματα τα οποία μπορούν να λυθούν μόνο με τη συνεχή και άμεση επικοινωνία. Το πρόβλημα αυτό είναι ιδιαίτερα σημαντικό και παρουσιάζεται γραφικά στην Εικόνα 3.2.



Εικόνα 3.2: *Η διαφορετική οπτική των συμμετεχόντων*

Η άμεση επικοινωνία περιορίζει σημαντικά την ανάγκη τεκμηρίωσης. Σε ένα τυπικό έργο, ένα βασικό ερώτημα που αντιμετωπίζουν οι ομάδες λογισμικού είναι "πόση τεκμηρίωση" είναι αναγκαία. Η παραδοσιακή προσέγγιση σχετικά με την τεκμηρίωση λογισμικού είναι ότι είναι αναγκαίο ένα σύστημα διαχείρισης τεκμηρίωσης όπου συλλέγονται όλες οι πληροφορίες του υπό ανάπτυξη συστήματος, όπου θα είναι διαθέσιμες σε όποιον ή όποτε χρειαστεί. Κάθε φορά που η ομάδα έργου χρειάζεται να αλλάξει, ας πούμε, ένα μέρος του συστήματος, μπορεί να δει σε ποιον ακριβώς κώδικα αναφέρεται η αλλαγή, ποιες απαιτήσεις αφορά, ποια σημεία του συστήματος θα πρέπει να ελεγχθούν ξανά, κ.λπ.

Η άμεση επικοινωνία, και ιδιαίτερα αυτή πρόσωπο με πρόσωπο είναι σχεδόν πάντα πιο αποτελεσματική και για την ανταλλαγή ιδεών μέσα σε μια ομάδα ανάπτυξης λογισμικού σε σχέση πάντα με την γραπτή τεκμηρίωση. Όλοι γνωρίζουν ότι η προσωπική συζήτηση για ένα πρόβλημα είναι ο πιο αποτελεσματικός τρόπος για να κατανοήσουμε το πρόβλημα ή την νέα ιδέα. Στην συνέχεια, αν κρίνουμε απαραίτητο τεκμηριώνουμε το αποτέλεσμα της συζήτησης. Αυτός είναι ο λόγος για τον οποίο οι ευέλικτες πρακτικές επικοινωνίας επικεντρώνονται περισσότερο σε μεμονωμένα άτομα που επικοινωνούν μεταξύ τους και διατηρούν τεκμηρίωση για τις περιπτώσεις όπου πολύπλοκες πληροφορίες πρέπει να ανακληθούν λεπτομερώς αργότερα.

Επιπλέον, η άμεση επικοινωνία εκτός από την προφορική επικοινωνία εμπεριέχει τη γλώσσα του σώματος, τον τόνο της φωνής, τις οπτικές εικόνες που την συμπληρώνουν, πληροφορία που συχνά είναι χρήσιμη και χάνεται με τη γραπτή επικοινωνία. Τέλος, η γραπτή επικοινωνία χρειάζεται περισσότερο χρόνο για να γίνει και ιδιαίτερα όταν εμπεριέχει διάλογο (π.χ. ανταλλαγή emails).

Η συνεχής και συχνή ανατροφοδότηση είναι ιδιαίτερα σημαντική σε ευέλικτα έργα και ενισχύεται σημαντικά με τη συνεργασία πρόσωπο με πρόσωπο. Ακόμη και όταν τα μέλη της ομάδας βρίσκονται σε διαφορετικά γεωγραφικά σημεία, είναι εξίσου ή ακόμη πιο σημαντικό να δημιουργηθεί μια εικονική άμεση συνεργασία μέσω υπηρεσιών τηλεδιάσκεψης.

Το μέγεθος της ομάδας στα περισσότερα ευέλικτα έργα είναι επίσης ένας σημαντικός παράγοντας για καλή επικοινωνία, καθώς οι περισσότεροι συμβουλεύουν ότι η ομάδα έργου δεν θα πρέπει να ξεπερνά τα 9 άτομα. Όσο μεγαλύτερη είναι η ομάδα τόσο πιο δύσκολος γίνεται ο συντονισμός και η επικοινωνία μεταξύ των μελών της. Σύμφωνα με τους Bustamante and Sawhney (2011), η ιδανική ευέλικτη ομάδα είναι μικρή, βρίσκεται στο ίδιο μέρος, επικοινωνεί πρόσωπο με πρόσωπο σε καθημερινή βάση και δεν έχει πάνω από 9 μέλη.

Τέλος, μια άλλη σημαντική πτυχή της επικοινωνίας πρόσωπο με πρόσωπο είναι ότι όταν οι άνθρωποι επικοινωνούν ελεύθερα, ανοιχτά και συχνά, υπάρχει μεγαλύτερη πιθανότητα να αναπτυχθεί εμπιστοσύνη μεταξύ των μελών της ομάδας, να έχουν τα μέλη της ομάδας έργου κοινή κατανόηση καθώς και κοινό σκοπό (Stellman & Greene, 2015; Lalsing, et al., 2012).

3.3.3 Ατομική παρακίνηση

Η σημαντική πίεση για ταχεία και επιτυχημένη υλοποίηση έργων πληροφορικής απαιτεί αποτελεσματική παρακίνηση του προσωπικού του έργου. Είναι σημαντικό να διασφαλίσουμε ότι κάθε μέλος της ομάδας έχει κίνητρα και χρησιμοποιεί τις ικανότητές του προς το συμφέρον της ομάδας ή του οργανισμού για τον οποίο εργάζεται.

Με τη λέξη παρακίνηση (motivation) αναφερόμαστε στη θέληση ενός μέλους της ομάδας έργου ή ενός εργαζομένου γενικότερα να καταβάλλει προσπάθεια για την επίτευξη των στόχων της ομάδας. Στην προσπάθεια περιγραφής της παρακίνησης αναπτύχθηκαν πολλές θεωρίες. Η ικανοποίηση αναγκών αποτελεί το βασικότερο λόγο που εργάζονται οι άνθρωποι και από το βαθμό ικανοποίησης τους εξαρτάται η απόδοση και η ποιότητα της εργασίας τους. Η συμπεριφορά του ανθρώπινου δυναμικού επηρεάζει την στάση, την απόδοση, την αφοσίωση των εργαζομένων και αποτελεί καθοριστικό παράγοντα στην βιωσιμότητα και ανταγωνιστικότητα ενός οργανισμού.

Το πιο γνωστό μοντέλο παρακίνησης είναι αυτό του Abraham Maslow. Σύμφωνα με τον Maslow, μόνον οι ανάγκες που δεν έχουν ικανοποιηθεί είναι παράγοντες παρακίνησης. Αυτό σημαίνει ότι μόνον αν πεινάτε θα αγοράσετε, θα καλλιεργήσετε ή (ανάλογα με την ένταση ή τη διάρκεια της πείνας) μπορεί ακόμα και να κλέψετε τροφή για να ικανοποιήσετε την πρωταρχική βιολογική ανάγκη της επιβίωσης. Επίσης, μόνο αν έχετε έντονο πάθος για επιτυχία, θα μελετήσετε και θα μάθετε όσο περισσότερα μπορείτε για να ικανοποιήσετε αυτήν την φιλοδοξία. Στο μοντέλο τα κίνητρο του ατόμου παρουσιάζονται σε πέντε επίπεδα. Στο χαμηλότερο επίπεδο βρίσκονται οι ανάγκες που σχετίζονται με την επιβίωση του ατόμου, ακολουθούν τα κίνητρα που σχετίζονται με τις ανάγκες ασφάλειας, την ανάγκη του ατόμου να ανήκει σε ένα κοινωνικό σύνολο ή μια ομάδα. Τα δύο υψηλότερα επίπεδα σχετίζονται με κίνητρα που προκύπτουν από την ανάγκη για εκτίμηση,

τόσο από το ίδιο το άτομο προς τον εαυτό του, όσο και από το περιβάλλον του, καθώς και με τις ανάγκες για προσωπική ανάπτυξη (ή αυτοπραγμάτωση).

Υπάρχουν πολλοί παράγοντες που κινητοποιούν τους εργαζόμενους, εσωτερικοί και εξωτερικοί,. Οι εσωτερικοί παράγοντες σχετίζονται με την ανάγκη του ατόμου για αναγνώριση της συνεισφοράς του ενώ αντίθετα οι εξωτερικοί παράγοντες προσφέρονται από τρίτους π.χ. από το περιβάλλον της εργασίας. Με άλλα λόγια, η εξωγενής παρακίνηση σχετίζεται με το εργασιακό περιβάλλον και το αν αυτό το εργασιακό περιβάλλον ικανοποιεί τις ανάγκες του ατόμου π.χ. με τις υλικές ανταμοιβές όπως οικονομικές (μισθός-πριμ), ασφάλεια, συνθήκες εργασίας (Τζωρτζάκης, 2019).

Το θέμα της παρακίνησης έχει μελετηθεί εκτενώς στη βιβλιογραφία. Ειδικότερα, στον τομέα της ανάπτυξης λογισμικού, οι Beecham et al. (2008) περιγράφουν λεπτομερώς έναν ολοκληρωμένο κατάλογο παραγόντων που αυξάνουν (βλέπε Πίνακα 3.1 με τα κίνητρα) ή μειώνουν την παρακίνηση (βλέπε Πίνακα 3.2 με τα αντικίνητρα).

Κίνητρα εργαζομένων στην ανάπτυξη λογισμικού
Μ.1 Ανταμοιβές και κίνητρα (π.χ. αυξημένη αμοιβή και παροχές που σχετίζονται με την απόδοση)
Μ.2 Αναπτυξιακές ανάγκες εργαζομένου (π.χ. ευκαιρίες κατάρτισης για τη διεύρυνση των δεξιοτήτων, ευκαιρίες για εξειδίκευση)
Μ.3 Ποικιλία εργασιών (π.χ. χρήση ποικίλων δεξιοτήτων)
Μ.4 Πορεία καριέρας (ευκαιρία για εξέλιξη, προοπτική προαγωγής, σχεδιασμός σταδιοδρομίας)
Μ.5 Ενδυνάμωση/ευθύνη (όπου η ευθύνη ανατίθεται στο άτομο και όχι στο καθήκον)
Μ.6 Καλή διοίκηση (π.χ. υποστήριξη ανώτερης διοίκησης, δημιουργία ομάδας, καλή επικοινωνία)
Μ.7 Αίσθηση ότι ανήκουν/υποστηρικτικές σχέσεις
Μ.8 Ισορροπία εργασίας/ζωής (ευελιξία στους χρόνους εργασίας, τοποθεσία εργασίας)
Μ. 9 Εργασία σε επιτυχημένη εταιρεία (π.χ. οικονομικά σταθερή)
Μ.10 Συμμετοχή εργαζομένων (π.χ. στη λήψη αποφάσεων) / συνεργασία με άλλους
Μ.11 Ανατροφοδότηση / αξιολόγηση
Μ. 12 Αναγνώριση (για ποιοτική εργασία, καλή δουλειά που γίνεται βάσει αντικειμενικών κριτηρίων)
Μ.13 Παροχή μετοχών / συμμετοχή στην ιδιοκτησία της εταιρείας
Μ.14 Εμπιστοσύνη/σεβασμός
Μ.15 Τεχνικά απαιτητική εργασία
Μ.16 Ασφάλεια εργασίας/σταθερό περιβάλλον
Μ.17 Καλός προσδιορισμός εργασίας (σαφείς στόχοι, προσωπικό ενδιαφέρον, γνώση του σκοπού της εργασίας, πώς ταιριάζει αυτή στη συνολική εικόνα, ικανοποίηση από την εργασία, παραγωγή αναγνωρίσιμου κομματιού ποιοτικής εργασίας)
Μ.18 Αυτονομία (π.χ. ελευθερία στην εκτέλεση καθηκόντων)
Μ. 19 Κατάλληλες συνθήκες εργασίας/περιβάλλον/καλός εξοπλισμός/εργαλεία/φυσικός χώρος/ησυχία
Μ.20 Αντίκτυπο εργασίας (βαθμός στον οποίο η εργασία έχει ουσιαστικό αντίκτυπο στη ζωή ή το έργο άλλων ανθρώπων)
Μ.21 Ύπαρξη επαρκών πόρων

Πίνακας 3.1: *Κίνητρα εργαζομένων στην ανάπτυξη λογισμικού*

Αντικίνητρα εργαζομένων στην ανάπτυξη λογισμικού
D.1 Ύπαρξη κίνδυνων
D.2 Ύπαρξη άγχους
D.3 Ανισότητα (π.χ. αναγνώριση με βάση τη διαίσθηση της διοίκησης ή τις προσωπικές προτιμήσεις)
D.4 Ενδιαφέρουσα εργασία αλλά για άλλα άτομα (π.χ. εξωτερική ανάθεση)
D.5 Αθέμιτο σύστημα ανταμοιβής (π.χ. η διοίκηση ανταμείβεται για την οργανωτική απόδοση, ανταμοιβή με βάση τη θέση και όχι την απόδοση)

Αντικίνητρα εργαζομένων στην ανάπτυξη λογισμικού
D.6 Έλλειψη ευκαιριών προώθησης/στασιμότητα/ βαρετή εργασία/ακαταλληλότητα για συγκεκριμένη εργασία
D.7 Κακή επικοινωνία (ανεπάρκεια ανατροφοδότησης/κακή επικοινωνία με τη διοίκηση)
D.8 Μη ανταγωνιστική αμοιβή/κακή αμοιβή/απλήρωτες υπερωρίες
D.9 Μη ρεαλιστικοί στόχοι/αδύνατες στην υλοποίηση προθεσμίες
D.10 Κακή σχέση με χρήστες ή/και συναδέλφους
D.11 Κακό περιβάλλον εργασίας (π.χ. έλλειψη επενδύσεων και πόρων, εργασία μακριά από την ομάδα)
D.12 Κακή διοίκηση (π.χ. συναντήσεις που είναι χάσιμο χρόνου)
D.13 Παραγωγή λογισμικού κακής ποιότητας (έλλειψη αισθήματος επίτευξης)
D.14 Κακή προσαρμογή στην κουλτούρα του οργανισμού/στερεότυπα/ασάφεια ρόλων
D.15 Έλλειψη επιρροής/μη εμπλοκή στη λήψη αποφάσεων

Πίνακας 3.2: *Αντικίνητρα εργαζομένων στην ανάπτυξη λογισμικού*

Η εφαρμογή ευέλικτης προσέγγισης μπορεί δυνητικά να αυξήσει τα κίνητρα των ατόμων μιας ευέλικτης ομάδας με διάφορους τρόπους. Για παράδειγμα:

- Τα ευέλικτα έργα χωρίζονται σε σύντομες επαναλήψεις και ο στόχος της κάθε επανάληψης μπορεί να είναι εύκολα ορατός από όλους.
- Οι ευέλικτες ομάδες έχουν την αυτονομία να παρακολουθούν και να διαχειρίζονται τον εαυτό τους και πρέπει να τους επιτρέπεται να το κάνουν, με ελάχιστη παρέμβαση από στελέχη της επιχείρησης, αφού οι παρεμβολές αποτελούν αντικίνητρο.
- Τα μέλη της ομάδας μπορούν να παρακινήσουν και να επηρεάσουν τη συμπεριφορά των άλλων μελών μέσω συχνών συναντήσεων και της επικοινωνίας.
- Σε μια ευέλικτη ομάδα, τα μέλη της μπορούν να παρακινηθούν ώστε να αναπτύξουν τις δεξιότητές τους μαθαίνοντας από πιο έμπειρο προσωπικό και ενθαρρύνοντας ο ένας τον άλλον να αναλάβει την ευθύνη για συγκεκριμένες εργασίες.
- Αντίστοιχα μια ευέλικτη ομάδα μπορεί να αποθαρρυνθεί αν υπάρχει:
- Έλλειψη ευκαιριών προαγωγής ή προώθησης της σταδιοδρομίας των μελών της ομάδας.
- Έλλειψη αναγκαίων πόρων.
- Μη ρεαλιστικά χρονοδιαγράμματα
- κ.λπ.

3.3.4 Βιώσιμη ανάπτυξη και σταθερός ρυθμός εργασίας

Μία από τις 12 αρχές της ευέλικτης φιλοσοφίας είναι ότι «Οι ευέλικτες διαδικασίες προάγουν τη βιώσιμη ανάπτυξη (sustainable development). Οι πελάτες, η ομάδα έργου και οι χρήστες θα πρέπει να μπορούν να διατηρούν σταθερό ρυθμό επ' αόριστον». Η εφαρμογή αυτής της αρχής είναι ένας κρίσιμος παράγοντας επιτυχίας για την επιτυχία των έργων που ακολουθούν την ευέλικτη προσέγγιση αλλά και γενικότερα.

Εάν ο ρυθμός της ομάδας δεν είναι βιώσιμος, είναι πιθανό να υπάρξουν αρκετά προβλήματα, όπως (Hartman, 2009):

- Τα σφάλματα στο λογισμικό θα αυξηθούν, αφού η εργασιακή εξουθένωση των μελών της ομάδας οδηγεί σε περισσότερα σφάλματα.
- Η απόδοση στην εργασία θα μειωθεί. Τα μέλη μια εξουθενωμένης ομάδες είναι λιγότερο παραγωγικά και κάνουν την ίδια εργασία σε περισσότερο χρόνο.

- Το ηθικό της ομάδας μειώνεται. Αυτό μπορεί να οδηγήσει σε εγκατάλειψη του έργου, ή στην εύρεση νέας εργασίας από τα μέλη της ομάδας.
- Το κλίμα που θα επικρατήσει μέσα στην ομάδα θα είναι δυσάρεστο και γενικότερα αρνητικό. Ένα πολύ συνηθισμένο φαινόμενο που παρατηρείται είναι η τάση απόδοσης ευθυνών για κάθε τι που δεν γίνεται με το σωστό τρόπο (blame game), αντί για ενθάρρυνση των μελών της ομάδας για να αναλαμβάνουν πρωτοβουλίες και να δοκιμάσουν νέες καινοτόμες προσεγγίσεις.
- Η ομάδα εγκαταλείπει τις καλές πρακτικές σε όφελος εκείνων που δίνουν γρήγορα αποτελέσματα αμφιβόλου ποιότητας και είναι ορατές είτε στον πελάτη είτε στη διοίκηση.

Συνεπώς μια βασική συνέπεια της μη βιώσιμης ανάπτυξης είναι η εργασιακή εξουθένωση (work burnout), ένα φαινόμενο το οποίο έχει μελετηθεί σε βάθος από μεγάλο αριθμό μελετητών. Η εργασιακή εξουθένωση εμφανίζεται ως έντονο επαγγελματικό άγχος μεγάλης διάρκειας στον εργασιακό χώρο, κάτι το οποίο το άτομο αδυνατεί να διαχειριστεί και να αντιμετωπίσει με αποτέλεσμα να αποδυναμώνεται και να καθίσταται ανίκανο να προσαρμοστεί στις απαιτήσεις του εργασιακού του περιβάλλοντος (Κουστέλιος & Κουστέλιου, 2001). Θα λέγαμε ότι η εργασιακή εξουθένωση είναι επακόλουθο της έντονης ψυχολογικής πίεσης που αισθάνεται το άτομο εξαιτίας του εργασιακού άγχους και συνιστά μια κατάσταση με κύρια χαρακτηριστικά τη σωματική, συναισθηματική, πνευματική και ψυχική εξάντληση ως αποτέλεσμα της μεγάλης διάρκειας έκθεσης του σε απαιτητικές συνθήκες εργασίας έντονης συναισθηματικής πίεσης. Είναι, λοιπόν, αντίδραση του ατόμου στο χρόνιο συναισθηματικό άγχος, που εκδηλώνεται ως εξάντληση συναισθηματικής, σωματικής και γνωστικής φύσεως, μείωση της παραγωγικότητας, αποπροσωποποίηση και ανάπτυξη δυσλειτουργικών συμπεριφορών (Perlman & Hartman, 1982).

Η εργασιακή εξουθένωση είναι ένα συχνό φαινόμενο στα έργα ανάπτυξης λογισμικού. Η εφαρμογή των ευέλικτων μεθόδων μειώνει την εργασιακή εξάντληση διότι η χρήση ευέλικτων μεθόδων μειώνει την ασάφεια των ρόλων μέσα στην ομάδα έργου και συνεπώς τις συγκρούσεις που προκύπτουν από αυτή την ασάφεια. Επίσης βοηθά στην καλύτερη κατανόηση της επιτυχίας και της αποτυχίας στα σύγχρονα έργα ανάπτυξης λογισμικού. Επίσης, η κατανομή και ανάθεση εργασιών μέσα σε ευέλικτα έργα γίνεται σταδιακά και οδηγεί σε καλύτερες, ομοιόμορφες κατανομές με αποτέλεσμα να μειώνεται το εργασιακό άγχος και να ελαχιστοποιείται η πιθανότητα εξουθένωσης. Επίσης, η ύπαρξη οργανωτικών δεξιοτήτων είναι ένας σημαντικός παράγοντας που μειώνει το άγχος και την εργασιακή εξουθένωση. Οι δεξιότητες αυτές επιτρέπουν την αποδοτικότερη ομαδική εργασία, την κατανόηση της δυναμικής της ομάδας κ.λπ. Η καθιερωμένη υπόθεση ότι οι προγραμματιστές χρειάζονται αποκλειστικά τεχνικές δεξιότητες και σχετική εμπειρία και όχι οργανωτικές δεξιότητες, δεν έχει εφαρμογή πλέον σε ευέλικτα περιβάλλοντα ανάπτυξης (Venkatesh et al., 2017).

3.4 Η ποιότητα στην ευέλικτη προσέγγιση

Η λέξη ποιότητα έχει πολλές διαφορετικές έννοιες ανάλογα με τις διαφορετικές συνθήκες στις οποίες χρησιμοποιείται. Οι διαφορετικές αυτές έννοιες μπορεί να είναι:

- Η ποιότητα του κατασκευαζόμενου προϊόντος (product quality).
- Η ποιότητα των διεργασιών που χρησιμοποιούνται για την παραγωγή προϊόντων ή την παροχή υπηρεσιών (process quality).
- Η ποιότητα ενός προϊόντος έτσι ώστε αυτό να είναι συμβατό με τις προδιαγραφές των πελατών (conformance quality).
- Η ποιότητα ενός προϊόντος η μιας υπηρεσίας όπως αυτή γίνεται αντιληπτή από τον πελάτη ή τον χρήστη (perceived quality).

Ένας από τους περισσότερο χρησιμοποιούμενους αλλά και αντιπροσωπευτικότερους ορισμούς είναι ο ορισμός που δίνεται από το Διεθνή Οργανισμού Τυποποίησης (International Standardization Organization - ISO) ο οποίος περιλαμβάνεται στο πρότυπο ISO 8402(1987) και αναφέρει: "Ποιότητα είναι το σύνολο των

χαρακτηριστικών ενός προϊόντος ή μίας υπηρεσίας που σχετίζονται με τη δυνατότητά του να ικανοποιεί δεδομένες ή συναγόμενες ανάγκες". Ο ορισμός υπονοεί ότι κάθε εργαζόμενος που ασχολείται με θέματα ποιότητας σε μια επιχείρηση-οργανισμό πρέπει να έχει τη δυνατότητα να αναγνωρίζει τα χαρακτηριστικά και τις ιδιότητες των προϊόντων ή των υπηρεσιών που σχετίζονται άμεσα ή έμμεσα με την ποιότητα.

Η ποιότητα του λογισμικού είναι μία από τις πιο σημαντικές προτεραιότητες στην ανάπτυξη λογισμικού. Οι ευέλικτες μέθοδοι είναι σχεδιασμένες ώστε να παράγουν λογισμικό γρηγορότερα, αλλά πρέπει επίσης να ικανοποιούν τις απαιτήσεις ποιότητας. Η ποιότητα στην ευέλικτη προσέγγιση είναι διάχυτη και σχετίζεται με όλες τις 12 αρχές που υπάρχουν στο μανιφέστο. Το γεγονός ότι το κάθε έργο στοχεύει στη μεγιστοποίηση της επιχειρηματικής αξίας που παραδίδεται στον πελάτη, η λογική των συνεχών επαναλαμβανόμενων παραδόσεων (Continuous Delivery), η καθημερινή συνεργασία με τον πελάτη κ.α. είναι αρχές που αποτελούν εκτός από ευέλικτες αρχές βασικές αρχές ποιότητας. Στις επόμενες παραγράφους θα εστιαστούμε σε ευέλικτες αρχές που δεν έχουν ακόμα αναλυθεί επαρκώς στο παρόν κεφάλαιο.

3.4.1 Τεχνική αρτιότητα

Η διαρκής έμφαση στην τεχνική αρτιότητα και στην εύρυθμη σχεδίαση ή γενικότερα η επιδίωξη της ποιότητας είναι ζωτικής σημασίας για τη δημιουργία επιτυχημένων προϊόντων. Ως εκ τούτου η τεχνική αριστεία βρίσκεται στον πυρήνα της ευέλικτης φιλοσοφίας. Η ενίσχυση της τεχνικής αριστείας ενός οργανισμού είναι το κλειδί για τη διατήρηση ενός υψηλού επιπέδου απόδοσης σε τρέχοντα προγράμματα και έργα καθώς και την προετοιμασία για νέα. Η τεχνική αριστεία είναι ένας διαρκής στόχος όλων των οργανισμών και των ατόμων.

Η τεχνική αριστεία και αρτιότητα επιτυγχάνεται με την εφαρμογή σε συνδυασμό ενός συνόλου μεθοδολογιών, τεχνικών εργαλείων, προτύπων που τελικά δίνουν ένα άρτιο και αξιόπιστο αποτέλεσμα. Για παράδειγμα, επιτυγχάνεται μέσω της εφαρμογής του Test-Driven Development (δηλαδή της συγγραφής κώδικα δοκιμών πριν από τη ανάπτυξη του κώδικα του συστήματος), με αναθεωρήσεις κώδικα (review), με την εφαρμογή του προγραμματισμού σε ζευγάρια (pair programming), με καλό ορισμό των Definition-of-Dones (θα επεξηγηθούν στο κεφάλαιο 7), με την παραγωγή νέων εκδόσεων του λογισμικού σε τακτά χρονικά διαστήματα (επαναληπτική ανάπτυξη – iterative development), με αναδιαμόρφωση του κώδικα (refactoring - βελτίωση κώδικα ακόμη και όταν δεν έχουν αλλάξει χαρακτηριστικά), κ.α.

Σύμφωνα με την μελέτη των Alami et al.(2022), η τεχνική αρτιότητα στην ευέλικτη ανάπτυξη προκύπτει από την κουλτούρα, δηλ. από μια συλλογή κοινών αξιών και πεποιθήσεων. Οι συμμετέχοντες στην μελέτη όρισαν ότι τα βασικά στοιχεία που επηρεάζουν την τεχνική αρτιότητα είναι:

1. Συνεχή προσοχή στον βιώσιμο κώδικα (sustainable code) που προϋποθέτει από τα μέλη της ομάδας μια μακροπρόθεσμή οπτική γωνία, δέσμευση για παραγωγή καθαρού κώδικα (clean code), συνεχή προσοχή στις λεπτομέρειες, και επαγγελματισμό.
2. Συνεχή έμφαση στη μάθηση
3. Συνεχή προσοχή στην ομάδα.
4. Τον επαγγελματισμό της ομάδας ανάπτυξης. Ο επαγγελματισμός επιτρέπει τη δημιουργία υψηλής ποιότητας κώδικα που προσφέρει σημαντική αξία στον πελάτη και μπορεί να κλιμακωθεί με βιώσιμο τρόπο
5. Η γενικότερη οργανωσιακή κουλτούρα. Η επιχείρηση πρέπει να ενθαρρύνει όλες τις προσπάθειες για την βελτίωση τεχνικής αρτιότητας, που σημαίνει περισσότερο χρόνο για πειραματισμό, και προτυποποίηση, αποδοχή των σφαλμάτων, κ.λπ.
6. Οι ευέλικτες ομάδες θα πρέπει να εντοπίζουν συνεχώς πτυχές που μπορούν να βελτιωθούν και να αναπτύσσουν σχέδια δράσης για να κάνουν αυτές τις βελτιώσεις.

Η επιδίωξη της τεχνικής αρτιότητας είναι μια αρετή, η οποία απαιτεί για την εφαρμογή της, την κατάλληλη κουλτούρα, όπως για παράδειγμα η εφαρμογή της αρχής του αναδυόμενου σχεδιασμού

(emergent design), γεγονός που σημαίνει ότι η αρχιτεκτονική του συστήματος δεν καθορίζονται εκ των προτέρων (ή μόνο σε μια βασική μορφή), αλλά προκύπτει κατά τη διάρκεια της ανάπτυξης του συστήματος.

Γενικότερα, δεν υπάρχει ένας τυποποιημένος, καθολικά αποδεκτός ορισμός του όρου της αρχιτεκτονικής. Ο όρος μπορεί να κατανοηθεί καλύτερα παρουσιάζοντας ορισμένα χαρακτηριστικά/ιδιότητες. Πιο συγκεκριμένα, η αρχιτεκτονική είναι (Φιτσιλής, 2018):

- ένα σύνολο αποφάσεων με σκοπό την οργάνωση ενός πληροφοριακού συστήματος,
- η επιλογή των δομικών στοιχείων ενός πληροφοριακού συστήματος καθώς και των διεπαφών τους,
- η συμπεριφορά των δομικών στοιχείων όπως αυτή καθορίζεται από την αλληλεπίδρασή τους,
- η σύνθεση των δομικών στοιχείων σε προοδευτικά μεγαλύτερα υποσυστήματα,
- το αρχιτεκτονικό στυλ που καθοδηγεί την οργάνωση των δομικών στοιχείων (δηλαδή αυτά τα ίδια τα στοιχεία, οι διεπαφές τους, οι συνεργασίες τους και η σύνθεσή τους).

Σε μια ευέλικτη αρχιτεκτονική, σε αντιδιαστολή με μια παραδοσιακή αρχιτεκτονική λογισμικού, αφήνουμε τις αρχιτεκτονικές αποφάσεις να παρθούν την τελευταία δυνατή στιγμή. Οι Waterman, Nobel and Allan (2015) μελέτησαν σε λεπτομέρεια τον όρο ευέλικτη αρχιτεκτονική (agile architecture) και τους παράγοντες που την επηρεάζουν. Προσδιόρισαν έξι παράγοντες που επηρεάζουν την ευέλικτη αρχιτεκτονική, ο συνδυασμός των οποίων οδηγεί στη δημιουργία πέντε ευέλικτων στρατηγικών. Οι έξι αυτοί παράγοντες είναι:

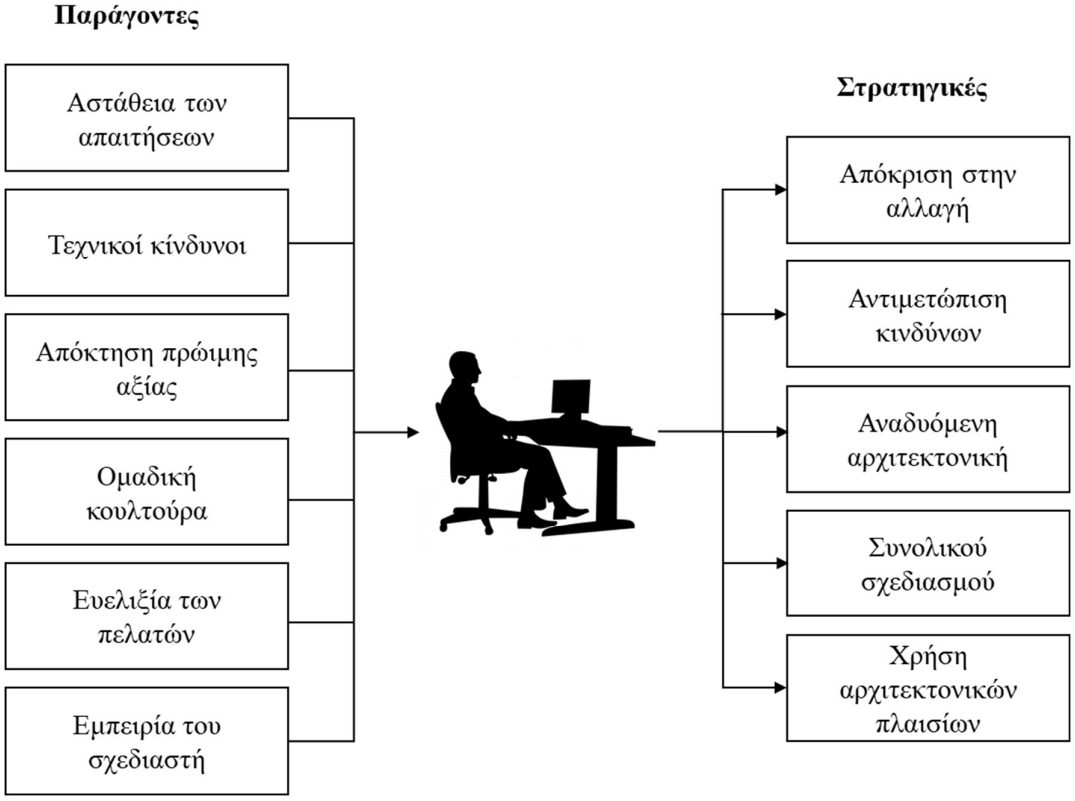
- Η αστάθεια των απαιτήσεων που προκύπτει από τις ελλιπείς απαιτήσεις ή τις μεταβαλλόμενες απαιτήσεις
- Ο τεχνικός κίνδυνος που περιγράφει την επίδραση που έχει η έκθεση σε δυνητικά αρνητικό αποτέλεσμα στην εκ των προτέρων προσπάθεια μιας ομάδας. Ο κίνδυνος από μια πολύπλοκη αρχιτεκτονική αυξάνεται.
- Η απόκτηση πρώιμης επιχειρηματικής αξίας (early value) αναφέρεται στην ανάγκη ενός πελάτη να κερδίσει αξία από ένα σύστημα ή προϊόν που κατασκευάζεται ακόμη (αντί να παρέχει απλώς ανατροφοδότηση) και πριν από την υλοποίηση όλων των λειτουργιών, με τη μορφή ενός ελάχιστου βιώσιμου προϊόντος (Minimum Viable Product – MVP). Οι επιχειρήσεις συχνά απαιτούν τη δημιουργία πρώιμης αξίας, αφού σε ένα δυναμικό εμπορικό περιβάλλον δεν μπορούμε να περιμένουμε την ανάπτυξη ενός πλήρους προϊόντος λογισμικού.
- Η ομαδική κουλτούρα επηρεάζει την ευελιξία της ομάδας και την προσπάθεια που καταβάλλει για τον εκ των προτέρων σχεδιασμό (up-front planning).
- Η ευελιξία των πελατών η οποία θα πρέπει να είναι αντίστοιχη της ομάδας έργου ώστε να υπάρχει συμφωνία στην διαχείριση των απαιτήσεων, στην ανάπτυξη της αρχιτεκτονικής, στην έκταση της τεκμηρίωσης κ.α.
- Η υπάρχουσα εμπειρία του σχεδιαστή που σχετίζεται με τον αντίκτυπο που έχει η άτυπη γνώση, την ικανότητα λήψης αποφάσεων και τον χρόνο που αφιερώνει η ευέλικτη ομάδα στον εκ των προτέρων σχεδιασμό. Έμπειροι σχεδιαστές που έχουν εύρος γνώσεων, είναι πιο πιθανό να γνωρίζουν κατάλληλα αρχιτεκτονικά πρότυπα για την επίλυση προβλημάτων και γενικότερα μπορούν να κατανοήσουν καλύτερα τι θα λειτουργήσει και τι όχι.

Αυτές οι δυνάμεις μπορούν να συνδυαστούν σε έξι διαφορετικές στρατηγικές οι οποίες προσδιορίζουν το επίπεδο του αρχιτεκτονικού σχεδιασμού που θα πρέπει να γίνει αρχικά. Οι στρατηγικές αυτές είναι:

- Η στρατηγική απόκρισης στην αλλαγή στοχεύει στη διαρκή αλλαγή της αρχιτεκτονικής, η οποία είναι κατάλληλη όταν υπάρχει μεγάλη αστάθεια στις απαιτήσεις.

- Η στρατηγική αντιμετώπισης κινδύνων, που είναι κατάλληλη όταν υπάρχουν κίνδυνοι με σημαντικό αντίκτυπο στο έργο.
- Η στρατηγική αναδυόμενης (emergent) αρχιτεκτονικής που παράγει μια αρχιτεκτονική στην οποία η ομάδα λαμβάνει εκ των προτέρων μόνο τις ελάχιστες αρχιτεκτονικές αποφάσεις, όπως η επιλογή της τεχνολογικής στοίβας (technology stack) και του αρχιτεκτονικού στυλ. Σε ορισμένες περιπτώσεις αυτές οι ελάχιστες αποφάσεις είναι σιωπηρές ή έχουν ήδη ληφθεί διότι αποτελούν εξωτερικούς περιορισμούς. Η στρατηγική αυτή είναι κατάλληλη όταν στοχεύουμε στη δημιουργία ενός ελάχιστου βιώσιμου προϊόντος.
- Η στρατηγική συνολικού σχεδιασμού στην αρχή του έργου (aggregate design) απαιτεί την ύπαρξη ενός πλήρους συνόλου απαιτήσεων και την ολοκλήρωση του συνολικού σχεδιασμού της αρχιτεκτονικής πριν την έναρξη της ανάπτυξης. Η αρχιτεκτονική μπορεί να εξελιχθεί κατά την διάρκεια ανάπτυξης του λογισμικού και είναι κατάλληλη να εφαρμοστεί όταν η ευελιξία του πελάτη είναι περιορισμένη.
- Η στρατηγική χρήση αρχιτεκτονικών πλαισίων και προτύπων η οποία εφαρμόζεται συνήθως όταν το περιβάλλον λειτουργίας του συστήματος λογισμικού είναι προκαθορισμένο.

Στην Εικόνα 3.3 παρουσιάζονται οι δυνάμεις σε συνδυασμό με τις στρατηγικές.



Εικόνα 3.3: *Οι παράγοντες που επηρεάζουν την ευέλικτη στρατηγική και οι εναλλακτικές στρατηγικές*

3.4.2 Επιδίωξη της απλότητας

Η κοινότητα της ευέλικτης προσέγγισης έχει ένα ρητό: «Η απλότητα είναι η τέχνη της μεγιστοποίησης της εργασίας που δεν γίνεται». Η ιδέα αυτή είναι κεντρική για την εξάλειψη των περιττών (waste). Συνεπώς, για να κάνουμε τη διαδικασία μας πιο ευέλικτη, πρέπει να κάνουμε λιγότερα. Θα πρέπει να προσέξουμε όμως

ώστε το τελικό αποτέλεσμα να είναι λειτουργικό. Όπως είπε ο Άλμπερτ Αϊνστάιν, «όλα πρέπει να γίνουν όσο το δυνατόν πιο απλά, αλλά ούτε ένα κάτι απλούστερα».

> Everything should be made as simple as possible, but not one bit simpler. (Albert Einstein)

Πολλοί ερευνητές έχουν αναλύσει τον όρο απλότητα και τη συσχέτισή του με την πολυπλοκότητα. Σύμφωνα με τον Apello (2011) η απλότητα συνήθως σχετίζεται με τη δυσκολία που δημιουργείται όταν κάποιος προσπαθεί να εξηγήσει ή να καταλάβει ένα φαινόμενο. Κάτι που είναι εύκολο να κατανοηθεί ή να εξηγηθεί είναι απλό, σε αντίθεση με κάτι περίπλοκο. Επομένως, μια πρώτη διάσταση έχει να κάνει με την ευκολία ή δυσκολία κατανόησης:

Απλό (Simple) = Εύκολο να κατανοηθεί

Περίπλοκο (Complicated) = Δύσκολο να κατανοηθεί

Μια δεύτερη διάσταση σχετίζεται με την συμπεριφορά του λογισμικού ή του συστήματος γενικότερα:

Συστηματική (Ordered) = Πλήρως προβλέψιμη

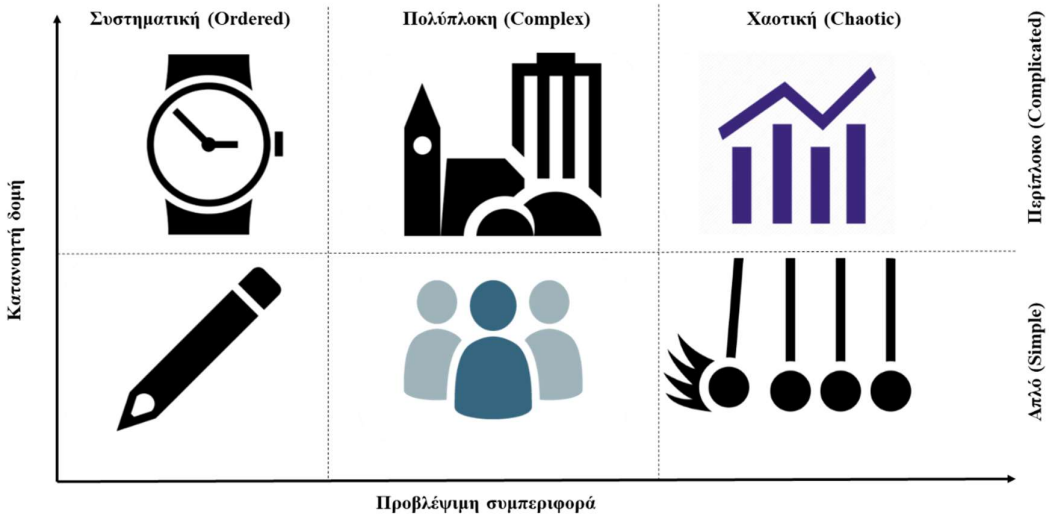
Πολύπλοκη (Complex) = Προβλέψιμη αλλά όχι πλήρως

Χαοτική (Chaotic) = Όχι προβλέψιμη

Για παράδειγμα ένα μολύβι είναι απλό στη λειτουργία του και είναι εύκολο να καταλάβει κανείς πως λειτουργεί. Ένα ρολόι είναι περίπλοκο διότι αν το ανοίγαμε θα έπαιρνε αρκετό χρόνο για να κατανοήσουμε τη λειτουργία του και ίσως όχι πλήρως. Συνεπώς το μολύβι και το ρολόι έχουν συστηματικά προβλέψιμη συμπεριφορά.

Μια ομάδα ανάπτυξης λογισμικού αποτελούμενη από τρία άτομα είναι επίσης απλή στη δομή της, αφού χρειάζονται μόνο μερικές συναντήσεις, ώστε να γνωρίσουμε τα ιδιαίτερα χαρακτηριστικά των μελών της ομάδας. Αντίστοιχα, μια πόλη δεν είναι απλή αλλά περίπλοκη, διότι κάποιος οδηγός ταξί χρειάζεται χρόνια για να γνωρίζει όλους τους δρόμους, τα ξενοδοχεία και τα εστιατόρια. Όμως, τόσο η ομάδα όσο και η πόλη είναι συστήματα πολύπλοκα, διότι ανεξάρτητα από το πόσο καλά τις γνωρίζουμε πάντα υπάρχουν εκπλήξεις. Είναι προβλέψιμα ως ένα βαθμό, αλλά δεν μπορούμε να πούμε με σιγουριά τι θα συμβεί αύριο.

Τέλος, ένα διπλό-πολλαπλό εκκρεμές (δύο ή περισσότερα εκκρεμή συνδεδεμένα μεταξύ τους) είναι επίσης ένα απλό σύστημα. Είναι εύκολο να κατασκευαστεί και να κατανοηθεί. Όμως, υφίσταται απρόβλεπτη χαοτική κίνηση λόγω μεγάλης ευαισθησίας στην αρχική κατάσταση, εκκίνηση, του εκκρεμούς. Τα χρηματιστήρια είναι επίσης χαοτικά συστήματα. Οι κινήσεις των χρηματιστηριακών μετοχών είναι εξ ορισμού απρόβλεπτες, αφού η τιμή της μετοχής εξαρτάται από ένα μεγάλο αριθμό παραγόντων, πολλοί από τους οποίους είναι υποκειμενικοί στην κρίση τους. Συνεπώς, σε αντίθεση με τα εκκρεμή, τα χρηματιστήρια είναι εξαιρετικά περίπλοκα. Αν συνδυάσουμε την δυνατότητα κατανόησης στον ένα άξονα με την συμπεριφορά του συστήματος στον δεύτερο άξονα προκύπτει η Εικόνα 3.4.



Εικόνα 3.4: *Σχέση μεταξύ συμπεριφοράς ομάδων και οργανωτικής δομής*

Άρα, το περίπλοκο αναφέρεται στο ό,τι η κατασκευή ενός συστήματος είναι περίπλοκη για να κατανοηθεί, εκτός και αν κάποιος είναι ειδικός, ενώ το πολύπλοκο και χαοτικό αναφέρεται στη συμπεριφορά του συστήματος, η οποία είναι απρόβλεπτη σε μικρότερο ή μεγαλύτερο βαθμό. Αυτό που είναι περίπλοκο δεν είναι και απαραίτητα πολύπλοκο, καθώς και αντίστροφα. Από την παραπάνω ανάλυση προκύπτουν δύο στρατηγικές, οι οποίες συχνά λανθασμένα ταυτίζονται:

- η απλοποίηση (Simplification) που ως πράξη κάνει τη δομή ενός συστήματος καλύτερα κατανοητή και
- η γραμμικοποίηση (Linearization) που είναι η πράξη που κάνει τη συμπεριφορά του συστήματος πιο προβλέψιμη

Η επίτευξη της απλότητας στην σημερινή ψηφιακή εποχή είναι η λυδία λίθος της σχεδίασης όλων των νέων προϊόντων. Μαθαίνουμε ότι απλότητα σημαίνει λογικότητα ενώ επαναστατούμε ενάντια στην τεχνολογία που είναι υπερβολικά πολύπλοκη. Ο λιτός και ξεκάθαρος σχεδιασμός των προϊόντων μετατρέπει την απλότητα σε εμπορική επιτυχία. Ωστόσο, υπάρχουν φορές που βρισκόμαστε παγιδευμένοι στο παράδοξο της απλότητας: θέλουμε κάτι το οποίο να είναι απλό και εύκολο στη χρήση, αλλά ταυτόχρονα να κάνει όλα τα πολύπλοκα πράγματα που θα μπορούσαμε ποτέ να επιθυμήσουμε. (Maeda, 2006)

Στο βιβλίο του με τίτλο «Οι νόμοι της απλότητας, σχεδιασμός, τεχνολογία, επιχειρήσεις, ζωή», ο John Maeda διάσημος Αμερικανός σχεδιαστής εξερευνά την περιοχή όπου οι επιχειρήσεις, ο σχεδιασμός και η τεχνολογία συγχωνεύονται για να δημιουργήσουν χώρο για τον "ανθρωπιστή τεχνολόγο" και προτείνει δέκα κανόνες για την εξισορρόπηση της απλότητας και της πολυπλοκότητας στους τομείς των επιχειρήσεων, της τεχνολογίας και του σχεδιασμού. Οι δέκα αυτοί κανόνες παρουσιάζονται στον Πίνακα 3.3.

Οι δέκα κανόνες της απλότητας	
Μείωση (Reduce)	Ο πιο συνηθισμένος τρόπος για να επιτευχθεί η απλότητα είναι μέσω της στοχαστικής μείωσης, της ελάττωσης του τελικού προϊόντος. Άλλοι τρόποι μείωσης είναι η σμίκρυνση των διαστάσεων του αντικειμένου, η απόκριση των λειτουργιών ιδιαίτερα όταν αυτές δεν χρησιμοποιούνται συχνά ή η ενσωμάτωση χαρακτηριστικών στη βασική λειτουργία του συστήματος.
Οργάνωση (Organize)	Η οργάνωση ενός συστήματος (π.χ. σε υποσυστήματα, αρθρώματα, αντικείμενα) κάνει ένα σύστημα να φαίνεται λιγότερο πολύπλοκο.
Χρόνος (Time)	Η εξοικονόμηση χρόνου οδηγεί στην απλότητα.

Οι δέκα κανόνες της απλότητας	
Εκμάθηση (Learn)	Η γνώση του αντικειμένου απλοποιεί το πρόβλημα.
Διαφορές (Differences)	Η απλότητα και η πολυπλοκότητα είναι έννοιες αλληλένδετες.
Περιβάλλον (Context)	Αυτό που βρίσκεται στην περιφέρεια/περιβάλλον του προβλήματος σίγουρα δεν είναι δευτερεύουσας σημασίας.
Συναίσθημα (Emotion)	Τα περισσότερα συναισθήματα είναι καλύτερα από λιγότερα.
Εμπιστοσύνη (Trust)	Θα πρέπει να εμπιστευόμαστε την απλότητα
Αποτυχία (Failure)	Θα πρέπει να αποδεχτούμε το γεγονός ότι μερικά πράγματα, δεν μπορούν ποτέ να γίνουν απλά.
Το Ένα (The One)	Η απλότητα αφορά την αφαίρεση του προφανούς και την προσθήκη του ουσιαστικού.
Τρεις βασικές παρατηρήσεις	
Απόσταση (Away)	Η απόσταση από την οποία παρατηρούμε το πρόβλημα έχει να κάνει με το πόσο απλό ή σύνθετο φαίνεται
Ανοικτό (Open)	Η ανοικτή αρχιτεκτονική, δομή, πληροφορία πάντα μειώνει την πολυπλοκότητα
Δύναμη (Power)	Χρησιμοποιώντας λιγότερη δύναμη, πόρους, πίεση πολλές φορές κερδίζουμε περισσότερα.

Πίνακας 3.3: *Οι δέκα κανόνες της απλότητας*

Ο κανόνας της μείωσης του εύρους του συστήματος σχετίζεται με τον κανόνα Pareto, ο οποίος ορίζει την αρχή 80-20. Η αρχή Pareto (γνωστή επίσης ως κανόνας 80/20, ή ως αρχή αποτελεσματικότητας κατά Pareto, ή ως ο νόμος των ζωτικών λίγων) δηλώνει ότι για πολλά γεγονότα περίπου το 80% των επιπτώσεων προέρχεται από το 20% των αιτιών. Ο Juran χρησιμοποίησε τον κανόνα Pareto για την κατηγοριοποίηση των προβλημάτων ποιότητας. Συνήθως, τα κύρια προβλήματα είναι λίγα, παρόλα αυτά οι επιπτώσεις τους είναι μεγάλες, ενώ τα δευτερεύοντα, τα οποία είναι πολλά δημιουργούν λίγες επιπτώσεις. Αντίστοιχα, για το λογισμικό αυτό μεταφράζεται ότι το 80% των χρηστών χρησιμοποιεί μόνο το 20% των λειτουργικών χαρακτηριστικών, το οποίο είναι πολύ σημαντικό όταν θέλουμε να αποφασίσουμε τη βελτιστοποίηση του συστήματος ή στην ιεράρχηση των απαιτήσεων.

Στο σημείο αυτό θα πρέπει να τονίσουμε ξανά ότι η απλότητα είναι μια πολύ βασική αρχή της ευέλικτης προσέγγισης η οποία είναι διάχυτη σε πολλές βασικές ευέλικτες πρακτικές:

- Η ανάπτυξη με βάση τις δοκιμές (Test Driven Development – TDD) ακολουθεί την απλότητα. Γράφουμε μόνο όσο κώδικα είναι αναγκαίος για να εκτελέσουμε επιτυχώς τις δοκιμές.
- Δημιουργία λογισμικού σε βαθμό που να δουλεύει ικανοποιητικά ώστε να λάβουμε σχόλια.
- Γράφουμε μόνο όση τεκμηρίωση είναι αναγκαία.
- Χρήση αποκλειστικά καρτών που τεκμηριώνουν τις ιστορίες των χρηστών (user stories) (Ως πελάτης, θέλω να κάνω κάτι, ώστε να αποκομίσω κάποια αξία). Δεν κάνουμε τίποτα εκτός αν η αξία που κερδίζουμε έχει περιγραφεί με σαφήνεια.
- Χρήση απλών πινάκων εργασιών για τη μέτρηση της προόδου.
- Η μείωση της ιεραρχίας μέσα στον οργανισμό (π.χ. στο Scrum χρησιμοποιούμε μόνο τρεις διακριτούς ρόλους).
- κ.α.

3.4.3 Διαρκής βελτίωση

Η διαρκής βελτίωση είναι μια αρχή η οποία διέπει τόσο την ευέλικτη προσέγγιση, όσο και τις παραδοσιακές μεθόδους διοίκησης. Το θέμα της διαρκούς βελτίωσης (continuous improvement), ή το θέμα της βελτίωσης της ποιότητας (quality improvement) έχει μελετηθεί διεξοδικά από πολλούς ερευνητές. Ο κύκλος διαρκούς βελτίωσης που παρουσιάζεται στην Εικόνα 3.5 παρουσιάζει τα βήματα που πρέπει να ακολουθήσουμε ώστε να έχουμε συνεχή βελτίωση στις διεργασίες μιας επιχείρησης ή ενός οργανισμού. Συνεπώς, θα βρει κανείς στη βιβλιογραφία ένα μεγάλο αριθμό μοντέλων που προσπαθούν να προσδιορίσουν τους τρόπους με τους οποίους επιτυγχάνεται. Ενδεικτικά αναφέρουμε:

- Το μοντέλο PDCA, ή αλλιώς ο κύκλος των Deming/Shewhart που περιλαμβάνει τέσσερα βασικά στάδια Plan-Do-Check-Act και στοχεύουν στη συνεχή βελτίωση. Ο εν λόγω κύκλος σχεδιάστηκε για να επιλύει τα προβλήματα του οργανισμού που σχετίζονται με την ποιότητα και να εφαρμόζονται νέες λύσεις. Πιο συγκεκριμένα:
    - Σχεδίασε (Plan): ορίστε ένα πρόβλημα και υποθέστε τις πιθανές αιτίες και λύσεις του.
    - Κάνε (Do): εφαρμόστε τη λύση στο πρόβλημα.
    - Έλεγξε (Check): αξιολογήστε τα αποτελέσματα.
    - Δράσε (Act): επιστρέψτε στο βήμα του σχεδιασμού εάν τα αποτελέσματα δεν είναι ικανοποιητικά, εάν όμως είναι, τυποποιήστε τη λύση που χρησιμοποιήσατε.
- Τη μεθοδολογία πέντε φάσεων (Define, measure, analyze, improve, and control – DMAIC) που αποτελεί τμήμα της προσέγγισης 6σ προκειμένου να αντιμετωπιστούν τα προβλήματα και να φτάσει η επιχείρηση σε επίπεδα απόδοσης 6σ (https://asq.org/quality-resources/dmaic).
- Το RADAR είναι το ακρώνυμο που χρησιμοποιεί το EFQM (European Foundation for Quality Management) για το διαγνωστικό εργαλείο που έχει αναπτύξει για να βοηθήσει τους οργανισμούς στη βελτίωση του τρόπου εργασίας τους.
- Κ.α.

Εικόνα 3.5: *Ο κύκλος της διαρκούς βελτίωσης*

Γενικότερα, η λιτή διοίκηση (Lean Management) είναι μια ολιστική προσέγγιση που στοχεύει στην παροχή του σωστού προϊόντος, στο σωστό χρόνο, στο σωστό μέρος, με τη σωστή ποιότητα. Η λιτή φιλοσοφία δεν είναι μια συγκεκριμένη διεργασία, αλλά αποτελεί ένα σύνολο αρχών, όπου η εστίαση είναι στη συνεχή βελτίωση των διεργασιών και ιδιαίτερα όσων παράγουν προστιθέμενη αξία για τον πελάτη και την ελαχιστοποίηση όσων εμπεριέχουν σπατάλη (waste). Για παράδειγμα, οι διεργασίες που σχετίζονται με πελάτες είναι διεργασίες που παράγουν αξία, σε αντίθεση με γραφειοκρατικές διεργασίες που εμπεριέχουν σπατάλη.

Ο Shingo (1982) μελέτησε τη σπατάλη στην κατασκευή προϊόντων (manufacturing) και εντόπισε επτά κατηγορίες σπατάλης:

- Το απόθεμα πρώτων υλών ή ημιτελών προϊόντων που βρίσκονται ακόμη σε παραγωγή (in progress inventory).
- Την υπερπαραγωγή (over-production).
- Την περιττή επεξεργασία.
- Την μεταφορά προϊόντων.
- Την περιττή κίνηση είτε προϊόντων από ένα σημείο της επιχείρησης σε ένα άλλο, είτε των εργαζομένων.
- Τον χρόνο αναμονής υλικών ή ημιτελών προϊόντων σε μη παραγωγικά βήματα, καθώς και
- Τα ελαττωματικά προϊόντα.

Αντίστοιχα οι Poppendieck και Poppendieck (2007) και οι Lárusdóttir Cajander, & Simader (2014) παρουσίασαν τις αντίστοιχες κατηγορίες σπατάλης στην ανάπτυξη λογισμικού. Οι κατηγορίες αυτές είναι:

- Μερικώς ολοκληρωμένη εργασία (partially done work). Αναφερόμαστε σε εργασία ανάπτυξης λογισμικού, που δεν μπορεί να ενσωματωθεί ακόμη στο προϊόν, διότι αποτελεί ένα ημιτελές τμήμα. Η μερικώς ολοκληρωμένη εργασία μπορεί να μειωθεί ως ποσοστό, αν οι απαιτήσεις των πελατών (user stories) περιγραφούν με μεγαλύτερη λεπτομέρεια και είναι μικρότερης έκτασης.
- Οι λειτουργίες του λογισμικού που δεν έχουν ξεκάθαρη αξία (value) για τον πελάτη ή οι λειτουργίες που δεν υποστηρίζουν την τρέχουσα εργασία του πελάτη. Τα επιπλέον χαρακτηριστικά του λογισμικού προσιδιάζουν στην υπερπαραγωγή, η οποία σύμφωνα με τον Ohno (2019) αποτελεί τη χειρότερη κατηγορία σπατάλης. Ο κανόνας είναι ότι αν η λειτουργία του λογισμικού δεν μπορεί να συσχετιστεί με μια συγκεκριμένη για τον πελάτη αξία, δεν πρέπει να αναπτυχθεί.
- Η «ανακάλυψη του τροχού» αποτελεί και αυτή μια σημαντική κατηγορία σπατάλης. Ως εκ τούτου, είναι αναγκαίο να αναπτυχθούν οργανωσιακοί μηχανισμοί, οι οποίοι να επιτρέπουν την διατήρηση της γνώσης εντός του οργανισμού, ως μέρος μιας μαθησιακής διαδικασίας. Είναι ζωτικής σημασίας να υπάρχει η δυνατότητα, να χρησιμοποιηθούν οι υπάρχουσες γνώσεις και εμπειρίες από τους εργαζόμενους.
- Η μεταφορά της γνώσης (handoff) από τον έναν εργαζόμενο σε έναν άλλο, π.χ. σε περίπτωση αποχώρησης ή αλλαγής εργασίας του εργαζομένου εντός του οργανισμού αποτελεί πηγή σπατάλης. Επιπλέον, με κάθε μεταφορά χάνεται τμήμα της άτυπης/άρρητης γνώσης (tacit knowledge) διότι είναι γνωστό ότι η άτυπη γνώση είναι δύσκολο να καταγραφεί και να κεφαλοποιηθεί. Οι Poppendieck και Poppendieck (2007) αναφέρουν ότι μόνο το 6% της αρχικής γνώσης απομένει μετά από μια αλυσίδα 4 τέτοιων μεταβάσεων. Ως εκ τούτου, είναι απαραίτητο να μειωθεί η μεταφορά γνώσης μεταξύ εργαζομένων προκειμένου να μειωθεί η σπατάλη.
- Η εναλλαγή μεταξύ εργασιών (task switching) δημιουργεί σπατάλη χρόνου. Θεωρείται ότι η εναλλαγή εργασίας πρέπει να περιοριστεί στο ελάχιστο.

- Οι καθυστερήσεις είναι ένα πολύ συχνό φαινόμενο που συμβαίνει σε πολλές διαφορετικές καταστάσεις. Ένας από τους πιο σημαντικούς τύπους καθυστέρησης είναι η αναμονή ατόμων σε διαφορετικούς χώρους. Συνήθως, οι προγραμματιστές λαμβάνουν μια κρίσιμη απόφαση περίπου κάθε 15 λεπτά. Αυτές οι αποφάσεις μπορούν να ληφθούν άμεσα μόνο εάν υπάρχουν οι απαιτούμενες πληροφορίες στον χώρο που εργάζεται η ομάδα έργου. Αυτό μπορεί να συμβεί μόνο αν η ομάδα έργου αλλά και ο πελάτης βρίσκονται στον ίδιο χώρο.
- Τα σφάλματα του λογισμικού (defects) προκαλούν προβλήματα και συνήθως οδηγούν στη δυσαρέσκεια του πελάτη. Συνεπώς, ο στόχος θα πρέπει να είναι η ανάπτυξη και η παράδοση του λογισμικού με το χαμηλότερο δυνατό ποσοστό σφαλμάτων.

Τα θέματα αυτά θα μελετηθούν εκτενέστερα στο κεφάλαιο 5 όπου παρουσιάζεται η φιλοσοφία της λιτής διοίκησης.

Βιβλιογραφία/Αναφορές

Alami, A., Krancher, O., & Paasivaara, M. (2022). The journey to technical excellence in agile software development. *Information and Software Technology*, 106959.

Apello, J. (2011). *Management 3.0. Leading Agile Developers. Developing Agile Leader*, Pearson Education.

Bakalova, Z. G. (2014). *Towards understanding the value-creation in agile projects*. University of Twente.

Beck K. (2005). *Extreme Programming Explained: Embrace Change* (2nd ed.). Addison Wesley.

Beecham, S., Baddo, N., Hall, T., Robinson, H., and Sharp, H., (2008). Motivation in Software Engineering: A Systematic Literature Review. Information and Software Technology, (50:9-10): 860-878.

Belbin, R. M. (2012). *Team roles at work*. Routledge.

Bustamante, A., & Sawhney, R. (2011). *Agile XXL: Scaling Agile for Project Teams*, Seapine Software.

Chow, T., & Cao, D. B. (2008). A survey study of critical success factors in agile software projects. *Journal of systems and software*, *81*(6), 961-971.

Cockburn, A. (2006). *Agile software development: the cooperative game*. Pearson Education.

Dutka, A. F. (1995). Dutka, A. F. AMA handbook for customer satisfaction. NTC Business books .

Farris, P. W., Bendle, N. T., Pfeifer, E., & David, J. (2010). Marketing Metrics: The Definitive Guide to Measuring Marketing Performance. Upper Saddle River, New Jersey: Pearson Education, Inc. ISBN 0-13-705829-2.

Hartman, Bob. (2009, July 24). New to Agile? Work at a sustainable pace. https://agileforall.com/new-to-agile-work-at-a-sustainable-pace/.

Hoda, R., & Murugesan, L. K. (2016). Multi-level agile project management challenges: A self-organizing team perspective. *Journal of Systems and Software*, *117*, 245-257.

Hoda, R., Noble, J., & Marshall, S. (2012). Developing a grounded theory to explain the practices of self-organizing Agile teams. *Empirical Software Engineering*, *17*(6), 609-639.

Hoda, R., Noble, J., & Marshall, S. (2012). Self-organizing roles on agile software development teams. *IEEE Transactions on Software Engineering*, *39*(3), 422-444.

Jones, T. O., & Sasser, W. E. (1995). Why satisfied customers defect. Harvard business review, 73(6), 88.Juran, J., & Gryna, F. (1993). Quality Planning and Analysis. New York: Mc Graw-Hill.

Karna, S., & Junnonen, J. M. (2017). Designers' Performance Evaluation in Construction Projects. Engineering, Construction, Architectural Management, Vol. 24, No.1 , pp. 154-169.

Kotler, P., Armstrong, G., Saunders, J., & Wong, V. (2001). "Οι Αρχές του Μάρκετινγκ". Αθήνα: Εκδόσεις Κλειδάριθμος.

Kotler, P., Keller, K., Sivaramakrushnan, S., & Cunningham, P. (2013). Marketing Management 14th (Pearson Education ed.). Canada: Canadian ed.

Lalsing, V., Kishnah, S., & Pudaruth, S. (2012). People factors in agile software development and project management. *International Journal of Software Engineering & Applications*, *3*(1), 117.

Lárusdóttir, M. K., Cajander, Å., & Simader, M. (2014, September). Continuous improvement in agile development practice. In *International Conference on Human-Centred Software Engineering* (pp. 57-72). Springer, Berlin, Heidelberg.

Maeda, J. (2006). *The laws of simplicity*. MIT press.

Meier, A., Kropp, M., Anslow, C., & Biddle, R. (2018, May). Stress in agile software development: practices and outcomes. In *International Conference on Agile Software Development* (pp. 259-266). Springer, Cham.

Morgan, G. (1998). Images of organization: The executive edition. *Thousand Oaks, CA*.

Ohno, T., & Bodek, N. (2019). *Toyota production system: beyond large-scale production*. Productivity press.

Perlman, B., & Hartman, E. A. (1982). Burnout: Summary and future research. *Human relations*, *35*(4), 283-305.

Poppendieck, M., & Poppendieck, T. D. (2007). *Implementing lean software development: from concept to cash*. Pearson Education.

Project Management Institute. (2013). Managing change in organizations: A practice guide. Project Management Institute.

Racheva, Z., Daneva, M., & Sikkel, K. (2009, June). Value creation by agile projects: Methodology or mystery?. In *International Conference on Product-Focused Software Process Improvement* (pp. 141-155). Springer, Berlin, Heidelberg.

Shenhar, A., Dvir, D., Levy, O., & Maltz, A. (2001). Project success: a multidimensional strategic concept. Long range planning, Vol.34, No.6 , pp. 699-725.

Shingo, S. (1982). *Study of'Toyota'Production System from Industrial Viewpoint*. Japan Management Association.

Stellman, A., & Greene, J. (2014). *Learning agile: Understanding scrum, XP, lean, and Kanban*. " O'Reilly Media, Inc.".

Trist, E. L. (1981). *The evolution of socio-technical systems* (Vol. 2). Toronto: Ontario Quality of Working Life Centre.

Venkatesh, V., Thong, J. Y., Chan, F. K., Hoehle, H., & Spohrer, K. (2020). How agile software development methods reduce work exhaustion: Insights on role perceptions and organizational skills. *Information Systems Journal*, *30*(4), 733-761.

Waterman, M., Noble, J., & Allan, G. (2015, May). How much up-front? A grounded theory of agile architecture. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (Vol. 1, pp. 347-357). IEEE.

Δερβιτσιώτης, Κ. (1993). Διοίκηση Ολικής Ποιότητας. Αθήνα.

Κουστέλιος, Α. & Κουστέλιου, Ι. (2001). Η επαγγελματική ικανοποίηση και επαγγελματική εξουθένωση στην εκπαίδευση. Ψυχολογία, 8(1), 30-39.

Σαρμανιώτη, Σ. (2020). *Μέθοδοι Μέτρησης Ικανοποίησης Πελατών σε Έργα Πληροφοριακών Συστημάτων*, Μεταπτυχιακή Εργασία, Πανεπιστήμιο Θεσσαλίας.

Τζωρτζάκης Κ. (2019). *Οργάνωση και Διοίκηση*, 5η έκδοση, Εκδοτικός Οίκος Rosili.

Τσιότρας Γ. (2016). *Διοίκηση Ολικής Ποιότητας*. Broken Hill Publishers

Φιτσιλής, Π. (2018). Σύγχρονα πληροφοριακά συστήματα επιχειρήσεων (2η έκδοση). Broken Hill Publishers

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Ποιοι είναι οι λόγοι που οδηγούν τις επιχειρήσεις στην μέτρηση της ικανοποίησης των πελατών;

Απάντηση/Λύση

Οι πιο σημαντικοί λόγοι που μια επιχείρηση καταφεύγει στη μέτρηση της ικανοποίησης των πελατών της, είναι οι εξής (Dutka, 1995):

- Η ικανοποίηση του πελάτη, επειδή αποτελεί την πιο αντικειμενική πληροφορία που προέρχεται από την αγορά, και δίνει τη δυνατότητα στην επιχείρηση να εκτιμήσει την τρέχουσα κατάσταση της αγοράς και να διαμορφώσει ανάλογα την μελλοντική της πολιτική.
- Η ικανοποίηση των πελατών και τα συμπεράσματα που προκύπτουν από τις σχετικές μελέτες επιτρέπουν τον προσδιορισμό των προτεραιοτήτων της επιχείρησης, και δίνουν τη δυνατότητα στην επιχείρηση να προβεί σε διορθωτικές ενέργειες όπου κρίνεται αναγκαίο.
- Η μέτρηση της ικανοποίησης των πελατών επιτρέπει τον προσδιορισμό «ευκαιριών» στη συγκεκριμένη αγορά.
- Η εφαρμογή των αρχών της συνεχούς βελτίωσης που επιβάλλονται από πρότυπα και πολιτικές ποιότητας, απαιτεί την ύπαρξη συγκεκριμένης διαδικασίας μέτρησης της ικανοποίησης των πελατών. Με αυτό τον τρόπο οι ενέργειες βελτίωσης βασίζονται σε πραγματικά δεδομένα, που είναι σύμφωνα με τις ανάγκες και τις επιθυμίες των πελατών.
- Η μέτρηση της ικανοποίησης μπορεί να βοηθήσει στην κατανόηση των γενικότερων αντιλήψεων του πελάτη και πιο συγκεκριμένα στον προσδιορισμό και την ανάλυση των αναγκών, των προσδοκιών και των επιθυμιών του.
- Το πρόβλημα της ύπαρξης διαφορετικής αντίληψης της ικανοποίησης ανάμεσα στον πελάτη και τη διοίκηση της εταιρείας μπορεί να προσδιοριστεί από την υλοποίηση ενός προγράμματος μέτρησης της ικανοποίησης. Με αυτόν τον τρόπο δίνεται η δυνατότητα να αμβλυνθούν αυτές οι διαφορές αντίληψης.

Κριτήριο αξιολόγησης 2

Ποιές είναι οι βασικές παράμετροι στην ικανοποίηση του πελάτη;

Απάντηση/Λύση

Οι Jones and Sasser (1995) προσδιορίζουν τα τέσσερα βασικά στοιχεία που έχουν επιπτώσεις στην ικανοποίηση του πελάτη, τα οποία είναι:

- Τα βασικά χαρακτηριστικά του προϊόντος ή της υπηρεσίας
- Οι βασικές υπηρεσίες υποστήριξης
- Οι διαδικασίες ανάκαμψης στην κακή εμπειρία που τυχόν να έχει ο πελάτης
- Η παροχή άριστης υπηρεσίας.

Κριτήριο αξιολόγησης 3

Ποιες είναι οι τεχνικές που βοηθούν στην ικανοποίηση των πελατών;

Απάντηση/Λύση

Οι βασικότερες τεχνικές που χρησιμοποιούνται είναι οι ακόλουθες:

- Σε όλες τις ευέλικτες ομάδες έργου θα πρέπει να υπάρχει ένας ιδιοκτήτης προϊόντος (product owner). Ο βασικός του ρόλος είναι να διασφαλίσει ότι οι απαιτήσεις είναι κατανοητές από την ομάδα ανάπτυξης του έργου.
- Ο ιδιοκτήτης του προϊόντος κάνει ιεράρχηση των απαιτήσεων είτε βάση της αξίας που έχουν αυτές για την επιχείρηση είτε με άλλα κριτήρια (π.χ. κίνδυνος)
- Ο ιδιοκτήτης του προϊόντος έχει συνεχή επικοινωνία με την ομάδα ανάπτυξης, σε καθημερινή βάση, ώστε να επεξηγήσει λειτουργίες, να διευκρινίσει χαρακτηριστικά του συστήματος, να αποσαφηνίζει απαιτήσεις και να κάνει προτεραιοποίηση αυτών, να λάβει αποφάσεις, να δώσει ανατροφοδότηση και να απαντήσει γρήγορα στις πολλές ερωτήσεις που προκύπτουν κατά τη διάρκεια ενός έργου.
- Η συχνή παράδοση λειτουργικών απαιτήσεων επιτρέπει στον ιδιοκτήτη του προϊόντος και στον πελάτη να έχουν πλήρη εικόνα του τρόπου ανάπτυξης του προϊόντος.
- Καθώς η ομάδα ανάπτυξης συνεχίζει να παρέχει σε τακτά χρονικά διαστήματα πλήρεις, λειτουργικές, δυνητικά αποσπώμενες λειτουργίες, η αξία για την επιχείρηση του συνολικού προϊόντος αυξάνεται σταδιακά, όπως και οι λειτουργικές του δυνατότητες.
- Αντίστοιχα, ο πελάτης συσσωρεύει αξία λαμβάνοντας νέες, έτοιμες προς χρήση λειτουργίες καθ' όλη τη διάρκεια του έργου, αντί να περιμένει μέχρι το τέλος του έργου για την παράδοση του προϊόντος.

Κριτήριο αξιολόγησης 4

Ορίστε την έννοια της συνεχούς παράδοσης (continuous delivery)

Απάντηση/Λύση

Η συνεχής παράδοση είναι η δυνατότητα να βάζουμε σε λειτουργία νέες εκδόσεις του λογισμικού καθημερινά ή ακόμη και πολλές φορές την ίδια ημέρα. Οι εκδόσεις αυτές περιλαμβάνουν αλλαγές όλων των τύπων - συμπεριλαμβανομένων νέων λειτουργιών, αλλαγών διαμόρφωσης, επιδιόρθωση σφαλμάτων κ.λπ.

Κριτήριο αξιολόγησης 5

Ποιες είναι οι αρχές της αυτο-οργάνωσης μιας ομάδας;

Απάντηση/Λύση

Οι αρχές της αυτο-οργάνωσης έχουν περιγραφεί στη βιβλιογραφία ως ομάδες που λειτουργούν (Morgan, 1998; Hoda et al., 2016):

- έχοντας ελάχιστη καθοδήγηση από την ανώτερη διοίκηση ή ανυπαρξία προδιαγραφών για την εκτέλεση των διεργασιών σε αντίθεση με τα διεργασιοκεντρικά (process-oriented) συστήματα ποιότητας που περιγράφουν με λεπτομέρεια το κάθε βήμα,
- οι καθημερινές αποφάσεις λαμβάνονται αποκλειστικά από την ομάδα,

- υπάρχει επαρκής ποικιλία στις διαθέσιμες δεξιότητες εντός της ομάδας, ώστε η ομάδα να μπορεί να καλύψει μια ποικιλία επιχειρηματικών απαιτήσεων,
- υπάρχει πλεονασμός λειτουργιών (redundancy) ως προς την ικανότητα της ομάδας να συμπληρώνει ή και να αντικαθιστά μέλη της σε συγκεκριμένες λειτουργίες όταν αυτό απαιτείται, και
- η ομάδα έχει την ικανότητα να μαθαίνει νέες δεξιότητες αλλά και να ανακαλύπτει νέους καλύτερους τρόπους για την εκτέλεση δραστηριοτήτων.

Κριτήριο αξιολόγησης 6

Ποια είναι τα α χαρακτηριστικά της «ομάδας» σύμφωνα με την ευέλικτη φιλοσοφία;

Απάντηση/Λύση

Μερικά από τα χαρακτηριστικά της «ομάδας» με την ευέλικτη έννοια:

- είναι μια **μικρή ομάδα ανθρώπων**, στην οποία έχει ανατεθεί στο ίδιο έργο ή δραστηριότητα, και σχεδόν όλα τα μέλη της ομάδας έχουν πλήρη απασχόληση εντός της ομάδας. Ένα μικρό τμήμα των μελών της ομάδας μπορεί να είναι μερικής απασχόλησης,
- έχει **κοινή ευθύνη** αφού τα αποτελέσματα, είτε θετικά είτε αρνητικά θα αποδοθούν σε ολόκληρη την ομάδα και όχι σε κάποιο συγκεκριμένο μέλος της ομάδας,
- αναμένεται να **διαθέτει όλες τις απαραίτητες ικανότητες και γνώσεις** (cross functional team), είτε πρόκειται για τεχνικές γνώσεις (προγραμματισμός, σχεδιασμός, έλεγχος), είτε γνώσεις διοίκησης (ικανότητα λήψης αποφάσεων), είτε γνώση του επιχειρηματικού περιβάλλοντος.
- έχει τη **δυνατότητα να λαμβάνει αποφάσεις** (empowered) συνήθως με ομοφωνία των μελών (consensus) και

είναι **αυτοδιοικούμενη** (self-directed) και **αυτοοργανωμένη** (self-managed).

Κριτήριο αξιολόγησης 7

Παρουσιάστε μερικά από τα κίνητρα των εργαζομένων για την ανάπτυξη λογισμικού;

Απάντηση/Λύση

Τα κίνητρα των εργαζομένων στην ανάπτυξη λογισμικού είναι:

- Ανταμοιβές και κίνητρα (π.χ. αυξημένη αμοιβή και παροχές που σχετίζονται με την απόδοση)
- Αναπτυξιακές ανάγκες εργαζομένου (π.χ. ευκαιρίες κατάρτισης για τη διεύρυνση των δεξιοτήτων, ευκαιρίες για εξειδίκευση)
- Ποικιλία εργασιών (π.χ. χρήση ποικίλων των δεξιοτήτων)
- Πορεία καριέρας (ευκαιρία για εξέλιξη, προοπτική προαγωγής, σχεδιασμός σταδιοδρομίας)
- Ενδυνάμωση/ευθύνη (όπου η ευθύνη ανατίθεται στο άτομο και όχι στο καθήκον)
- Καλή διοίκηση (π.χ. υποστήριξη ανώτερης διοίκησης, δημιουργία ομάδας, καλή επικοινωνία)
- Αίσθηση ότι ανήκουν/υποστηρικτικές σχέσεις
- Κ.λπ.

Κριτήριο αξιολόγησης 8

Παρουσιάστε μερικά από τα αντικίνητρα των εργαζομένων για την ανάπτυξη λογισμικού;

Απάντηση/Λύση

Μερικά από τα αντικίνητρα των εργαζομένων στην ανάπτυξη λογισμικού είναι:

- Ύπαρξη κίνδυνων
- Ύπαρξη άγχους
- Ανισότητα (π.χ. αναγνώριση με βάση τη διαίσθηση της διοίκησης ή τις προσωπικές προτιμήσεις)
- Ενδιαφέρουσα εργασία αλλά για άλλα άτομα (π.χ. εξωτερική ανάθεση)
- Αθέμιτο σύστημα ανταμοιβής (π.χ. η διοίκηση ανταμείβεται για την οργανωτική απόδοση, ανταμοιβή με βάση τη θέση και όχι την απόδοση)
- Έλλειψη ευκαιριών προώθησης/στασιμότητα/ βαρετή εργασία/ακαταλληλότητα για συγκεκριμένη εργασία
- Κακή επικοινωνία (ανεπάρκεια ανατροφοδότησης/κακή επικοινωνία με τη διοίκηση)
- Μη ανταγωνιστική αμοιβή/κακή αμοιβή/απλήρωτες υπερωρίες
- Μη ρεαλιστικοί στόχοι/αδύνατες στην υλοποίηση προθεσμίες
- Κακή σχέση με χρήστες ή/και συναδέλφους
- Κ.λπ.

Κριτήριο αξιολόγησης 9

Ποιες είναι οι επιπτώσεις ενός μη-βιώσιμου ρυθμού εργασίας;

Απάντηση/Λύση

Εάν ο ρυθμός της ομάδας δεν είναι βιώσιμος, είναι πιθανό να υπάρξουν αρκετά προβλήματα, όπως (Hartman, 2009):

- Τα σφάλματα στο λογισμικό θα αυξηθούν, αφού η εργασιακή εξουθένωση των μελών της ομάδας οδηγεί σε περισσότερα σφάλματα.
- Η απόδοση στην εργασία θα μειωθεί. Τα μέλη μιας εξουθενωμένης ομάδας είναι λιγότερο παραγωγικά και κάνουν την ίδια εργασία σε περισσότερο χρόνο.
- Το ηθικό της ομάδας μειώνεται. Αυτό μπορεί να οδηγήσει σε εγκατάλειψη του έργου, ή στην εύρεση νέας εργασίας από τα μέλη της ομάδας.
- Το κλίμα που θα επικρατήσει μέσα στην ομάδα θα είναι δυσάρεστο και γενικότερα αρνητικό. Ένα πολύ συνηθισμένο φαινόμενο που παρατηρείται είναι η τάση απόδοσης ευθυνών για κάθε τι που δεν γίνεται με το σωστό τρόπο (blame game), αντί για ενθάρρυνση των μελών της ομάδας για να αναλαμβάνουν πρωτοβουλίες και να δοκιμάσουν νέες καινοτόμες προσεγγίσεις.
- Η ομάδα εγκαταλείπει τις καλές πρακτικές σε όφελος εκείνων που δίνουν γρήγορα αποτελέσματα αμφιβόλου ποιότητας και είναι ορατές είτε στον πελάτη είτε στη διοίκηση.

Κριτήριο αξιολόγησης 10

Ορίστε την έννοια της εργασιακής εξουθένωσης;

Απάντηση/Λύση

Η επαγγελματική εξουθένωση εμφανίζεται ως έντονο επαγγελματικό άγχος μεγάλης διάρκειας στον εργασιακό χώρο, το οποίο το άτομο αδυνατεί να διαχειριστεί και να αντιμετωπίσει με αποτέλεσμα να αποδυναμώνεται και να καθίσταται ανίκανο να προσαρμοστεί στις απαιτήσεις του εργασιακού του περιβάλλοντος (Κουστέλιος & Κουστέλιου, 2001).

Θα λέγαμε ότι η επαγγελματική εξουθένωση είναι επακόλουθο της έντονης ψυχολογικής πίεσης που αισθάνεται το άτομο εξαιτίας του εργασιακού άγχους και συνιστά μια κατάσταση με κύρια χαρακτηριστικά τη σωματική, συναισθηματική, πνευματική και ψυχική εξάντληση ως αποτέλεσμα της μεγάλης διάρκειας έκθεσης του σε απαιτητικές συνθήκες εργασίας έντονης συναισθηματικής πίεσης. Είναι λοιπόν, αντίδραση του ατόμου στο χρόνιο συναισθηματικό άγχος, που εκδηλώνεται ως εξάντληση συναισθηματικής, σωματικής και γνωστικής φύσεως, μείωση της παραγωγικότητας, αποπροσωποποίηση και ανάπτυξη δυσλειτουργικών συμπεριφορών (Perlman & Hartman, 1982).

Κριτήριο αξιολόγησης 11

Παρουσιάστε τις διαφορετικές οπτικές γωνίες της ποιότητας;

Απάντηση/Λύση

Οι διαφορετικές αυτές έννοιες της ποιότητας μπορεί να είναι:

- Η ποιότητα του κατασκευαζόμενου προϊόντος (product quality).
- Η ποιότητα των διεργασιών που χρησιμοποιούνται για την παραγωγή προϊόντων ή την παροχή υπηρεσιών (process quality).
- Η ποιότητα ενός προϊόντος έτσι ώστε αυτό να είναι συμβατό με τις προδιαγραφές των πελατών (conformance quality).
- Η ποιότητα ενός προϊόντος η μιας υπηρεσίας όπως αυτή γίνεται αντιληπτή από τον πελάτη ή τον χρήστη (perceived quality).

Κριτήριο αξιολόγησης 12

Παρουσιάστε τους παράγοντες που επηρεάζουν την ευέλικτη αρχιτεκτονική;

Απάντηση/Λύση

Οι έξι παράγοντες που επηρεάζουν την ευέλικτη αρχιτεκτονική είναι:

- Η αστάθεια των απαιτήσεων που προκύπτει από τις ελλιπείς απαιτήσεις ή τις μεταβαλλόμενες απαιτήσεις
- Ο τεχνικός κίνδυνος που περιγράφει την επίδραση που έχει η έκθεση σε δυνητικά αρνητικό αποτέλεσμα στην εκ των προτέρων προσπάθεια μιας ομάδας. Ο κίνδυνος από μια πολύπλοκη αρχιτεκτονική αυξάνεται.

- Η απόκτηση πρώιμης επιχειρηματικής αξίας (early value) αναφέρεται στην ανάγκη ενός πελάτη να κερδίσει αξία από ένα σύστημα ή προϊόν που κατασκευάζεται ακόμη (αντί να παρέχει απλώς ανατροφοδότηση) και πριν από την υλοποίηση όλων των λειτουργιών, με τη μορφή ενός ελάχιστου βιώσιμου προϊόντος (Minimum Viable Product – MVP). Οι επιχειρήσεις συχνά απαιτούν τη δημιουργία πρώιμης αξίας, αφού σε ένα δυναμικό εμπορικό περιβάλλον δεν μπορούμε να περιμένουμε την ανάπτυξη ενός πλήρους προϊόντος λογισμικού.
- Η ομαδική κουλτούρα επηρεάζει την ευελιξία της ομάδας και την προσπάθεια που καταβάλλει για τον εκ των προτέρων σχεδιασμό (up-front planning).
- Η ευελιξία των πελατών η οποία θα πρέπει να είναι αντίστοιχη της ομάδας έργου ώστε να υπάρχει συμφωνία στην διαχείρισης των απαιτήσεων, στην ανάπτυξη της αρχιτεκτονικής, στην έκταση της τεκμηρίωσης κ.α.
- Η υπάρχουσα εμπειρία του σχεδιαστή που σχετίζεται με τον αντίκτυπο που έχει η άτυπη γνώση, την ικανότητα λήψης αποφάσεων και τον χρόνο που αφιερώνει η ευέλικτη ομάδα στον εκ των προτέρων σχεδιασμό. Έμπειροι σχεδιαστές που έχουν εύρος γνώσεων, είναι πιο πιθανό να γνωρίζουν κατάλληλα αρχιτεκτονικά πρότυπα για την επίλυση προβλημάτων και γενικότερα μπορούν να κατανοήσουν καλύτερα τι θα λειτουργήσει και τι όχι.

Κριτήριο αξιολόγησης 13

Ποια είναι η διαφορά του απλού από το περίπλοκο;

Απάντηση/Λύση

Απλό (Simple) = Εύκολο να κατανοηθεί

Περίπλοκο (Complicated) = Δύσκολο να κατανοηθεί

Κριτήριο αξιολόγησης 14

Αναφέρετε μερικούς από τους κανόνες που οδηγούν στην απλότητα;

Απάντηση/Λύση

Μερικοί από τους κανόνες που οδηγούν στην απλότητα είναι:

- Μείωση (Reduce). Ο πιο συνηθισμένος τρόπος για να επιτευχθεί η απλότητα είναι μέσω της στοχαστικής μείωσης, της ελάττωσης του τελικού προϊόντος. Άλλοι τρόποι μείωσης είναι η σμίκρυνση των διαστάσεων του αντικειμένου, η απόκριση των λειτουργιών ιδιαίτερα όταν αυτές δεν χρησιμοποιούνται συχνά ή η ενσωμάτωση χαρακτηριστικών στη βασική λειτουργία του συστήματος.
- Οργάνωση (Organize). Η οργάνωση ενός συστήματος (π.χ. σε υποσυστήματα, αρθρώματα, αντικείμενα) κάνει ένα σύστημα να φαίνεται λιγότερο πολύπλοκο.
- Χρόνος (Time). Η εξοικονόμηση χρόνου οδηγεί στην απλότητα.
- Εκμάθηση (Learn). Η γνώση του αντικειμένου απλοποιεί το πρόβλημα.
- Κ.λπ.

Κριτήριο αξιολόγησης 15

Ποιες είναι οι κατηγορίες σπατάλης στην ανάπτυξη λογισμικού;

Απάντηση/Λύση

Οι κατηγορίες σπατάλης στην ανάπτυξη λογισμικού είναι:

- Μερικώς ολοκληρωμένη εργασία (partially done work). Αναφερόμαστε σε εργασία ανάπτυξης λογισμικού, που δεν μπορεί να ενσωματωθεί ακόμη στο προϊόν, διότι αποτελεί ένα ημιτελές τμήμα.
- Οι λειτουργίες του λογισμικού που δεν έχουν ξεκάθαρη αξία (value) για τον πελάτη ή οι λειτουργίες που δεν υποστηρίζουν την τρέχουσα εργασία του πελάτη. Τα επιπλέον χαρακτηριστικά του λογισμικού προσιδιάζουν στην υπερπαραγωγή, η οποία σύμφωνα με τον Ohno (2019) αποτελεί τη χειρότερη κατηγορία σπατάλης. Ο κανόνας είναι ότι αν η λειτουργία του λογισμικού δεν μπορεί να συσχετιστεί με μια συγκεκριμένη για τον πελάτη αξία, δεν πρέπει να αναπτυχθεί.
- Η «ανακάλυψη του τροχού» αποτελεί και αυτή μια σημαντική κατηγορία σπατάλης. Ως εκ τούτου, είναι αναγκαίο να αναπτυχθούν οργανωσιακοί μηχανισμοί, οι οποίοι να επιτρέπουν την διατήρηση της γνώσης εντός του οργανισμού, ως μέρος μιας μαθησιακής διαδικασίας. Είναι ζωτικής σημασίας να υπάρχει η δυνατότητα, να χρησιμοποιηθούν οι υπάρχουσες γνώσεις και εμπειρίες από τους εργαζόμενους.
- Η μεταφορά της γνώσης (handoff) από τον έναν εργαζόμενο σε έναν άλλο, π.χ. σε περίπτωση αποχώρησης ή αλλαγής εργασίας του εργαζομένου εντός του οργανισμού αποτελεί πηγή σπατάλης. Επιπλέον, με κάθε μεταφορά χάνεται τμήμα της άτυπης/άρρητης γνώσης (tacit knowledge) διότι είναι γνωστό ότι η άτυπη γνώση είναι δύσκολο να καταγραφεί και να κεφαλοποιηθεί. Οι Poppendieck και Poppendieck (2007) αναφέρουν ότι μόνο το 6% της αρχικής γνώσης απομένει μετά από μια αλυσίδα 4 τέτοιων μεταβάσεων. Ως εκ τούτου, είναι απαραίτητο να μειωθεί η μεταφορά γνώσης μεταξύ εργαζομένων προκειμένου να μειωθεί η σπατάλη.
- Η εναλλαγή μεταξύ εργασιών (task switching) δημιουργεί σπατάλη χρόνου. Θεωρείται ότι η εναλλαγή εργασίας πρέπει να περιοριστεί στο ελάχιστο.
- Οι καθυστερήσεις είναι ένα πολύ συχνό φαινόμενο που συμβαίνει σε πολλές διαφορετικές καταστάσεις. Ένας από τους πιο σημαντικούς τύπους καθυστέρησης είναι η αναμονή ατόμων σε διαφορετικούς χώρους. Συνήθως, οι προγραμματιστές λαμβάνουν μια κρίσιμη απόφαση περίπου κάθε 15 λεπτά. Αυτές οι αποφάσεις μπορούν να ληφθούν άμεσα μόνο εάν υπάρχουν οι απαιτούμενες πληροφορίες στον χώρο που εργάζεται η ομάδα έργου. Αυτό μπορεί να συμβεί μόνο αν η ομάδα έργου αλλά και ο πελάτης βρίσκονται στον ίδιο χώρο.
- Τα σφάλματα του λογισμικού (defects) προκαλούν προβλήματα και συνήθως οδηγούν στη δυσαρέσκεια του πελάτη. Συνεπώς, ο στόχος θα πρέπει να είναι η ανάπτυξη και η παράδοση του λογισμικού με το χαμηλότερο δυνατό ποσοστό σφαλμάτων.

Μέρος Β – Βασικές ευέλικτες μέθοδοι

Η μέθοδος Scrum

The reasonable man adapts himself to the world;
the unreasonable one persists in trying
to adapt the world to himself.
Therefore, all progress depends
on the unreasonable man.

George Bernard Shaw

Σύνοψη
Στο τέταρτο κεφάλαιο παρουσιάζεται το πιο δημοφιλές πλαίσιο ανάπτυξης λογισμικού, η διεργασία Scrum. Κύριο χαρακτηριστικό της Scrum αποτελεί το γεγονός ότι βασίζεται κατά κύριο λόγο στην ομαδική εργασία και εγγυάται ποιοτικότερο λογισμικό εντός συγκεκριμένου σύντομου χρονικού διαστήματος. Όπως και οι υπόλοιπες ευέλικτες μεθοδολογίες έτσι και η διεργασία Scrum είναι προσανατολισμένη στους ανθρώπους και όχι στις διαδικασίες. . Στο κεφάλαιο αυτό θα παρουσιαστούν με λεπτομέρεια όλα τα χαρακτηριστικά της διεργασίας Scrum δηλαδή οι ρόλοι, τα τεχνουργήματα (artifacts) και οι τελετές (ceremonies).

4 Η διεργασία Scrum

4.1 Μια σύντομη εισαγωγή στη διεργασία Scrum

Η διεργασία Scrum αποτελεί το πιο δημοφιλές ευέλικτο πλαίσιο για την ανάπτυξη λογισμικού. Αποτελεί έναν επαναληπτικό κύκλο ζωής ανάπτυξης λογισμικού που έχει ως βασική έννοια την έννοια της επανάληψης που εκφράζεται με την αγγλική λέξη sprint. Σκοπός είναι όσο το δυνατόν μικρότερο χρονικό διάστημα κύκλου ανάπτυξης και παράδοσης τμημάτων κώδικα του συστήματος που έχουν υλοποιηθεί σε κάθε sprint. Για την υποστήριξη αυτής της διεργασίας, οι ομάδες Scrum χρησιμοποιούν συγκεκριμένους ρόλους, τεχνουργήματα (artifacts) και τελετές (ceremonies). Επιπλέον, με σκοπό τη διασφάλιση ότι η ομάδα έχει πετύχει τους στόχους κάνουμε ελέγχους, επιθεωρήσεις και επισκοπήσεις καθ' όλη τη διάρκεια του έργου.

Ο όρος Scrum[1] χρησιμοποιήθηκε για πρώτη φορά σε ένα επιστημονικό άρθρο των Hirotaka Takeuchi και Ikujiro Nonaka το 1986 με τίτλο «The New New Product Development Game». Ο όρος είναι ένα δάνειο από το άθλημα ράγκμπι, όπου ένα Scrum είναι μια μέθοδος επανέναρξης του παιχνιδιού στο ποδόσφαιρο ράγκμπι, κατά το οποίο οι παίκτες μαζεύονται σε ένα κυκλικό σχηματισμό και προσπαθούν να αποκτήσουν την κατοχή της μπάλας (Βλέπε Εικόνα 4.1). Και ακριβώς για αυτό το λόγο, ο όρος επιλέχθηκε από τους συγγραφείς του άρθρου επειδή δίνει έμφαση στην ομαδική εργασία. Στη συνέχεια ο Jeff Sutherland ξεκίνησε το πρώτο έργο Scrum το 1993. Ο Jeff Sutherland, σε συνεργασία με τον Ken Schwaber, παρουσίασαν το Scrum ως μια τυποποιημένη διεργασία ανάπτυξης λογισμικού το 1995 (Sutherland & Schwaber, 2007). Στη συνέχεια, η ανάπτυξη της μεθόδου ήταν ραγδαία έτσι ώστε σήμερα να χρησιμοποιείται από όλες σχεδόν τις εταιρείες ανάπτυξης λογισμικού αλλά και γενικότερα ως μια μεθοδολογία διαχείρισης έργων γενικού σκοπού. Η επίσημη έκδοση του Scrum από τους συγγραφείς της μεθόδου Ken Schwaber and Jeff Sutherland μπορεί να

[1] Ο όρος Scrum προφέρεται στα Ελληνικά ως σκραμ

βρεθεί στην ιστοσελίδα https://scrumguides.org/download.html, όπου είναι διαθέσιμη και η ελληνική μετάφραση.

Εικόνα 4.1: *Ο σχηματισμός Scrum στο rugby*

Η επιτυχία της μεθόδου οφείλεται στον τρόπο με τον οποίο προσεγγίζει και διαχειρίζεται την ανάπτυξη του λογισμικού. Προσδίδοντας ευελιξία στην οργάνωση και διαχείριση της διεργασίας ανάπτυξης λογισμικού, αυξάνεται η παραγωγικότητα της ομάδας, αλλά και η ικανότητα προσαρμογής της σύμφωνα με τις εκάστοτε ανάγκες που προκύπτουν. Δίνεται μεγάλη έμφαση στο πώς θα πρέπει να οργανωθεί η ομάδα ώστε να μπορέσει να επιτύχει το μέγιστο βαθμό αποτελεσματικότητας σε ένα διαρκώς μεταβαλλόμενο περιβάλλον. Η μέθοδος προβλέπει ένα εντελώς διαφορετικό μοτίβο οργάνωσης και διοίκησης του έργου σε σχέση με τις παραδοσιακές μεθόδους.

Το Scrum βασίζεται στην εμπειρική προσέγγιση ή τον εμπειρισμό, που σημαίνει ότι βασιζόμαστε στα δεδομένα, στα γεγονότα, στην εμπειρία που έχει αποκτηθεί και στα στοιχεία που προκύπτουν. Ο εμπειρισμός, ως φιλοσοφική προσέγγιση έχει τη βάση του στον Αριστοτέλη, έχει τις ρίζες του στην ιδέα πως οτιδήποτε γνωρίζουμε για τον κόσμο, είναι αυτά που μας επιτρέπουν να αντιληφθούμε με τις αισθήσεις μας και είναι επαληθεύσιμα μέσω της εμπειρικής απόδειξης. Στηρίζεται στην a posteriori γνώση, δηλαδή στην μη ύπαρξη έμφυτων αλλά ύστερων κατακτήσιμων ιδεών, και διακρίνει την εξωτερική αίσθηση μέσω της οποίας λαμβάνουμε τα εξωτερικά ερεθίσματα και την εσωτερική αίσθηση, μέσω της οποίας τα επεξεργαζόμαστε και δημιουργούμε τις έννοιες και στη συνέχεια τις γνώσεις.[2]

Ο εμπειρισμός βρίσκεται στον πυρήνα του Scrum, δηλαδή:

- Η γνώση προέρχεται από την εμπειρία.
- Παίρνουμε αποφάσεις για όσα είναι γνωστά.
- Δημιουργούμε επαυξήσεις του προϊόντος ώστε να είμαστε διαφανείς προς τον πελάτη
- Με βάση αυτά που μαθαίνουμε από την επαύξηση του λογισμικού προσαρμόζουμε την πορεία του έργου

Σύμφωνα με τους συγγραφείς της μεθόδου, η επιτυχημένη χρήση του Scrum σχετίζεται άμεσα με το να γίνουν οι άνθρωποι πιο ικανοί στο να ζουν με πέντε αξίες:

[2] https://el.wikipedia.org/

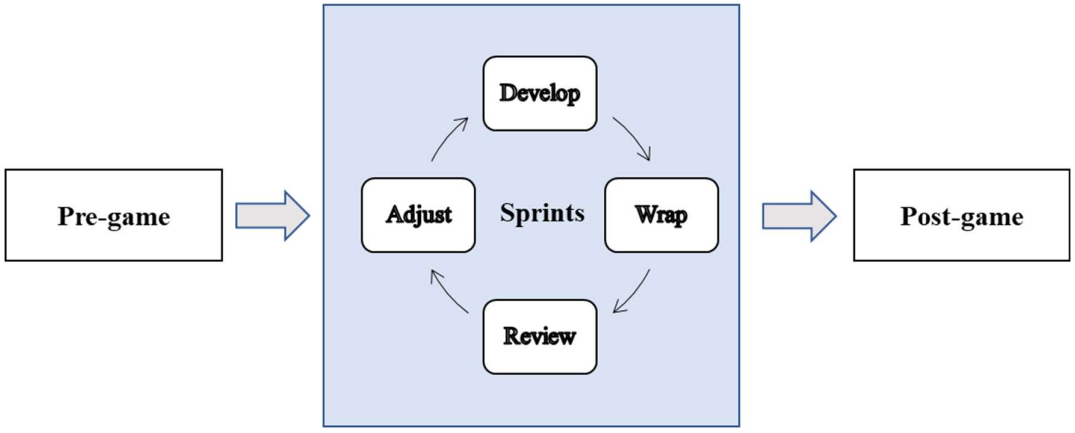
Δέσμευση (Commitment),
Εστίαση (Focus),
Ανοικτότητα (Openness),
Σεβασμό (Respect)
Θάρρος (Courage)

Εν συντομία, η δέσμευση των μελών της ομάδας επιτρέπει στα μέλη της ομάδας να προγραμματίσουν και να υλοποιήσουν καλύτερα το προϊόν, η εστίαση στο τελικό αποτέλεσμα τους επιτρέπει να ιεραρχούν τις απαιτήσεις και να εστιάζονται στα χαρακτηριστικά που προσθέτουν αξία, η ανοικτότητα και ο σεβασμός επιτρέπουν στα μέλη της ομάδας να συνεργάζονται καλύτερα και πιο αποτελεσματικά, ενώ το θάρρος τους επιτρέπει να αναλαμβάνουν κινδύνους και να επιχειρούν να εφαρμόσουν καινοτόμες λύσεις.

Το Scrum εκτός από μια μέθοδος ανάπτυξης λογισμικού είναι μια νέα κουλτούρα, ένας άλλος τρόπος σκέψης αλλά και ένας οδηγός στο πώς να παραχθεί αξία για τον πελάτη. Είναι λοιπόν ένας οδικός χάρτης παραγωγής αξίας (value roadmap). Αυτός ο οδικός χάρτης αποτελείται από στάδια που μας οδηγούν από το όραμα του προϊόντος στην υλοποίηση του προϊόντος, αλλά και στην ανασκόπηση του τρόπου εργασίας με σκοπό τη συνεχή βελτίωση.

Η μέθοδος Scrum χωρίζεται σε 3 διακριτές φάσεις (Schwaber and Beedle, 2008):

- Αρχική διερεύνηση (pre-game)
- Σχεδιασμός (game)
- Ολοκλήρωση (post game) (βλέπε Εικόνα 4.2)



Εικόνα 4.2: *Οι φάσεις της διεργασίας Scrum*

Η φάση της αρχικής διερεύνησης χωρίζεται σε δύο υπο-φάσεις:

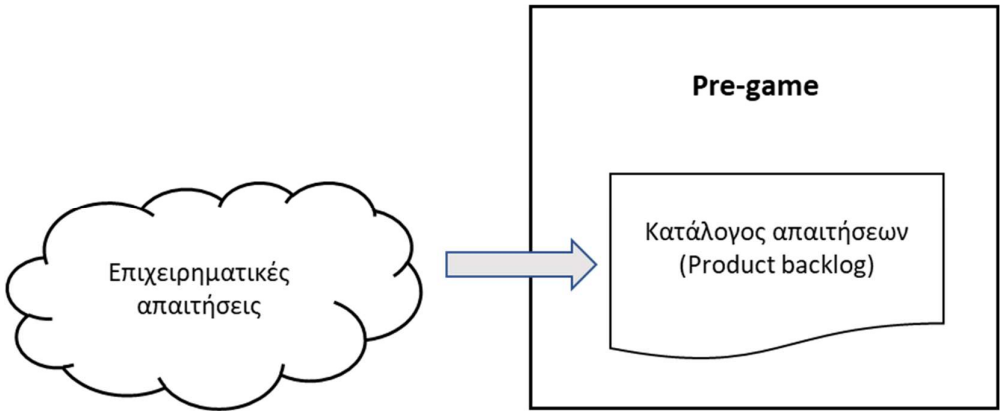
- Α) της ανάλυσης και
- Β) του υψηλού επιπέδου σχεδιασμού.

Κατά τη φάση της ανάλυσης (Βλέπε Εικόνα 4.3) γίνεται ο καθορισμός των προδιαγραφών του συστήματος καθώς και των απαιτήσεων του χρήστη από αυτό. Σημαντικό ρόλο στη διεργασία αυτή διαδραματίζει ο πελάτης, ο οποίος βρίσκεται σε διαρκή επικοινωνία με την ομάδα έργου και πρέπει να ορίσει το ποια θα είναι η λειτουργία του συστήματος. Το αποτέλεσμα αυτής της συνεργασίας μεταξύ πελάτη και ομάδας έργου είναι ένας κατάλογος χαρακτηριστικών του προϊόντος (product backlog list) που περιέχει όλες τις γνωστές, μέχρι την παρούσα στιγμή, απαιτήσεις. Οι απαιτήσεις κατατάσσονται ανάλογα με την

προτεραιότητά τους και υπολογίζεται η προσπάθεια που απαιτείται για την υλοποίησή τους. Ο κατάλογος αυτός ενημερώνεται συνεχώς καθ' όλη τη διάρκεια του έργου με νέες απαιτήσεις ή με νέα και πιο λεπτομερή στοιχεία, καθώς επίσης με ακριβέστερες εκτιμήσεις για την αναγκαία προσπάθεια καθώς και με νέες προτεραιότητες. Οι απαιτήσεις στη διεργασία Scrum αποτυπώνονται με τη χρήση ιστοριών χρηστών (user stories), οι οποίες περιγράφουν και αποτυπώνουν την αλληλεπίδραση του χρήστη με το σύστημα.

Η ομάδα ανάπτυξης θα πρέπει να ορίσει τις προτεραιότητες καθώς και το βαθμό σημαντικότητας των επιμέρους λειτουργιών του συστήματος που πρόκειται να αναπτυχθεί. Οι επιμέρους λειτουργίες αναλύονται μία προς μία, έτσι ώστε να διαπιστωθεί ο βαθμός δυσκολίας υλοποίησής τους, όπως και η τυχόν ενσωμάτωση επιπλέον τεχνογνωσίας στη διαδικασία της υλοποίησής τους. Αυτό, βέβαια, προσδίδει επιπλέον δυσκολία στην ομάδα ανάπτυξης, καθώς η ομάδα θα αφιερώσει ένα μεγάλο χρονικό διάστημα στην εκπαίδευση κάποιων μελών της, ώστε να μπορέσουν να ανταπεξέλθουν πλήρως στις ανάγκες του έργου.

Κατά τη φάση του υψηλού επιπέδου σχεδιασμού γίνεται ο αρχικός σχεδιασμός του συστήματος με βάση τις προδιαγραφές του συστήματος.



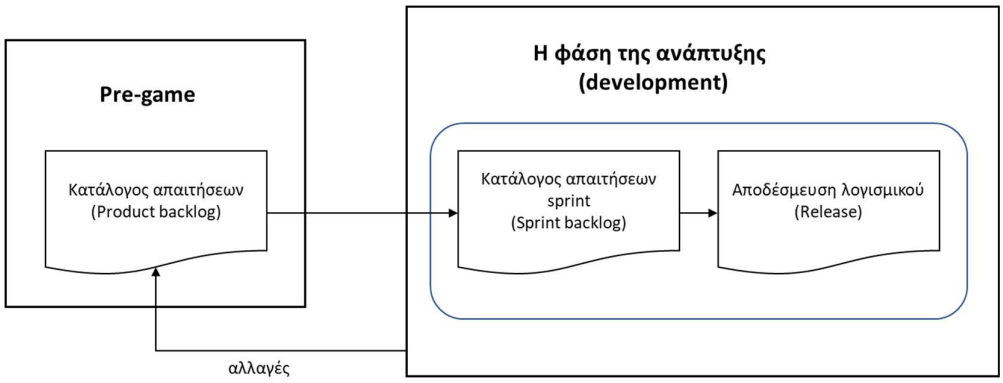
Εικόνα 4.3: *Η φάση του pre-game*

Η φάση της ανάπτυξης είναι ιδιαίτερα σημαντική και είναι αυτή που χαρακτηρίζει την ευελιξία της μεθόδου, καθότι η ομάδα ανάπτυξης πρέπει να λάβει υπόψη πολλές παραμέτρους, όπως χρόνος παράδοσης, ποιότητα, απαιτήσεις, πηγές, ενσωμάτωση επιπλέον τεχνογνωσίας κ.ά., που ίσως διαφοροποιηθούν κατά τη διάρκεια της ανάπτυξης, αλλά και να είναι ικανή να αντιμετωπίσει πολλούς αστάθμητους παράγοντες και κινδύνους.

Η ανάπτυξη στη Scrum γίνεται σε επαναληπτικούς κύκλους εργασίας, οι οποίοι ονομάζονται sprints. Τα sprints είναι φάσεις επαναληπτικού κύκλου ανάπτυξης που περιλαμβάνουν όλες τις κλασικές φάσεις ανάπτυξης του λογισμικού: καταγραφή απαιτήσεων, ανάλυση, σχεδιασμό, ανάπτυξη, παράδοση, έλεγχο. Οι επαναλήψεις δεν πρέπει να διαρκούν περισσότερο από έναν μήνα και εκτελούνται σειριακά χωρίς οποιεσδήποτε χρονικές καθυστερήσεις μεταξύ τους. Τα sprints έχουν συγκεκριμένο χρονοδιάγραμμα, το οποίο ποτέ δεν παρακάμπτεται, ακόμη και αν η καθορισμένη εργασία δεν έχει ολοκληρωθεί. Στην αρχή κάθε sprint η ομάδα επιλέγει τα user stories, τα οποία αποτελούν τις απαιτήσεις των πελατών, από μια λίστα στην οποία είναι ταξινομημένα με βάση την προτεραιότητά τους (backlog). Τα user stories δεν μπορούν να αλλάξουν κατά τη διάρκεια του sprint. Κάθε ημέρα η ομάδα οργανώνει σύντομες συναντήσεις για να ελέγξει την πρόοδό της και για να προσδιοριστούν τα επόμενα βήματα που χρειάζονται για την ολοκλήρωση της εργασίας.

Στο τέλος του sprint η ομάδα παρουσιάζει το αποτέλεσμα στον πελάτη και συγκεντρώνει σχόλια που αναφέρονται σε διορθώσεις και σε παραλήψεις και θα ενσωματωθούν στο επόμενο sprint. Η διεργασία Scrum στοχεύει στο τέλος κάθε επανάληψης να υπάρχει διαθέσιμο λειτουργικό και δοκιμασμένο προϊόν. Για τον λόγο αυτό τα μέλη της ομάδας θα πρέπει να συμφωνούν για το πότε μια εργασία θεωρείται ολοκληρωμένη (done). Το πότε μια εργασία θεωρείται ολοκληρωμένη καθορίζεται από τα κριτήρια αποδοχής του κάθε user story, το οποίο έχει επικρατήσει να ονομάζεται "Definition of Done". Τα κριτήρια αποδοχής

χρησιμοποιούνται για να αξιολογήσουμε πότε μια εργασία έχει ολοκληρωθεί και μπορεί να ενσωματωθεί στο εμπλουτισμένο προϊόν. Με τον τρόπο αυτό εξασφαλίζεται η υψηλότερη δυνατή ποιότητα για το τελικό προϊόν. Η φάση παρουσιάζεται αναλυτικά στην Εικόνα 4.4.



Εικόνα 4.4: *Η φάση της ανάπτυξης*

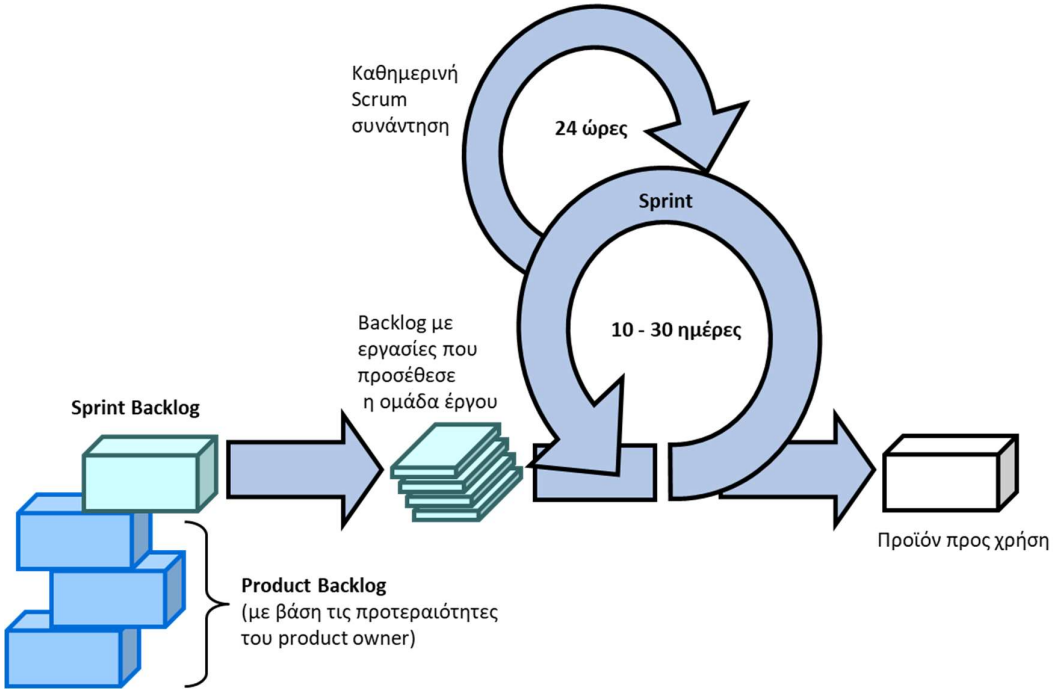
Στη φάση της ολοκλήρωσης εκτελούνται όλες εκείνες οι διαδικασίες που απαιτούνται για την περάτωση και την τελική παράδοση του έργου στον πελάτη. Η ομάδα ανάπτυξης, αφού επιβεβαιώσει, πάντα σε συνεργασία με τον πελάτη, ότι όλες οι απαιτήσεις έχουν υλοποιηθεί, κινεί τις διαδικασίες παράδοσης του συστήματος.

Στην Εικόνα 4.5 παρουσιάζονται τα βασικά βήματα και η λογική της μεθόδου Scrum.

Όπως αναφέραμε η διεργασία Scrum χρησιμοποιεί συγκεκριμένους ρόλους, τεχνουργήματα (artifacts) και τελετές (ceremonies) (βλέπε Εικόνα 4.6).

Έξι είναι οι ρόλοι που έχουν οριστεί ως αναγκαίοι για την ανάπτυξη έργων με την ευέλικτη διεργασία Scrum (Schwaber, & Beedle, 2002). Καθένας από αυτούς τους ρόλους έχει συγκεκριμένες υπευθυνότητες και εργασίες. Οι ρόλοι αυτοί είναι:

- η ομάδα Scrum (Scrum team),
- ο διαχειριστής Scrum (θα αναφέρεται εφεξής ως Scrum master)
- ο ιδιοκτήτης του προϊόντος (Product Owner),
- ο πελάτης (Customer),
- ο χρήστης (User) και
- η διοίκηση (Administration).



Εικόνα 4.5: *Η διεργασία Scrum*

Αντίστοιχα, τα τεχνουργήματα (artifacts) που παράγονται ή/και χρησιμοποιούνται από τη διεργασία Scrum είναι:

- ο κατάλογος των απαιτήσεων του προϊόντος (product backlog),
- ο κατάλογος των απαιτήσεων του sprint (sprint backlog), και
- η επαύξηση.

Τέλος, οι τελετές (ή γεγονότα) της διεργασίας Scrum είναι οι ακόλουθες:

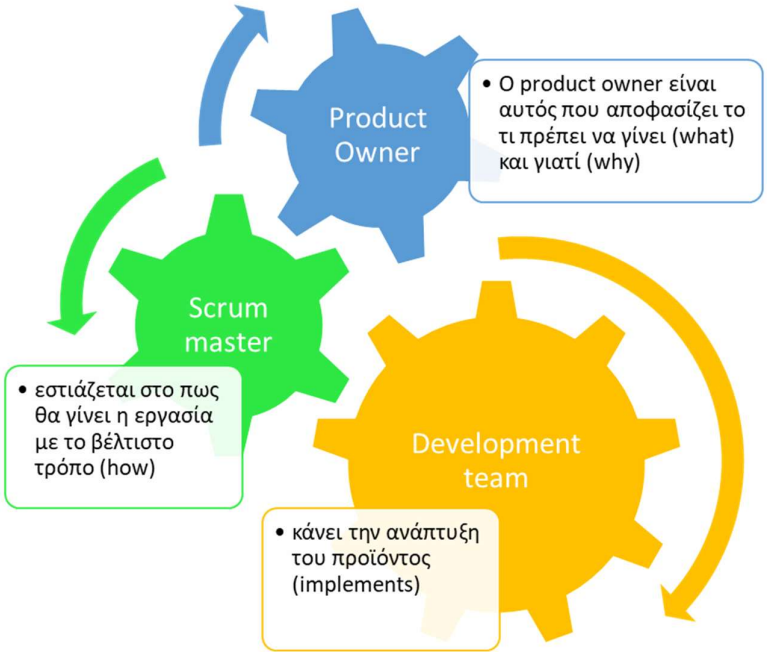
- το sprint,
- ο σχεδιασμός του sprint (sprint planning),
- η καθημερινή συνάντηση (daily scrum),
- η ανασκόπηση[3] του sprint (sprint review), καθώς και
- η επισκόπηση του sprint (sprint retrospective).

[3] Η λέξη review μεταφράζεται από την αγγλική ως ανασκόπηση, ενώ η λέξη retrospective ως επισκόπηση. Η επισκόπηση ως εργασία είναι η γενική, ανακεφαλαιωτική, βασική εξέταση του φαινομένου υπό μελέτη π.χ. η μελέτη της βιβλιογραφίας που γίνεται από τον ερευνητή προκειμένου να επιλέξει ένα ερευνητικό θέμα. Αντίθετα η ανασκόπηση είναι πιο συγκεκριμένη και περισσότερο εις βάθος, για τις ανάγκες του συγκεκριμένου sprint.

Εικόνα 4.6: *Τα βασικά στοιχεία της διεργασίας Scrum*

4.2 Ρόλοι και υπευθυνότητες στη διεργασία Scrum

Η ομάδα Scrum είναι μια ομάδα που έχει ως σκοπό την ανάπτυξη λογισμικού σε σύντομα τακτά χρονικά διαστήματα. Υπάρχουν τρεις βασικοί ρόλοι που υπάρχουν σε μια ομάδα Scrum: Ο ιδιοκτήτης του προϊόντος (product owner), ο Scrum master και η ομάδα ανάπτυξης (development team). Ο product owner είναι αυτός που αποφασίζει τι πρέπει να γίνει και γιατί, ο Scrum master εστιάζεται στο πώς θα γίνει η εργασία με το βέλτιστο τρόπο, ενώ η ομάδα ανάπτυξης κάνει την ανάπτυξη του προϊόντος (βλέπε Εικόνα 4.7).



Εικόνα 4.7: *Οι βασικοί ρόλοι της διεργασίας Scrum*

Άλλοι συμμετέχοντες μπορεί επίσης να είναι μέλη της ομάδας έργου, σε διαφορετικό βαθμό ανάλογα με τη φύση του έργου. Σε μεγαλύτερες επιχειρήσεις, συνήθως υπάρχουν αρκετά μέλη της επιχειρηματικής ομάδας που συμμετέχουν στη διαδικασία ανάπτυξης.

4.2.1 Η ομάδα ανάπτυξης Scrum

Ας ξεκινήσουμε λέγοντας λίγα λόγια για τις ομάδες γενικότερα. Οι οργανισμοί στη σημερινή πραγματικότητα στρέφονται όλο και περισσότερο σε δομές που βασίζονται σε ομάδες για να αντιμετωπίσουν την αυξανόμενη πολυπλοκότητα του περιβάλλοντος στο οποίο λειτουργούν. Οι ομάδες προσφέρουν μεγαλύτερη προσαρμοστικότητα, παραγωγικότητα και δημιουργικότητα από ό,τι μπορεί να προσφέρει ένας εργαζόμενος ατομικά και να δώσει πιο σύνθετες, καινοτόμες και ολοκληρωμένες λύσεις σε μεγάλη ποικιλία προβλημάτων. Αν και οι ομάδες έχουν σημαντικές δυνατότητες, η αποτυχία δεν είναι σπάνια, και μπορεί να έχει σημαντικές επιπτώσεις στον οργανισμό φορέα (π.χ. χαμένες προθεσμίες, χαμηλή παραγωγικότητα, χαμένα έσοδα, ελαττωματικά προϊόντα).

Η αποτυχία της ομάδας μπορεί να οφείλεται σε μια ποικιλία παραγόντων όπως ο κακός προγραμματισμός, η έλλειψη υποστήριξης από τη διοίκηση ή ο τρόπος λειτουργίας της ίδιας της ομάδας. Στην πραγματικότητα, πολλοί ερευνητές απέδειξαν ότι ο τρόπος λειτουργίας της ομάδας είναι αυτός που εξασφαλίζει την αποτελεσματικότητα αυτής. Στην πραγματικότητα, πολλές ομάδες ποτέ δεν καταφέρνουν να αξιοποιήσουν το πλήρες δυναμικό τους, άλλες πάλι αποτυγχάνουν εντελώς. Αυτό εγείρει το ερώτημα, ποιοι είναι οι παράγοντες που εξασφαλίζουν την επιτυχία μιας ομάδας; Η κύρια απάντηση στο ερώτημα αυτό είναι η ομαδική εργασία. Αν και η απάντηση είναι πολύ απλή και ο καθένας μας έχει άποψη για τι είναι και πώς προωθείται σε μια ομάδα, είναι ένα σημαντικό και ιδιαίτερα δημοφιλές ερευνητικό θέμα (Salas et al., 2005).

Οι Salas και λοιποί υποστηρίζουν ότι οι ομάδες απαιτούν για τη λειτουργία τους ένα σύνθετο μείγμα παραγόντων που περιλαμβάνουν την οργανωτική υποστήριξη, ατομικές δεξιότητες, καθώς και δεξιότητες ομαδικής εργασίας. Τα πέντε βασικά συστατικά που εντόπισαν στην έρευνά τους είναι: ηγεσία ομάδας, αμοιβαία παρακολούθηση της απόδοσης (mutual performance monitoring), πνεύμα αλληλοϋποστήριξης (backup behavior), προσαρμοστικότητα (adaptability) και προσανατολισμός ομάδας. Κάθε ένα από τα πέντε αυτά χαρακτηριστικά είναι προαπαιτούμενο για την αποτελεσματικότητα της ομάδας, αλλά και κάθε στοιχείο μπορεί να εκδηλωθεί διαφορετικά ανάλογα με το είδος της εργασίας και τους περιορισμούς ή τις ανάγκες της ομάδας.

Η ηγεσία της ομάδας αφορά την ικανότητα του ηγέτη να κατευθύνει και να συντονίζει τις δραστηριότητες άλλων μελών της ομάδας, να αξιολογεί την απόδοση της ομάδας, να αναθέτει εργασίες, να αναπτύσσει τις γνώσεις, τις δεξιότητες και τις ικανότητες της ομάδας, να παρακινεί τα μέλη της ομάδας, να σχεδιάζει και να οργανώνει και να δημιουργεί μια θετική ατμόσφαιρα.

- Η αμοιβαία παρακολούθηση της απόδοσης είναι η ικανότητα των μελών της ομάδας να αναπτύσσουν κοινές αντιλήψεις για το περιβάλλον της ομάδας και να εφαρμόζουν τις κατάλληλες στρατηγικές εργασιών για την παρακολούθηση της απόδοσης των άλλων μελών της ομάδας.
- Το πνεύμα αλληλοϋποστήριξης αφορά την ικανότητα πρόβλεψης των αναγκών άλλων μελών της ομάδας μέσω της γνώσης των ευθυνών τους. Αυτό περιλαμβάνει τη δυνατότητα μετάθεσης εργασιών μεταξύ των μελών της ομάδας για την επίτευξη ισορροπίας φόρτου εργασίας κατά τη διάρκεια περιόδων εργασίας με υψηλό φόρτο εργασίας ή μεγάλης πίεσης.
- Η προσαρμοστικότητα είναι η ικανότητα προσαρμογής της στρατηγικής της ομάδας ή της αλλαγής πορείας δράσης με βάση τις πληροφορίες που συλλέγονται από το περιβάλλον.
- Ο προσανατολισμός ομάδας αφορά την πεποίθηση των μελών να λαμβάνεται υπόψη η συμπεριφορά των άλλων μελών κατά τη διάρκεια της ομαδικής αλληλεπίδρασης και η πίστη στη σημασία των στόχων της ομάδας έναντι των στόχων των μεμονωμένων μελών.

Για τη σωστή λειτουργία της ομάδας απαιτούνται τρεις μηχανισμοί συντονισμού: κοινά νοητικά μοντέλα (shared mental model), επικοινωνία κλειστού βρόχου (closed loop communication) και αμοιβαία

εμπιστοσύνη (mutual trust). Ένα κοινό νοητικό μοντέλο οδηγεί σε μια κατάσταση κατά την οποία η γνώση που έχει κάθε μέλος μιας ομάδας σχετικά με τις επερχόμενες ενέργειες της ομάδας είναι τουλάχιστον παρόμοια με τη γνώση των άλλων μελών της ομάδας για τις ίδιες ενέργειες. Στο Scrum τα κοινά νοητικά μοντέλα υποστηρίζονται μέσω της συμμετοχής του ιδιοκτήτη προϊόντος, την εστίαση στο όραμα του έργου και τον προγραμματισμό που γίνεται με τη χρήση του product backlog και την επισκόπηση του sprint. Η καθημερινή συνάντηση είναι επίσης σημαντική για την κατανόηση των καθηκόντων των μελών της ομάδας. Η επικοινωνία κλειστού βρόχου εστιάζεται στο γεγονός ότι το μήνυμα πρέπει να γίνει αντιληπτό σωστά από τον παραλήπτη ανεξαρτήτως του μέσου που χρησιμοποιείται για τη μετάδοσή του. Στο Scrum η επικοινωνία γίνεται μέσω των θεσμοθετημένων τελετών αλλά δεν υπάρχει μηχανισμός που να εξασφαλίζει ότι η πληροφορία έχει γίνει κατανοητή από τον παραλήπτη. Η αμοιβαία εμπιστοσύνη είναι αναγκαία για τη σωστή λειτουργία της ομάδας και αφορά την κοινή πεποίθηση ότι τα μέλη της ομάδας θα εκτελέσουν τις εργασίες τους και θα προστατεύσουν τα συμφέροντα της ομάδας. Στο Scrum δεν υπάρχει μηχανισμός που να αναπτύσσει την αμοιβαία εμπιστοσύνη.

Σε μια αντίστοιχη μελέτη που έγινε στην εταιρεία Google, η οποία αφορούσε πάνω από 180 ενεργές ομάδες και έγιναν περισσότερες από 200 συνεντεύξεις με υπαλλήλους της Google, διαπιστώθηκε ότι υπάρχουν πέντε βασικές συμπεριφορές που υπάρχουν στις επιτυχημένες ομάδες (Rozovsky, 2015):

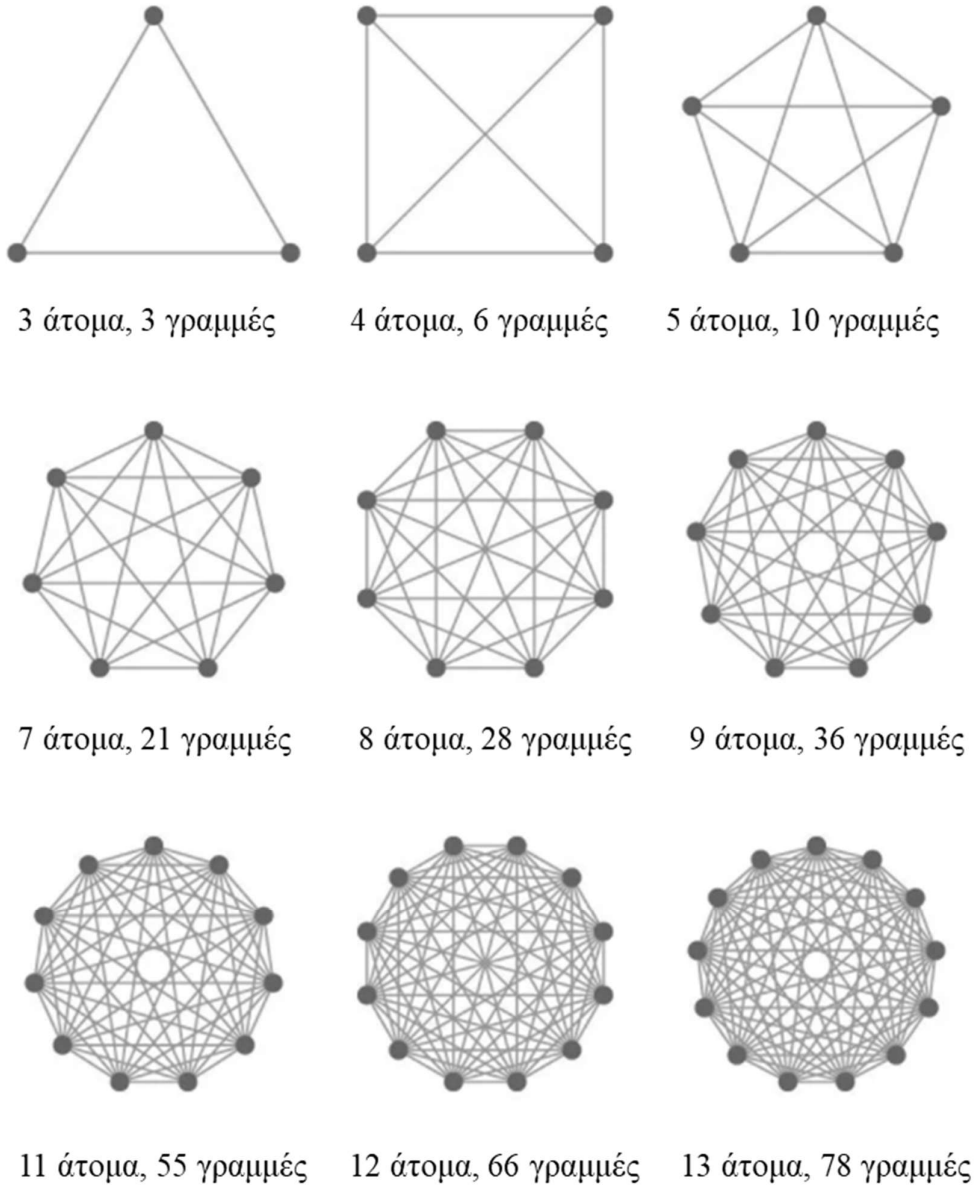
- Η ψυχολογική ασφάλεια που παρέχει η ομάδα στα μέλη της, ώστε να μπορούν να αναλαμβάνουν κινδύνους χωρίς να νιώθουν ανασφάλεια.
- Η αξιοπιστία των εργαζομένων στην ομάδα ως προς την επίτευξη των στόχων (dependability). Τα μέλη της ομάδας γνωρίζουν ότι οι συνάδελφοί τους θα ολοκληρώσουν την εργασία τους εγκαίρως και με τον καλύτερο δυνατό τρόπο.
- Η καθαρότητα της οργανωτικής δομής, των ρόλων και των στόχων τόσο της ομάδας όσο και του κάθε μέλους αυτής.
- Η σημασία που έχει η εργασία για το κάθε μέλος της ομάδας προσωπικά.
- Η πίστη ότι η εργασία που κάνουμε είναι σημαντική (impact) και χρήσιμη για τους χρήστες μας.

Σε ένα παιχνίδι σαν το ποδόσφαιρο ή σε κάποιο άλλο ομαδικό άθλημα, οι παίκτες είναι αυτοί που βγαίνουν στο γήπεδο και με το ταλέντο τους και την ικανότητά τους παίζουν και κερδίζουν το παιχνίδι. Όλοι οι άλλοι, προπονητής, γυμναστές, μάνατζερ, διευκολύνουν αυτή τη διαδικασία, και όταν η ομάδα νικά, είναι όλοι ευχαριστημένοι και επιτυχημένοι.

Αντίστοιχα, η ομάδα ανάπτυξης Scrum είναι το ταλέντο που είναι υπεύθυνο για τη δημιουργία, την ανάπτυξη του πραγματικού προϊόντος και την επιτυχία του εγχειρήματος. Είναι αυτή που καθορίζει πώς θα γίνει κάτι αλλά και με ποιο ρυθμό. Με άλλα λόγια, καθορίζει το πώς θα αναπτυχθούν οι απαιτήσεις και πόσες απαιτήσεις μπορούν να υλοποιηθούν σε ένα sprint.

Η ομάδα ανάπτυξης περιορίζεται σκόπιμα σε περίπου έξι μέλη, συν-πλην τρία. Το βέλτιστο μέγεθος ομάδας ανάπτυξης είναι να είναι αρκετά μικρή, ώστε να παραμείνει ευέλικτη και αρκετά μεγάλη για να μπορεί να ολοκληρώσει σημαντικές εργασίες κατά τη διάρκεια ενός sprint. Επιπλέον, αυτό το μέγεθος επιτρέπει μια ομάδα να είναι αυτοδύναμη και αυτοοργανούμενη, με ποικιλία δεξιοτήτων, αλλά δεν πρέπει να είναι πολύ μεγάλο ώστε να είναι δυσκίνητη. Θυμηθείτε, ότι οι γραμμές επικοινωνίας μέσα σε μια ομάδα αυξάνονται γεωμετρικά ανάλογα με τον αριθμό των μελών. Επίσης όταν έχουμε περισσότερα μέλη χάνουμε την ικανότητα αυτοοργάνωσης και χρειαζόμαστε έναν διαχειριστή έργου, που να είναι υπεύθυνος για το συντονισμό όλης της επικοινωνίας (Layton & Ostermiller, 2017).

Στο Scrum με το μικρό αριθμό των μελών της ομάδας δεν έχουμε το κόστος των πολλών γραμμών επικοινωνίας (Brooks, 1995) ή την ανάγκη ενός διαχειριστή έργου (βλέπε Εικόνα 4.8).



Εικόνα 4.8: *Οι γραμμές επικοινωνίας σε συνάρτηση με τον αριθμό μελών μιας ομάδας*

Η αποκλειστική απασχόληση των μελών της ομάδας ανάπτυξης

Αποτελεί καλή πρακτική οι εργαζόμενοι στην ομάδα έργου να είναι πλήρους απασχόλησης ώστε να μην υπάρχει το φαινόμενο της εναλλαγής μεταξύ διαφορετικών εργασιών (context switching, multitasking) το οποίο έχει αποδειχτεί ότι μειώνει την αποδοτικότητα, επιβραδύνει τους εργαζόμενους, εμποδίζει τη δημιουργικότητα και προκαλεί άγχος στους εργαζομένους (Dean & Webb, 2011). Η παράλληλη εργασία σε διαφορετικά έργα μπορεί επίσης να θεωρηθεί ως σημαντική αιτία παρατεταμένης διάρκειας ενός έργου. Επιπλέον, η εναλλαγή μεταξύ εργασιών αυξάνει ακόμη περισσότερο τους χρόνους ολοκλήρωσης αυτών και αναγκάζει τα μέλη της ομάδας έργου να σταματούν συχνά για να θυμούνται πού βρίσκονταν κάθε φορά που αλλάζουν σε διαφορετική εργασία, ιδιαίτερα αν αυτό συμβαίνει σε διαφορετικά έργα (Goldratt, 1997).

Επίσης, επειδή η ευελιξία επικεντρώνεται στην άμεση δημιουργία αξίας για τους πελάτες, οι οργανισμοί που επιλέγουν να δημιουργήσουν ομάδες έργων με μέλη μη αποκλειστικής απασχόλησης αμβλύνουν άμεσα τα αποτελέσματά τους, διότι η ευελιξία ζητά από τους οργανισμούς:

- Να επικεντρώνονται στην παράδοση των έργων και της παραγόμενης αξίας σε όσο πιο σύντομα χρονικά διαστήματα.
- Να είναι πρόθυμοι να αναθέσουν στους καλύτερους εργαζομένους του οργανισμού, αυτά τα λίγα σε αριθμό ζωτικής σημασίας και υψηλής αξίας στρατηγικά έργα.
- Να δίνουν προτεραιότητα στα έργα αυτά χωρίς να επιτρέπουν σε θέματα παραγωγής και λειτουργίας να δημιουργούν ανεπίλυτα εμπόδια.

Στην πράξη, αν αναθέσουμε εργασίες στους εργαζόμενους σε πολλά διαφορετικά έργα, καθυστερούμε τα έργα υψηλής αξίας. Αυτό ισχύει τόσο για τις επιχειρήσεις όσο και για τις τεχνολογικές ομάδες. Επιπλέον είναι τέτοιο, σήμερα, το ειδικό βάρος της τεχνολογίας που σύγχρονη επιχείρηση θα έπρεπε να είναι μια επιχείρηση τεχνολογίας και ως εκ τούτου, οι καλύτεροι επιχειρηματικά και τεχνολογικά εργαζόμενοι της επιχείρησης πρέπει να είναι αφοσιωμένοι σε έργα που δημιουργούν αξία με βάση την τεχνολογία. Αυτή αποτελεί μια σημαντική αλλαγή στη στρατηγική σκέψη και δομή των επιχειρήσεων (Belling, 2020).

Η διαλειτουργικότητα των μελών της ομάδας ανάπτυξης

Εκτός από τις μικρές, αποκλειστικής απασχόλησης, αυτοδιαχειριζόμενες ομάδες έργου, οι ευέλικτες μέθοδοι υποθέτουν την προσέγγιση ότι αυτές οι ομάδες είναι και διαλειτουργικές (cross-functional). Οι διαλειτουργικές ομάδες μπορούν να εργαστούν πιο γρήγορα και αποτελεσματικά σε σχέση με τις ομάδες που αποτελούνται αποκλειστικά από ειδικούς, ή ακόμη χειρότερα, που βασίζονται σε ειδικούς εκτός της ομάδας. Αυτό σημαίνει ότι όλοι στην ομάδα ανάπτυξης πρέπει να είναι ικανοί ώστε να εκτελέσουν όλα όσα απαιτεί το αντικείμενο του έργου. Η προσέγγιση αυτή έχει πλεονεκτήματα και μειονεκτήματα που θα αναδείξουμε στην παρακάτω συζήτηση.

Η διαλειτουργικότητα των μελών της ομάδας ανάπτυξης έχει τα παρακάτω οφέλη:

- Επιτρέπει την συμβολή όλων των μελών της ομάδας στην ανάπτυξη και στην εξεύρεση της βέλτιστης λύσης.
- Επιτρέπει τον προγραμματισμό σε ζεύγη (pair programming), μια από τις πιο γνωστές και δημοφιλείς ευέλικτες πρακτικές.
- Βελτιώνει τις δεξιότητες των μελών της ομάδας ανάπτυξης.
- Επιτρέπει στους εργαζόμενους να εργάζονται σε ποικιλία εργασιών και διατηρεί το ενδιαφέρον τους για το έργο αμείωτο.

Υπάρχουν διάφοροι τρόποι για τη δημιουργία διαλειτουργικών εργαζομένων σε ευέλικτες ομάδες ανάπτυξης:

- Η μη χρήση τίτλων για τις θέσεις μέσα στην ομάδα (π.χ. διευθυντής έργου, υπεύθυνος ποιότητας) ενθαρρύνει την άμιλλα, αφού τα νεότερα μέλη της ομάδας επιζητούν να μάθουν ή να παράγουν πιο γρήγορα. Αντίστοιχα, τα πιο έμπειρα μέλη της ομάδας θέλουν να ενημερώνονται για τις νέες τεχνικές εξελίξεις ώστε να μην ξεπεραστούν οι γνώσεις τους από τα νεαρότερα μέλη της ομάδας. Συνεπώς, η έλλειψη τίτλων μεταφέρει το κέντρο βάρους από την ιεραρχία στις κατάλληλες δεξιότητες, ενθαρρύνοντας έτσι την ανάπτυξη δεξιοτήτων.
- Η χρήση της πρακτικής προγραμματισμού σε ζεύγη (pair programming) (Williams & Kessler, 2003). Η πρακτική αυτή προέρχεται από τη μέθοδο Extreme Programming (Beck, 2005), και στην οποία δύο προγραμματιστές συνεργάζονται για το ίδιο κομμάτι λειτουργικότητας. Ο προγραμματιστής Α αναπτύσσει τον κώδικα, ενώ ο προγραμματιστής Β είναι ελεύθερος να σκεφτεί, να παρατηρεί και να διορθώνει, ενώ ταυτόχρονα σκέπτεται στρατηγικά τη λειτουργικότητα (επεκτασιμότητα,

κίνδυνοι κ.ο.κ.). Ο ρόλος του προγραμματιστή Α και Β αλλάζει κατά τη διάρκεια της ημέρας. Η στενή αυτή συνεργασία οδηγεί τελικά σε λιγότερα λάθη και μεγαλύτερη παραγωγικότητα.

- Παρακολούθηση εν ώρα εργασίας (job shadowing). Και πάλι, δύο προγραμματιστές συνεργάζονται, αλλά σε αυτή την περίπτωση, μόνο ο ένας εργάζεται, ενώ ο άλλος είναι εκπαιδευόμενος, παρακολουθεί και μαθαίνει.

Οι ομάδες με πολλά άτομα συνήθως οργανώνονται σε μικρότερες ομάδες και κάθε μια αναλαμβάνει την ανάπτυξη διαφορετικών υποσυστημάτων/χαρακτηριστικών του προϊόντος, ενώ ταυτόχρονα διατηρούν την μεταξύ τους επικοινωνία. Γενικότερα, η προσθήκη περισσότερων ατόμων στην επίλυση ενός προβλήματος ή σε ένα έργο, κάνει την εκτέλεση του έργου πιο δύσκολη για τον απλό λόγο ότι αυξάνονται οι ανάγκες επικοινωνίας. Ας θυμηθούμε τον «κανόνα» που ο F. Brooks παρουσιάζει στο βιβλίο του «The Mythical Man-Month», ότι το «να προσθέσεις προσωπικό σε ένα έργο που έχει καθυστερήσει πολύ συχνά επιφέρει επιπλέον καθυστέρηση» (Brooks, 1995). Όμως, το να είμαστε σε θέση να οργανωνόμαστε σε μεγαλύτερες ομάδες είναι συχνά αναγκαίο, διότι τα προβλήματα είναι συχνά μεγάλα σε μέγεθος αλλά και πολύπλοκα. Για να επιτευχθεί η κλιμάκωση της οργάνωσης δημιουργήθηκε η τεχνική **Scrum of Scrums** – που μερικές φορές αναφέρεται ως SoS.

Η μεθοδολογία Scrum of Scrums εφαρμόστηκε για πρώτη φορά το 1996 από τους Jeff Sutherland και Ken Schwaber, που χρειάζονταν έναν τρόπο να συντονίσουν οκτώ επιχειρηματικές μονάδες (business units) με πολλές γραμμές προϊόντων (product lines) ανά επιχειρηματική μονάδα ή/και να συγχρονίσουν μεμονωμένες ομάδες μεταξύ τους. Έτσι δοκίμασαν έναν νέο τρόπο να κλιμακώσουν τις ομάδες Scrum για να επιτύχουν αυτόν τον στόχο. Η εμπειρία ενέπνευσε τον Sutherland να δημοσιεύσει ένα άρθρο το 2001 με τίτλο «Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies», το οποίο ανέφερε για πρώτη φορά δημόσια το Scrum of Scrums (Sutherland, 2001).

Συνοψίζοντας θα μπορούσαμε να πούμε ότι μια ομάδα Scrum έχει τα εξής κύρια χαρακτηριστικά:

- Αποτελείται από 5-9 άτομα. Σε μεγαλύτερα έργα δημιουργούνται πολλές ομάδες Scrum.
- Τα μέλη της ομάδας εκτελούν όλες τις εργασίες ανάπτυξης λογισμικού (cross-functional).
- Αποτελείται από μέλη πλήρους απασχόλησης, ενώ μερική απασχόληση μπορεί να υπάρξει μόνο για εξειδικευμένες εργασίες (π.χ. database administrator).
- Η ομάδα είναι αυτοργανωμένη.
- Τα μέλη της ομάδας εργάζονται στον ίδιο χώρο.
- Τα μέλη της ομάδας συναντιούνται καθημερινά.
- Τα μέλη της ομάδας δεν έχουν τίτλους εργασίας.
- Η ομάδα είναι συνολικά υπεύθυνη για το τελικό αποτέλεσμα.

4.2.2 Ο Scrum master

Ο Scrum master είναι υπεύθυνος για τη διευκόλυνση της διαδικασίας ανάπτυξης, διασφαλίζοντας ότι η ομάδα έργου χρησιμοποιεί όλο το φάσμα των κατάλληλων ευέλικτων αξιών, πρακτικών και κανόνων. Ο Scrum master έχει χαρακτηριστεί στην ευέλικτη βιβλιογραφία ως «ηγέτης- υπηρέτης» που είναι υπεύθυνος για την προώθηση και τη διατήρηση του Scrum στο πλαίσιο του έργου (Sutherland and Schwaber, 2020; Greenleaf, 1988). Η ηγεσία-υπηρέτης είναι μια φιλοσοφία ηγεσίας όπου ο στόχος του ηγέτη είναι να υπηρετήσει την ομάδα και δεν εστιάζεται μονόπλευρα στην υλοποίηση των στόχων της επιχείρησης. Η ηγεσία-υπηρέτης περιγράφεται ως μια ανθρωποκεντρική προσέγγιση στην οποία ο ηγέτης δεν παρακινείται από την ανάγκη για εξουσία, αλλά από το πάθος στην εξυπηρέτηση των μελών της ομάδας, ενεργώντας πέρα από το προσωπικό του συμφέρον, φροντίζοντας για τους οπαδούς, δημιουργώντας ευκαιρίες ανάπτυξης, αυξάνοντας την αυτονομία των οπαδών και ενθαρρύνοντάς τους να σκέφτονται ανεξάρτητα, δημιουργώντας έτσι χώρο για συμμετοχική λήψη αποφάσεων και χρησιμοποιώντας πειθώ αντί για εξαναγκασμό. Ένας

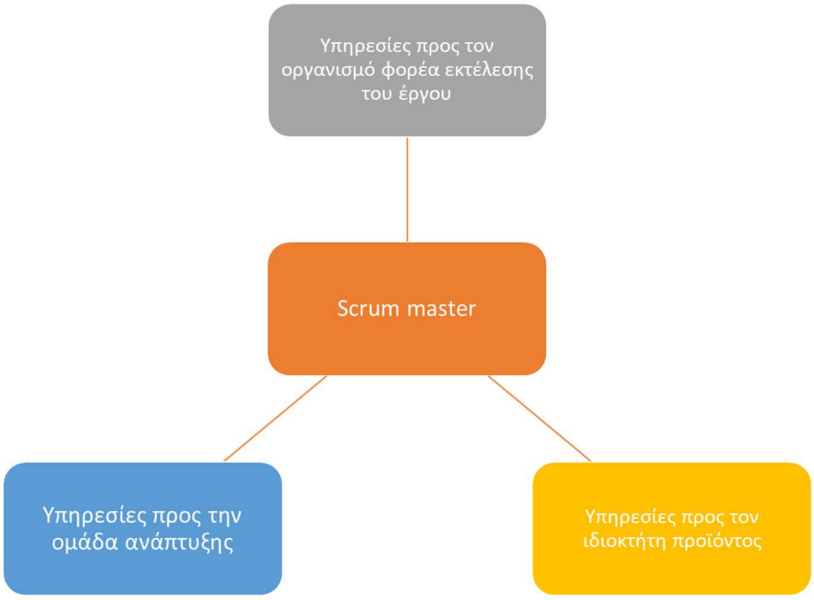
ηγέτης-υπηρέτης μοιράζεται την εξουσία, βάζει πάνω απ' όλα τις ανάγκες των εργαζομένων και πρωταρχικά βοηθά τα άτομα της ομάδας έργου να αναπτυχθούν και να αποδώσουν όσο το δυνατόν καλύτερα (Van Dierendonck, 2011).

Η έννοια της ηγεσίας-υπηρέτης είναι κεντρική για το ρόλο του Scrum master, αφού είναι αυτή που χαρακτηρίζει με τον καλύτερο τρόπο τον ρόλο. Στη διεργασία Scrum ο ηγέτης δεν είναι ένας ρόλος προικισμένος με εξουσία αλλά είναι ένα ρόλος με κύριο μέλημα τη διευκόλυνση της ομάδας στο να πετύχει τους στόχους της. Με βάση τις αρχές της ηγεσίας-υπηρέτης, και τις αρχές του Scrum, ένας Scrum master θα πρέπει να υπηρετεί τους συμμετέχοντες, να προωθεί την αίσθηση της κοινότητας, να υποστηρίζει τη συλλογική λήψη αποφάσεων, καθώς και να είναι σημείο αναφοράς στην εφαρμογή της ευέλικτης διεργασίας (βλέπε Εικόνα 4.9) (Holtzhausen & de Klerk, 2018).

Εικόνα 4.9: *Ο Scrum master ως ηγέτης-υπηρέτης*

Επίσης, ο Scrum master διεξάγει καθημερινές συναντήσεις συντονισμού και αφαιρεί τυχόν εμπόδια που συναντά η ομάδα (Noll et al,. 2017). Οι υπηρεσίες που πρέπει να προσφέρει ένας Scrum master είναι προς τρεις κατευθύνσεις (βλέπε Εικόνα 4.10) (Sutherland and Schwaber, 2020):

- Προς τον οργανισμό - φορέα εκτέλεσης του έργου.
- Προς την ομάδα ανάπτυξης.
- Προς τον ιδιοκτήτη προϊόντος.



Εικόνα 4.10: *Οι υπηρεσίες που πρέπει να προσφέρει ένας Scrum master*

Ο Scrum master εξυπηρετεί τον οργανισμό – φορέα εκτέλεσης του έργου με διάφορους τρόπους, μεταξύ των οποίων:

- Καθοδηγώντας τον οργανισμό στην υιοθέτησή της διεργασίας Scrum ειδικά όταν δεν υπάρχει εμπειρία, αποτελώντας ένα σημείο αναφοράς για την εφαρμογή της ευέλικτης διεργασίας (Process anchor),
- Σχεδιάζοντας τα συγκεκριμένα έργα Scrum στα οποία συμμετέχει.
- Βοηθώντας τους εργαζόμενους και συμμετέχοντες να κατανοήσουν και να εφαρμόσουν την διεργασία Scrum και γενικότερα τις αρχές της εμπειρικής ανάπτυξης προϊόντων.
- Κάνοντας αλλαγές που αυξάνουν την παραγωγικότητα της ομάδας έργου στην οποία ανήκει.
- Συνεργαζόμενος με άλλους Scrum master για την αποτελεσματικότερη εφαρμογής της διεργασίας Scrum στον οργανισμό.

Ο Scrum master προσφέρει υπηρεσίες προς την ομάδα ανάπτυξης όπως μεταξύ άλλων:

- Να καθοδηγεί τα μέλη της ομάδας
- Να βοηθά την ομάδα στη δημιουργία νέων εκδόσεων του λογισμικού που είναι υψηλής επιχειρηματικής αξίας
- Να επιλύει τα προβλήματα που θέτουν εμπόδια στην επιτυχή ανάπτυξη του Scrum, και
- Να διασφαλίζει ότι όλες οι δραστηριότητες της ομάδας είναι παραγωγικές, και εντός του χρονικών ορίων

Ο Scrum master προσφέρει υπηρεσίες προς τον ιδιοκτήτη προϊόντος όπως μεταξύ άλλων:

- Να βοηθάει στον ορισμό των στόχων για το υπ’ ανάπτυξη προϊόν
- Να βοηθά στη διαχείριση αλλά και στο σαφή ορισμό του Product Backlog,
- Να βοηθάει την ομάδα ανάπτυξης στο να κατανοήσει την επιχειρηματική ανάγκη καθώς και τις ιστορίες χρηστών.

- Να βοηθάει στο σχεδιασμό του προϊόντος
- Να διευκολύνει τη συνεργασία ανάμεσα στα μέλη της ομάδας αλλά και μεταξύ της ομάδας και των συμμετεχόντων

4.2.3 Ο ιδιοκτήτης προϊόντος (product owner)

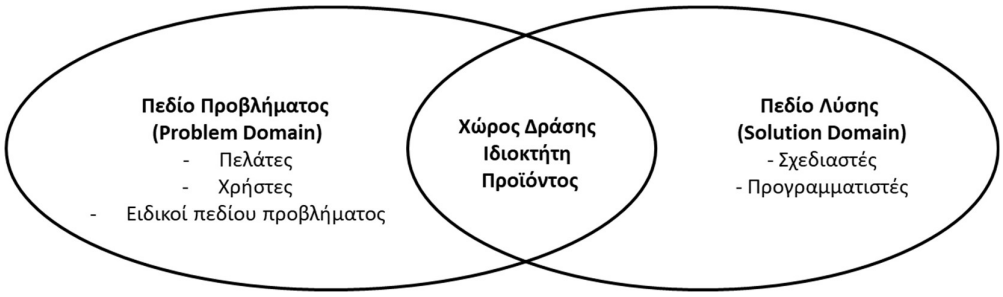
Στον βιβλίο «Scrum Guide» (Schwaber 2009), ο Ken Schwaber αναφέρει σχετικά με τον ιδιοκτήτη του προϊόντος:

> Ο ιδιοκτήτης του προϊόντος είναι ο μόνος υπεύθυνος για τη διαχείριση του product backlog και τη διασφάλιση της αξίας του έργου που εκτελεί η ομάδα. Το άτομο αυτό διαχειρίζεται το product backlog και διασφαλίζει ότι είναι ορατό σε όλους.

Συνεπώς, ο ιδιοκτήτης του προϊόντος είναι υπεύθυνος στο να προσδιορίσει τα επιθυμητά χαρακτηριστικά του προϊόντος στην ομάδα έργου, καθώς και για την ιεράρχηση αυτών. Συνεπώς, ο ιδιοκτήτης προϊόντος εντοπίζει τα χρήσιμα χαρακτηριστικά του προϊόντος, τα καταγράφει σε μια λίστα απαιτήσεων κατά προτεραιότητα, σε συνεργασία με την ομάδα έργου. Συνήθως τα χαρακτηριστικά αυτά καταγράφονται με τη μορφή «ιστοριών χρηστών» (user stories).

Ο ιδιοκτήτης του προϊόντος λειτουργεί ως «γέφυρα» ή «σύνδεσμος» μεταξύ του χώρου του προβλήματος και του χώρου λύσης, και κινείται σε αυτό που ονομάζουμε χώρο συνύπαρξης. Ο ιδιοκτήτης του προϊόντος πρέπει να κατανοεί και τους δύο κόσμους, τον επιχειρηματικό κόσμο και τον κόσμο της ανάπτυξης λογισμικού, και να αποτελεί έναν μεταφραστή μεταξύ αυτών των δύο κόσμων (βλέπε Εικόνα 4.11).

Συνεπώς ο ιδιοκτήτης του προϊόντος έχει καθοριστικό ρόλο στην ανάπτυξη του συστήματος και πρέπει να είναι διαθέσιμος ανά πάσα στιγμή ώστε να μπορεί να ανταποκριθεί σε κάθε περίπτωση που η ομάδα ανάπτυξης χρειάζεται διευκρινήσεις ή αποφάσεις σχετικά με την πορεία ανάπτυξης.



Εικόνα 4.11: *Χώρος Δράσης του «Ιδιοκτήτη Προϊόντος»*

4.2.3.1 Βασικές εργασίες του ρόλου του «Ιδιοκτήτη Προϊόντος»

Εξαιτίας αυτού του γεγονότος, ο ιδιοκτήτης του προϊόντος φέρει την ευθύνη να διασφαλίσει ότι το προϊόν θα είναι επιτυχημένο και προσφέρει προστιθέμενη αξία στον πελάτη. Γενικότερα ο ιδιοκτήτης του προϊόντος πρέπει να εκτελέσει τις ακόλουθες εργασίες (Βλέπε Εικόνα 4.12):

Εικόνα 4.12: *Βασικές εργασίες του ρόλου του «Ιδιοκτήτη Προϊόντος»*

<u>Καθορισμός του οράματος</u>

Μια από τις πρώτες δραστηριότητες του ιδιοκτήτη προϊόντος σύμφωνα με τον Deemer et al. (2012) είναι ο καθορισμός του οράματος του προϊόντος.

Το όραμα μιας επιχείρησης είναι η αιτιολόγηση της ύπαρξης μιας επιχείρησης ή ενός προϊόντος. Το όραμα αφορά τον προσανατολισμό της επιχείρησης προς το μέλλον και ενσωματώνει τις προσδοκίες, και τις επιθυμίες του επιχειρηματία για την πορεία της επιχείρησης καθώς αποτυπώνει το πώς θα ήθελε ο επιχειρηματίας να δει την επιχείρησή του στο μέλλον. Αντίστοιχα, το όραμα του προϊόντος είναι μια περιγραφή της «αξίας» του προϊόντος: ποια είναι τα προβλήματα που επιλύει, για ποιον προορίζεται και γιατί τώρα είναι η κατάλληλη στιγμή για να φτιαχτεί. Το όραμα του προϊόντος δίνει στην ομάδα ανάπτυξης την εποπτική εικόνα καθώς και για τι εργάζονται και γιατί. Ένα σωστό όραμα προϊόντος πρέπει να πληροί τα ακόλουθα χαρακτηριστικά:

- Πρέπει να είναι φιλόδοξο αφού αποβλέπει στο μέλλον
- Να οδηγεί σε ενέργεια
- Να συνδέεται με τους στρατηγικούς στόχους της επιχείρησης
- Να είναι κατευθυντικό

Το όραμα είναι σημαντικό, διότι είναι αυτό που θα εμπνεύσει τους συμμετέχοντες και την ομάδα έργου ώστε να εργαστεί με συνέπεια αλλά και ενθουσιασμό ώστε να επιτύχει τους στόχους του έργου. Για το λόγο αυτό θα πρέπει ο ιδιοκτήτης του προϊόντος να ακολουθεί μια σειρά πρακτικών όπως:

- Να επικοινωνεί το όραμα με όλους τους συμμετέχοντες
- Να εξελίσσει το όραμα καθώς εξελίσσεται/αλλάζει το περιβάλλον ή καθώς μαθαίνει κατά τη διάρκεια του έργου

- Να εστιάζεται στην αξία του προϊόντος, όπως αυτή εκλαμβάνεται από τον πελάτη και όχι στις τεχνικές λεπτομέρειες
- Να ταιριάζει το όραμα του προϊόντος με τη στρατηγική του πελάτη
- Να επαληθεύει το όραμα τόσο με τον πελάτη όσο και με την ομάδα έργου
- Ένα σύνολο ερωτήσεων που πρέπει να απαντηθούν ώστε να δημιουργηθεί ένα όραμα που πληροί όλες τις παραπάνω προϋποθέσεις είναι μια σύντομη περιγραφή του πελάτη στόχου, των βασικών πλεονεκτημάτων του προϊόντος έναντι του ανταγωνισμού καθώς και του μότο[4] (βλέπε Εικόνα 4.13).

Για τον πελάτη _____ πελάτης στόχος _____
που χρειάζεται _____ περιγραφή της ανάγκης _____

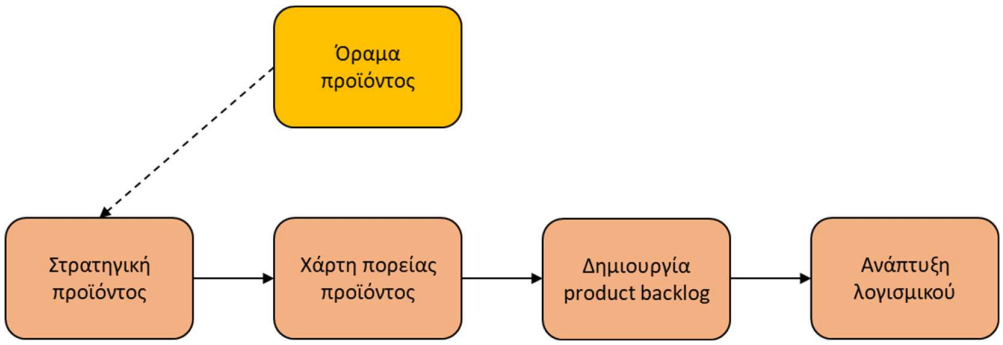
Το προϊόν _____όνομα προϊόντος _____
που προσφέρει _____βασικά πλεονεκτήματα και γιατί να το προμηθευτεί_____

Το προϊόν είναι
διαφορετικό από _____τα χαρακτηριστικά των βασικών ανταγωνιστικών προϊόντων_____
διαφορετικό από _____τα χαρακτηριστικά των βασικών ανταγωνιστικών προϊόντων_____

Το μότο _____γιατί το προϊόν μας είναι το καλύτερο_____

Εικόνα 4.13: *Βήματα ορισμού οράματος προϊόντος*

Συνεπώς το όραμα αποτελεί ένα αναγκαίο βήμα που θα πρέπει να κάνει ο ιδιοκτήτης προϊόντος καθώς αποτελεί την αφετηρία δημιουργίας ενός επιτυχημένου προϊόντος. Η σχέση αυτή απεικονίζεται στην Εικόνα 4.14.



Εικόνα 4.14: *Το όραμα προϊόντος είναι το πρώτο βήμα υλοποίησης ενός επιτυχημένου προϊόντος*

Ένα χρήσιμο εργαλείο για να μπορέσει η ομάδα έργου να διατηρήσει και να υλοποιήσει αυτό το όραμα είναι η δημιουργία ενός χάρτη πορείας προϊόντος (product roadmap). Ο χάρτης πορείας προϊόντος είναι ένα οπτικό βοήθημα που σκιαγραφεί το όραμα και την κατεύθυνση του προϊόντος στον χρόνο. Αποτελεί τόσο το στρατηγικό οδηγό αναφοράς των συμμετεχόντων όσο και σχέδιο εκτέλεσης. Στην Εικόνα 4.15 παρουσιάζεται ο χάρτης πορείας ενός προϊόντος το οποίο υλοποιείται σε ένα έργο που περιέχει 12 sprints και έχει τέσσερις αποδεσμεύσεις (releases).

[4] Το μότο είναι μια σύντομη πρόταση ή φράση που έχει επιλεγεί για να εκφράσει εν συντομία τις πεποιθήσεις ή τα ιδανικά ενός ατόμου, μιας ομάδας, μιας οικογένειας ή ενός οργανισμού.

Εικόνα 4.15: *Παράδειγμα χάρτη πορείας προϊόντος*

Διαχείριση του καταλόγου των απαιτήσεων του προϊόντος (product backlog)

Μία από τις πιο σημαντικές ευθύνες για έναν ιδιοκτήτη προϊόντος Scrum είναι η διαχείριση των ανεκτέλεστων απαιτήσεων του προϊόντος. Η ευθύνη του ιδιοκτήτη του προϊόντος είναι να δημιουργήσει τον κατάλογο με τις ιστορίες χρηστών που έχουν αξία για την επιχείρηση ή έχουν καθυστερήσει και να τους δώσει προτεραιότητα με βάση τη συνολική στρατηγική και τους επιχειρηματικούς στόχους. Επιπλέον, ο ιδιοκτήτης του προϊόντος θα πρέπει να χαρτογραφήσει τις εξαρτήσεις του έργου (dependencies), ώστε να προκύψει η σωστή ακολουθία ανάπτυξης.

Ωστόσο, ο κατάλογος απαιτήσεων δεν είναι μια στατική λίστα. Είναι ένα ζωντανό έγγραφο που αλλάζει και πρέπει να ενημερώνεται συνεχώς με βάση τις εξελισσόμενες ανάγκες του έργου σε όλη την περίοδο ανάπτυξης.

Επειδή ο κατάλογος απαιτήσεων προϊόντος αλλάζει συχνά, ο ιδιοκτήτης προϊόντος πρέπει να κάνει τον κατάλογο προσβάσιμο και διαθέσιμο σε όλους τους συμμετέχοντες (ιδιαίτερα στην ομάδα έργου), για να εξασφαλίσει καλύτερη απόδοση της ομάδας έργου. Για το λόγο αυτό η ανάπτυξη λογισμικού σήμερα βασίζεται στη χρήση συνεργατικών εργαλείων ανάπτυξης (teamworking/collaboration tools).

Η ιεράρχηση των αλλαγών (prioritization)

Η ιεράρχηση προτεραιοτήτων είναι μία από τις κύριες έννοιες που παρουσιάζονται στις ευέλικτες πρακτικές. Η έννοια και η δραστηριότητα της ιεράρχησης ισοδυναμεί με τη λήψη απόφασης για τη βέλτιστη ταξινόμηση των απαιτήσεων ή των ιστοριών χρηστών με τη χρήση κριτηρίων, συνήθως βάση της επιχειρηματικής αξίας. Η κατανόηση της ιεράρχησης είναι απαραίτητη για όλα τα έργα, αλλά καθίσταται ιδιαίτερα σημαντική στις ευέλικτες μεθόδους, διότι η ομάδα έργου θα πρέπει να παραδίδει μια νέα έκδοση λογισμικού σε τακτά χρονικά διαστήματα (π.χ. κάθε μήνα). Δεν πρέπει να ξεχνάμε ότι ένα ευέλικτο έργο είναι χρονομετρημένο και με σταθερό σύνολο πόρων, και συνεπώς η ιεράρχηση είναι απολύτως αναγκαία προκειμένου να ικανοποιηθούν οι χρονικοί και οι περιορισμοί πόρων.

Επιπλέον, η διαδικασία ιεράρχησης βοηθά την ευέλικτη ομάδα να εξετάσει και να εντοπίσει τα ελάχιστα χαρακτηριστικά του προϊόντος που είναι απαραίτητα για τη δημιουργία αξίας πελάτη. Συνεπώς, πριν θέσουμε τις προτεραιότητες στις απαιτήσεις ενός προϊόντος, είναι απαραίτητο να κατανοήσουμε καλά τους παράγοντες που σχετίζονται με την επιχειρηματική αξία καθώς και με ένα επιτυχημένο τελικό αποτέλεσμα.

Ο ιδιοκτήτης του προϊόντος, με άλλα λόγια, θα πρέπει να βρει το σημείο ισορροπίας μέσα στο «σιδερένιο τρίγωνο» (iron triangle) των περιορισμών του έργου, που όπως ήδη έχουμε προαναφέρει έχει ως πλευρές το κόστος, το εύρος και τον χρόνο που απαιτεί ένα έργο.

Για παράδειγμα, εάν το υπό ανάπτυξη προϊόν πρέπει να κυκλοφορήσει στην αγορά εντός έξι μηνών, αυτό σημαίνει ότι θα πρέπει να υλοποιήσουμε το εύρος του έργου, υλοποιώντας αυτές τις ιστορίες χρηστών με την μεγαλύτερη επιχειρηματική αξία. Η ιεράρχηση των απαιτήσεων γίνεται χρησιμοποιώντας ειδικές μεθόδους, οι όπως θα παρουσιαστούν στο Κεφάλαιο 7, όπως:

- Η μέθοδος MoSCoW
- Ταξινόμηση (ranking)
- Analytical Hierarchical Process (AHP)
- Το μοντέλο Kano
- Μέθοδος στάθμισης με βάρη
- Η μέθοδος των 100 δολαρίων
- Κ.λπ.

Επίβλεψη της υλοποίησης του έργου

Εκτός από την ανάπτυξη του οράματος του προϊόντος, του product backlog και της ιεράρχησης των προτεραιοτήτων, ο ιδιοκτήτης του προϊόντος θα πρέπει να αφιερώσει σημαντικό χρόνο στην επίβλεψη της πραγματικής ανάπτυξης του προϊόντος. Είναι κύριος συμμετέχων σε κάθε εκδήλωση, γεγονός του έργου, συμπεριλαμβανομένου του σχεδιασμού του sprint, της ανασκόπησης του sprint, της αποδοχής του προϊόντος, κ.λπ.

Επίσης, ο ιδιοκτήτης του προϊόντος διασφαλίζει ότι όλοι οι στόχοι που πρέπει να επιτευχθούν σε ένα sprint έχουν επιτευχθεί φροντίζοντας να πληρούνται τόσο οι μη λειτουργικές όσο και οι λειτουργικές απαιτήσεις από την ομάδα ανάπτυξης. Ο ιδιοκτήτης του προϊόντος ορίζει τα κριτήρια αποδοχής για κάθε ιστορία χρήστη και για κάθε sprint. Τα κριτήρια αποδοχής είναι ζωτικής σημασίας για την ομάδα του Scrum, επειδή είναι αυτά με τα οποία καθορίζεται η πρόοδος του έργου. Χωρίς αυτά, η ομάδα ανάπτυξης δεν θα ήταν σε θέση να ορίσει πότε μια εργασία ή μια ιστορία χρήστη έχει ολοκληρωθεί, ή να κατανοήσει ποιες είναι οι καλές πρακτικές που χρησιμοποιεί η ομάδα.

Οικονομική διαχείριση του έργου

Μια από τις βασικές δραστηριότητες ενός διαχειριστή έργου είναι η οικονομική διαχείριση. Η οικονομική διαχείριση του έργου γίνεται σε α) επίπεδο αποδέσμευσης (release), β) σε επίπεδο sprint και γ) σε επίπεδο καταλόγου απαιτήσεων έργου (product backlog).

Στην οικονομική διαχείριση σε επίπεδο έκδοσης ο ιδιοκτήτης προϊόντος θα πρέπει να αποφασίσει για την κατάλληλη χρονική στιγμή της αποδέσμευσης της έκδοσης. Για παράδειγμα, αν διαπιστώσει ότι η ομάδα έργου μπορεί να παράγει ένα προϊόν, με επιπλέον χαρακτηριστικά, που μπορεί να δημιουργήσουν επιπλέον έσοδα, εάν δουλέψει μια επιπλέον εβδομάδα, τότε πρέπει να αντισταθμίσει τα προβλεπόμενα έσοδα με τα αναμενόμενα έξοδα και να αποφασίσει για την τελική ημερομηνία κυκλοφορίας του προϊόντος.

Επίσης, μπορεί να αποφασίσει να μην χρηματοδοτήσει μια επιπλέον εβδομάδα ανάπτυξης, εάν η εργασία που προγραμματίζεται να γίνει σε αυτό το χρονικό πλαίσιο, δεν δημιουργεί επιπλέον επιχειρηματική αξία. Μπορεί επίσης να αποφασίσει ότι οι στόχοι στους οποίους πρέπει να εργαστεί η ομάδα έργου πρέπει να αλλάξουν ή ότι ολόκληρη η ομάδα πρέπει να σταματήσει την παραγωγή λόγω προβλημάτων που αντιμετωπίζουν άλλοι συμμετέχοντες.

Στόχος της οικονομικής διαχείρισης σε επίπεδο sprint είναι να διασφαλιστεί ότι υπάρχει ικανοποιητική απόδοση επένδυσης (ROI – Return Of Investment). Η απόδοση επένδυσης (ROI) είναι ένα μέτρο απόδοσης που χρησιμοποιείται για την αξιολόγηση της αποδοτικότητας ή της κερδοφορίας μιας επένδυσης ή για τη

σύγκριση της αποτελεσματικότητας πολλών διαφορετικών επενδύσεων. Η απόδοση επένδυσης (ROI) προσπαθεί να μετρήσει άμεσα το ποσό απόδοσης μιας συγκεκριμένης επένδυσης, σε σχέση με το κόστος της επένδυσης. Ο υπολογισμός της ROI σε επίπεδο sprint μπορεί να είναι σύνθετη ή/και ανακριβής διότι είναι δύσκολο να προσδιορίσουμε με ακρίβεια για κάθε συγκεκριμένη ιστορία χρήστη που περιλαμβάνει το sprint την επιμέρους αξία που προσθέτει αυτή στην αξία του συνολικού προϊόντος. Γενικότερα η αναμενόμενη απόδοση επένδυσης για ένα sprint προσδιορίζεται ως εξής:

```
ROI = (αριθμός χρηστών * ποσοστό νέων χρηστών/αύξησης χρήσης
      * επιχειρηματική αξία sprint) – κόστος ανάπτυξης
```

Σε αυτόν τον τύπο υπάρχουν δύο κεντρικά ποσά: α) τα επιπλέον έσοδα που είναι ένας εξωτερικός παράγοντας και δημιουργούνται από την νέα αποδέσμευση του λογισμικού και β) τα έξοδα είναι εσωτερικός παράγοντας και αφορούν το λειτουργικό κόστος της ομάδας έργου.

Η οικονομική διαχείριση σε επίπεδο καταλόγου απαιτήσεων έργου (product backlog) σχετίζεται με την ιεράρχηση των απαιτήσεων καθώς πάντα θα πρέπει να προηγούνται οι ιστορίες χρηστών με μεγαλύτερη επιχειρηματική αξία.

Τα λάθη στην εφαρμογή του ρόλου

Η εφαρμογή του ρόλου του ιδιοκτήτη προϊόντος μπορεί να οδηγήσει σε λάθη και προβλήματα. Στις επόμενες παραγράφους θα παρουσιάσουμε μερικά από τα πιο συνηθισμένα λάθη (Pichler, 2010).

Ο ιδιοκτήτης προϊόντος **χωρίς ικανή επιρροή**. Αυτό μπορεί να έχει διάφορες αιτίες όπως:

- ο ιδιοκτήτης του προϊόντος δεν έχει αρκετή επιρροή στη διοίκηση, η χρηματοδότηση του έργου προέρχεται από λάθος διοικητικό επίπεδο ή λάθος άτομο,
- η διοίκηση δεν εμπιστεύεται πλήρως τον ιδιοκτήτη του προϊόντος ή
- δεν έχει την αναγκαία δικαιοδοσία στη λήψη αποφάσεων.

Στην περίπτωση αυτή υπάρχουν καθυστερήσεις αφού ο ιδιοκτήτης προϊόντος πρέπει να συμβουλεύεται την διοίκηση για κάθε απόφαση, ενώ δεν μπορεί να αναλάβει εξ' ολοκλήρου την ευθύνη.

Ο ιδιοκτήτης προϊόντος που έχει **υπερβολικό φόρτο εργασίας**. Οι ιδιοκτήτες προϊόντων με υπερβολικό φόρτο εργασίας γίνονται γρήγορα πρόβλημα για την πρόοδο του έργου. Συμπτώματα καταπόνησης και υπερβολικού φόρτου εργασίας περιλαμβάνουν την παραμέληση εργασιών, όπως τη διαχείριση του product backlog, τις καθυστερήσεις ή απουσία από τις συναντήσεις του έργου, την καθυστέρηση απόκρισης σε ερωτήματα ή αντιμετώπισης προβλημάτων. Δύο είναι οι βασικές αιτίες της υπερβολικής εργασίας από τον ιδιοκτήτη του προϊόντος:

- δεν υπάρχει αρκετός χρόνος για να εκτελέσει όλες τις εργασίες που προβλέπει ο ρόλος είτε
- δεν υπάρχει αρκετή υποστήριξη από την ομάδα έργου.

Για να επίλυση του προβλήματος θα πρέπει

- να εστιαστεί ο εργαζόμενος στο έργο αποφεύγοντας άλλες πιθανές αναθέσεις και εργασίες, θεωρώντας ότι ο ρόλος είναι εργασία πλήρους απασχόλησης, και
- να εξασφαλίσουμε ότι η ομάδα έργου αφιερώνει χρόνο σε κάθε sprint για τη συνεργασία με τον ιδιοκτήτη του προϊόντος. Η μέθοδος Scrum προβλέπει ότι το 10% του συνολικού χρόνου της ομάδας σε κάθε sprint θα πρέπει να αφιερώνεται στην υποστήριξη του ιδιοκτήτη του προϊόντος (Schwaber, 2007).

Ορισμένοι οργανισμοί **χωρίζουν τον ρόλο** του ιδιοκτήτη προϊόντος διανέμοντας τα καθήκοντα σε πολλά άτομα, για παράδειγμα, χρησιμοποιώντας ένα ρόλο για το διευθυντή προϊόντος (product manager) και ένα ρόλο για τον ιδιοκτήτη προϊόντος. Ο διευθυντής προϊόντος είναι υπεύθυνος για το μάρκετινγκ και τις πωλήσεις του προϊόντος και είναι αυτός που διαμορφώνει το όραμα του προϊόντος σε σχέση πάντα με την αγορά. Από την άλλη πλευρά ο ιδιοκτήτης προϊόντος έχει εσωτερικό προσανατολισμό, είναι αυτός που

κατέχει το product backlog, ορίζει το sprint backlog και συνεργάζεται με την ομάδα έργου. Αυτή η προσέγγιση δημιουργεί εμπόδια, αφού η ευθύνη διαχέεται προκαλώντας καθυστερήσεις. Στην περίπτωση αυτή η επιχείρηση θα πρέπει να φροντίσει για τη σωστή εφαρμογή του ρόλου κάνοντας οργανωτικές αλλαγές ώστε ένα άτομο να είναι υπεύθυνο τόσο για τη στρατηγική όσο και την τακτική διαχείριση του προϊόντος. Υπάρχουν ακόμη περιπτώσεις που ο ρόλος ανατίθεται σε μια επιτροπή. Η **επιτροπή ιδιοκτητών προϊόντος** είναι μια ομάδα ατόμων όπου κανείς δεν είναι υπεύθυνος για το συνολικό προϊόν. Συνεπώς, δεν υπάρχει ένα άτομο υπεύθυνο για την καθοδήγηση της ομάδας, τη δημιουργία ενός κοινού στόχου και τη διευκόλυνση στη λήψη απόφασης. Η ύπαρξη μιας τέτοιας επιτροπής αυξάνει τον αναγκαίο χρόνο για συναντήσεις, την επίλυση διαφορών που προκύπτουν από συγκρουόμενα συμφέροντα και πολιτικές, μείωση της συνεργασίας, κ.λπ. και η ύπαρξή της θεωρείται ιδιαίτερα αρνητικό γεγονός.

Ο **απομακρυσμένος ιδιοκτήτης προϊόντος** που εργάζεται σε απόσταση από την ομάδα έργου. Η απόσταση μπορεί να υπάρχει με διάφορες μορφές, π.χ. σε άλλο χώρο, πόλη, ήπειρο. Αν και τα τελευταία χρόνια, η εξ' αποστάσεως εργασία είναι κάτι πολύ συνηθισμένο, η πιο αποτελεσματική μορφή επικοινωνίας μέσα σε μια ομάδα είναι η επικοινωνία πρόσωπο με πρόσωπο. Εάν η συνεγκατάσταση δεν είναι μια διαθέσιμη επιλογή, ο ιδιοκτήτης του προϊόντος θα πρέπει να αφιερώνει όσο το δυνατόν περισσότερο χρόνο με την υπόλοιπη ομάδα του Scrum. Αποτελεί καλή πρακτική οι ιδιοκτήτες προϊόντων να είναι στον ίδιο χώρο με την ομάδα έργου σε όλες τις συναντήσεις σχεδιασμού, αξιολόγησης ή ανασκόπησης.

Συμπερασματικά θα μπορούσαμε να πούμε λοιπόν για τον ιδιοκτήτη του προϊόντος ότι:

- Είναι υπεύθυνος για τη δημιουργία οράματος προϊόντος
- Είναι υπεύθυνος για τη διαχείριση του Product Backlog
- Είναι υπεύθυνος για την ιεράρχηση των απαιτήσεων
- Είναι υπεύθυνος για την επίβλεψη της ανάπτυξης λογισμικού
- Είναι υπεύθυνος για την προσαρμογή τους προϊόντος στις μεταβαλλόμενες ανάγκες
- Αποτελεί τον συνδετικό ιστό μεταξύ πελάτη και ομάδας ανάπτυξης
- Είναι υπεύθυνος για την αξιολόγηση της προόδου στο τέλος κάθε sprint
- Κάνει την οικονομική διαχείριση του έργου

4.2.4 Ο πελάτης (customer)

Ο πελάτης συμμετέχει ενεργά σε όλη την διαδικασία ανάπτυξης του λογισμικού, ιδιαίτερα στην καταγραφή των απαιτήσεων, στη διαδικασία επιλογής και ιεράρχησης αυτών, καθώς και στον έλεγχο της καλής λειτουργίας του τελικού προϊόντος. Συνήθως ο πελάτης είναι ταυτόχρονα και ο χρήστης του λογισμικού, οπότε, εκτός από τις γνώσεις του όσον αφορά το υπό ανάπτυξη προϊόν, μπορεί άμεσα να αναγνωρίσει το παραγόμενο αποτέλεσμα και να ζητήσει την τροποποίησή του όποτε χρειασθεί. Ο πελάτης βρίσκεται σε διαρκή επικοινωνία με την ομάδα έργου, πολλές φορές εργάζεται μαζί με την ομάδα έργου και απαντάει άμεσα στις ερωτήσεις της ομάδας αυξάνοντας έτσι την παραγωγικότητά της.

4.2.5 Η διοίκηση (administration)

Η διοίκηση είναι υπεύθυνη για τη λήψη της τελικής απόφασης καθώς και για τα πρότυπα και τις συμβάσεις που ακολουθούνται στο έργο. Συμμετέχει ενεργά στον καθορισμό των στόχων και των απαιτήσεων του έργου, ενώ επίσης είναι υπεύθυνη για την επιλογή του product owner καθώς και για την επίλυση των υψηλού επιπέδου προβλημάτων.

4.3 Τα τεχνουργήματα της διεργασίας Scrum

4.3.1 Ο κατάλογος των απαιτήσεων του προϊόντος (product backlog)

Το μεγαλύτερο ερώτημα που υπάρχει στην αρχή κάθε έργου είναι, τι ακριβώς φτιάχνουμε; Συνήθως γνωρίζουμε τη γενική μορφή του συστήματος που θα κατασκευαστεί. Δηλαδή γνωρίζουμε, για παράδειγμα, ότι κατασκευάζουμε έναν επεξεργαστή κειμένου. Ωστόσο, πάντα υπάρχουν σκοτεινά σημεία που θα πρέπει να διερευνήσουμε ή ζητήματα που πρέπει να διευθετηθούν σχετικά με το πώς θα λειτουργήσει το σύστημα σε επιμέρους λειτουργίες. Για παράδειγμα, θα υποστηρίζει ο επεξεργαστής κειμένου δυνατότητες χρήσης μέσω κινητών συσκευών, ή συνεργατικής γραφής;

Όταν χρησιμοποιούμε μια διεργασία ανάπτυξης μορφής καταρράκτη, ξεκινάμε με την παραδοχή ότι θα συγκεντρώσουμε όλες τις απαιτήσεις στην αρχή του έργου και ότι θα προσδιορίσουμε το προϊόν πλήρως. Η βασική ιδέα είναι ότι, αν σχεδιάσουμε και καταγράψουμε καλύτερα τις απαιτήσεις στην αρχή του έργου, δεν θα συναντήσουμε σκοτεινά σημεία κατά τη διάρκεια της κύριας φάσης ανάπτυξης του έργου. Αντίθετα, στην ευέλικτη ανάπτυξη παραιτούμαστε από την ψευδαίσθηση ότι όλα μπορούν να καταγραφούν επακριβώς στην αρχή προς όφελος μιας προσέγγισης ότι θα καταγράψουμε ό,τι είναι εφικτό και σε υψηλό επίπεδο και στη συνέχεια θα περιγράψουμε τις λεπτομέρειες σταδιακά καθώς προχωρά το έργο. Επιπλέον, τεκμηριώνουμε σε ένα κατάλογο όλες τις επιθυμητές λειτουργίες που δεν υπάρχουν ακόμη στο προϊόν (product backlog), ώστε να γνωρίζουμε τη γενική κατεύθυνση που πρέπει να ακολουθήσουμε. Σε αντίθεση με ένα παραδοσιακό έγγραφο απαιτήσεων, μια λίστα με τα ανεκτέλεστα χαρακτηριστικά του προϊόντος υπό κατασκευή είναι εξαιρετικά δυναμική. Νέα στοιχεία προστίθενται, υπάρχοντα στοιχεία τροποποιούνται ή αφαιρούνται, δίνουμε προτεραιότητες για την υλοποίηση αυτών, ενώ ταυτόχρονα μαθαίνουμε περισσότερα για το προϊόν, τους χρήστες του, την ομάδα και ούτω καθεξής.

Η δημιουργία του product backlog ξεκινά στη φάση της αρχικής διερεύνησης (pre-game) με τη δημιουργία οράματος και ενός οδικού χάρτη για το προϊόν υπό κατασκευή. Η δημιουργία ενός οδικού χάρτη προϊόντος είναι ένας συνηθισμένος τρόπος ώστε όλοι οι συμμετέχοντες του έργου να αποκτήσουν κοινή αντίληψη για το προϊόν (βλέπε Εικόνα 4.14).

Τα βήματα για τη δημιουργία του οδικού χάρτη του προϊόντος είναι :

- 1. Να αποφασίσουμε για το είδος και τα χαρακτηριστικά του προϊόντος (το όραμα).
- 2. Να σχεδιάσουμε τον οδικό χάρτη του προϊόντος.
- 3. Να επικοινωνήσουμε το σχέδιο στους συμμετέχοντες.

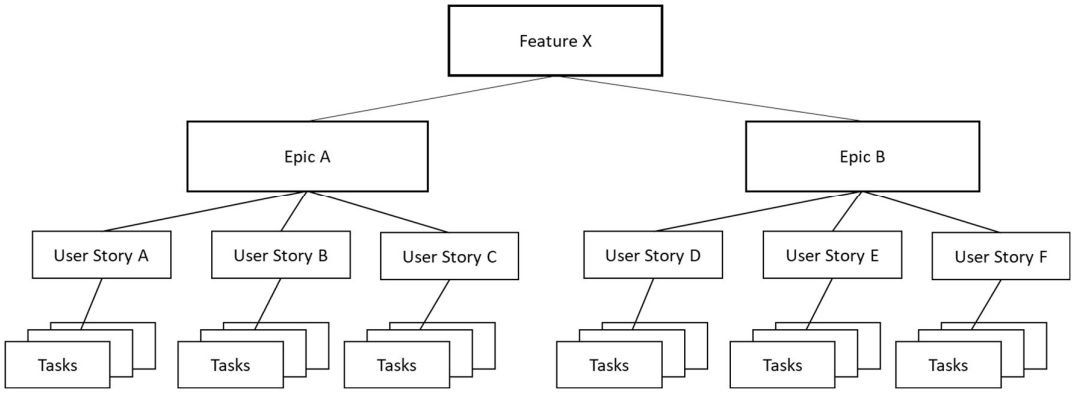
Ο οδικός χάρτης προϊόντος είναι μια ολιστική, υψηλού επιπέδου προβολή των χαρακτηριστικών που είναι απαραίτητες για την επίτευξη του στόχου προϊόντος. Η ύπαρξή του μας βοηθά να βρούμε εξαρτήσεις μεταξύ δραστηριοτήτων («Αν το κάνουμε αυτό, θα πρέπει λογικά να το κάνουμε αυτό») καθώς και τα κενά στα χαρακτηριστικά γίνονται εύκολα ορατά («Πού είναι η λειτουργία..;»). Επίσης, ο οδικός χάρτης προϊόντος είναι το αρχικό ανεκτέλεστο προϊόν (η κύρια λίστα των εργασιών που θα πρέπει να εκτελέσουμε) και καθώς προχωράμε στο έργο, τα επιμέρους στοιχεία αναλύονται όλο και περισσότερο (προοδευτική επεξεργασία), ενώ συμπεριλαμβάνουμε στοιχεία διαφορετικών μεγεθών:

- **Ιστορίες χρηστών (user stories)** που αποτελούν μικρές αλλά διακριτές λειτουργίες του συστήματος
- **Επικά (epics)**, ομάδες συναφών ιστοριών χρηστών, μεσαίας έκτασης
- **Χαρακτηριστικά (features)** είναι ένα σύνολο χαρακτηριστικών/λειτουργικότητας του συστήματος μεγάλης έκτασης.
- **Θέματα (themes)** είναι ένα σύνολο λειτουργιών του προϊόντος που επιτρέπει στον χρήστη να κάνει μια διακριτή εργασία.

Η ορολογία που χρησιμοποιείται για τον οδικό χάρτη προϊόντος (product roadmap) αλλά και για τον κατάλογο απαιτήσεων προϊόντος είναι η ακόλουθη (product backlog). Οι ακόλουθοι ορισμοί σχετίζονται με τους οδικούς χάρτες προϊόντων και τους καταλόγους απαιτήσεων προϊόντων. Δεν αποτελούν μέρος της μεθόδου Scrum, αλλά αποτελούν μέρος μιας συλλογής κοινών πρακτικών που χρησιμοποιούνται πολύ συχνά:

- **Θέματα**: Τα θέματα είναι λογικές ομαδοποιήσεις των δυνατοτήτων του προϊόντος. Εάν ένα χαρακτηριστικό είναι μια νέα δυνατότητα που έχει ένας πελάτης, ένα θέμα είναι η λογική ομαδοποίηση αυτών των χαρακτηριστικών. Είναι η λειτουργικότητα που δίνει τη δυνατότητα στον πελάτη να επιλέξει ένα προϊόν σε σχέση με ένα άλλο. Ο οδικός χάρτης του προϊόντος προσδιορίζει ή/και ομαδοποιεί χαρακτηριστικά ως μέρος ενός θέματος. Για παράδειγμα η δυνατότητα μιας εφαρμογής κοινωνικών δικτύων να διαμοιράζεται αρχεία με άλλες εφαρμογές είναι ένα θέμα.
- **Χαρακτηριστικά**: Τα χαρακτηριστικά είναι δυνατότητες που θα έχουν οι πελάτες μετά την ανάπτυξη προϊόντος. Ο οδικός χάρτης του προϊόντος συνήθως αποτελείται από απαιτήσεις σε επίπεδο χαρακτηριστικών. Για παράδειγμα η δυνατότητα μιας εφαρμογής κοινωνικών δικτύων να διαμοιράζεται αρχεία με την εφαρμογή του messenger είναι ένα χαρακτηριστικό. Εναλλακτικά χρησιμοποιείται και ο όρος initiative (πρωτοβουλία).
- **Epics:** Το epic είναι ένα σύνολο εργασίας συναφούς περιεχομένου που σχετίζεται με ένα χαρακτηριστικό και μπορεί να διασπαστεί περεταίρω σε ιστορίες χρηστών. Η λειτουργικότητα που επιτρέπει το διαμοιρασμό ενός αρχείου αφού επιλέξουμε το είδος του αρχείου που θέλουμε. Το epic είναι η μεγαλύτερη μονάδα λειτουργικότητας που μπορούμε να συμπεριλάβουμε σε ένα sprint.
- **Ιστορίες χρηστών:** Οι ιστορίες χρηστών είναι οι μικρότερες μορφές απαιτήσεων που μπορούν να υλοποιηθούν. Μια ιστορία χρήστη αποτελείται από μια ενέργεια που δημιουργεί ή ενσωματώνει αξία για ένα τελικό χρήστη. Για παράδειγμα, η αγορά ενός προϊόντος μέσω κινητού τηλεφώνου από καλάθι αγορών με χρήση κάρτας Visa είναι μια ιστορία χρήστη. Η αγορά ενός αντικειμένου χρησιμοποιώντας μια MasterCard είναι μια διαφορετική ιστορία χρήστη. Οι ιστορίες χρηστών πρέπει να είναι αρκετά μικρές για να προστεθούν στα sprint και να αρχίσουν να αναπτύσσονται. Σε επόμενο κεφάλαιο θα αναφερθούμε αναλυτικά στις ιστορίες χρηστών, στον τρόπο συγγραφής καθώς και στις καλές πρακτικές που αφορούν την ανάπτυξή τους.
- **Εργασίες**: Οι εργασίες είναι τα βήματα που απαιτούνται για την υλοποίηση μιας ιστορίας χρήστη. Κατά τον προγραμματισμό των sprint, μια ιστορία χρήστη χωρίζεται σε εργασίες.

Στην Εικόνα 4.16 παρουσιάζουμε την ιεραρχία αποδόμησης του προϊόντος σε επιμέρους στοιχεία μέχρι να φτάσουμε στις ιστορίες χρηστών.



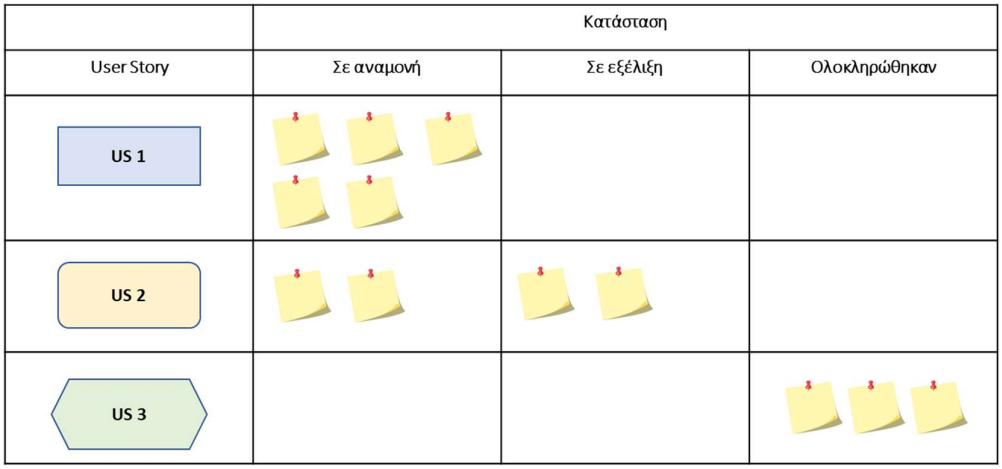
Εικόνα 4.16: *Ιεραρχία οργάνωσης καταλόγου απαιτήσεων*

- Όπως ήδη έχουμε αναφέρει το product backlog είναι ένα τεχνούργημα της μεθόδου Scrum και είναι η κύρια λίστα υποχρεώσεων για ολόκληρο το έργο. Όλα τα έργα Scrum έχουν ένα product backlog. Το product backlog ανήκει και συντηρείται από τον ιδιοκτήτη του προϊόντος.
- Οι απαιτήσεις του οδικού χάρτη προϊόντος αποτελούν την πρώτη έκδοση του product backlog. Στην έναρξη του έργου, οι απαιτήσεις υψηλότερης προτεραιότητας αναλύονται ώστε να δημιουργηθούν οι πρώτες ιστορίες χρηστών. Στην Εικόνα 4.17 παρουσιάζεται ένα παράδειγμα product backlog από μια εφαρμογή δεδομένων COVID-19.

PRODUCT BACKLOG					
Α/Α	ΚΩΔ	ΑΠΑΙΤΗΣΗ	ΠΕΡΙΓΡΑΦΗ ΑΠΑΙΤΗΣΗΣ	ΕΙΔΟΣ	ΕΚΤΙΜΩΜΕΝΗ ΠΡΟΣΠΑΘΕΙΑ USER STORY POINTS
1	PR1	Οργάνωση ομάδας	Επικοινωνία και επιλογή εργαλείων επικοινωνίας	OVERHEAD	0,25
2	PR3	Εγκατάσταση εργαλειων για την υλοποίηση του έργου	Δοκιμή εργαλείων και Σύνδεση με Βιβλιοθήκες	OVERHEAD	0,5
3	PR3	Δημιουργία Βάσης Δεδομένων , Δημιουργία POJO κλάσεων και Δημιουργία JPA controllers	Σύνδεση, Δημιουργία βάσης και Δημιουργία πινάκων	OVERHEAD	1
4	R1	Δημιουργία GUI (αρχική οθόνη, μενού)	Δημιουργία Οθονών και Δημιουργία Menu	ΑΠΑΙΤΗΣΗ	0,25
5	R2.1	Οθόνη διαχείρισης δεδομένων Εισαγωγή (επέκταση GUI)	Λίστα επιλογής κατηγορίας δεδομένων, Πλήκτρα εισαγωγή χωρών - Εισαγωγή δεδομένων, Κλήση API και Ανάκτηση δεδομένων από Json, Buiding της ΒΔ με το GUI και Εισαγωγή δεδομένων στη ΒΔ	ΑΠΑΙΤΗΣΗ	4
6	R2.2	Οθόνη διαχείρισης δεδομένων Διαγραφή (επέκταση GUI)	Διαγραφή δεδομένων, Διαγραφή χωρών, Μήνυμα επιβεβαίωσης Διαγραφής	ΑΠΑΙΤΗΣΗ	1
7	R3.1	Οθόνη προβολής δεδομένων χώρας (επέκταση GUI)	Διασύνδεση με τη ΒΔ, Επιλογή από λίστα χώρας, Ανάκτηση δεδομένων σε πίνακα (3 σελίδες/κατηγορία), Πεδία Ημερομηνίας από-εως, Πλήκτρο Προβολή σε χάρτη, Πλήκτρο Προβολή σε Διάγραμμα, Διαγραφή Χώρας, Binding της ΒΔ με το GUI και Διαγραφή δεδομένων	ΑΠΑΙΤΗΣΗ	4
8	R3.2	Οθόνη προβολής δεδομένων σε διάγραμμα (επέκταση GUI)	Εμφάνιση Δεδομένων σε Διάγραμμα , Επιλογή μιας καμπύλης από τρείς/ εμφάνιση σωρευτικών δεδομένων	ΣΥΝΤΗΡΗΣΗ	4
9	R3.3	Οθόνη προβολής δεδομένων χώρας σε χάρτη (επέκταση GUI)	Εμφάνιση των Δεδομένων της Επιλεχθήσας Χώρας σε Χάρτη	ΑΠΑΙΤΗΣΗ	4
10	R4	Οθόνη προβολής δεδομένων χωρών σε χάρτη (επέκταση GUI)	Πεδίο βασικής επιλογής χώρας,Πεδίο πολλαπλής επιλογής χωρών, Πεδία ημερομηνίας από-εώς, Εμφάνιση στο χάρτη με Mark, Pop-up με εμφάνιση δεδομένων, Binding της ΒΔ με το GUI	ΑΠΑΙΤΗΣΗ	2
				ΣΥΝΟΛΑ	21

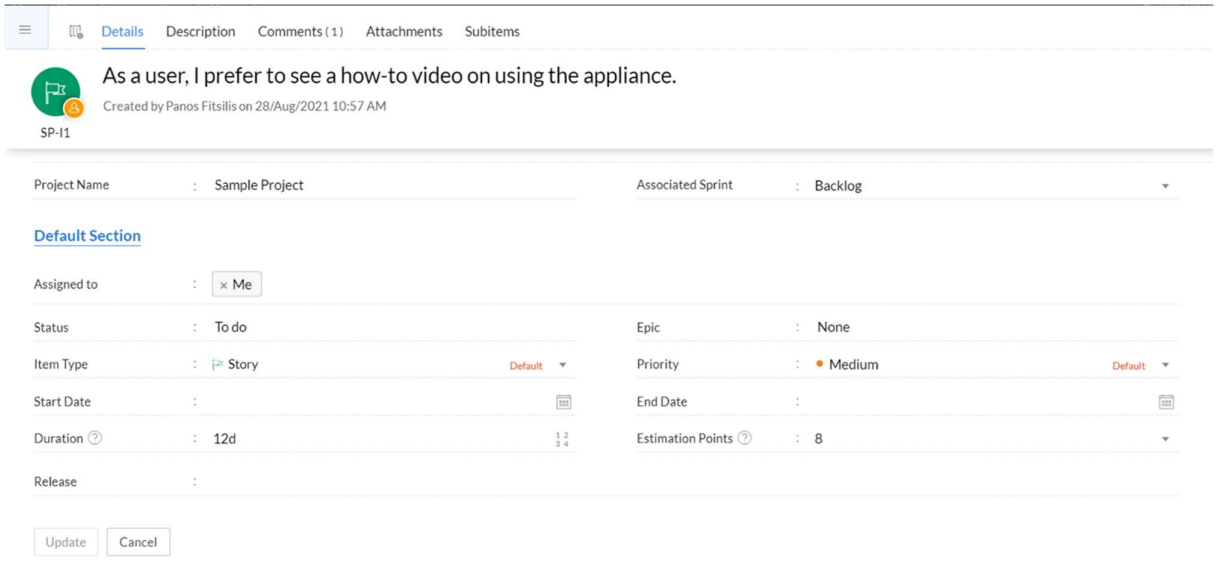
Εικόνα 4.17: *Παράδειγμα Product Backlog*

Το product backlog είναι μια φυσική αναπαράσταση των εργασιών (κάθε εργασία αποτελείται από επιμέρους εργασίες που αναπαρίστανται με ένα χαρτί τύπου post-it) που η ομάδα θα υλοποιήσει κατά τη διάρκεια των sprint (βλέπε Εικόνα 4.18). Στην πρώτη στήλη στον πίνακα βρίσκονται οι εργασίες που δεν έχουν ξεκινήσει ακόμη (waiting ή to do), στη δεύτερη στήλη οι εργασίες που βρίσκονται σε εξέλιξη (Work In Progress - WIP), ενώ στην τρίτη στήλη είναι οι εργασίες που έχουν ολοκληρωθεί (done) (Kniber, 2015).



Εικόνα 4.18: *Πίνακας ανεκτέλεστων εργασιών*

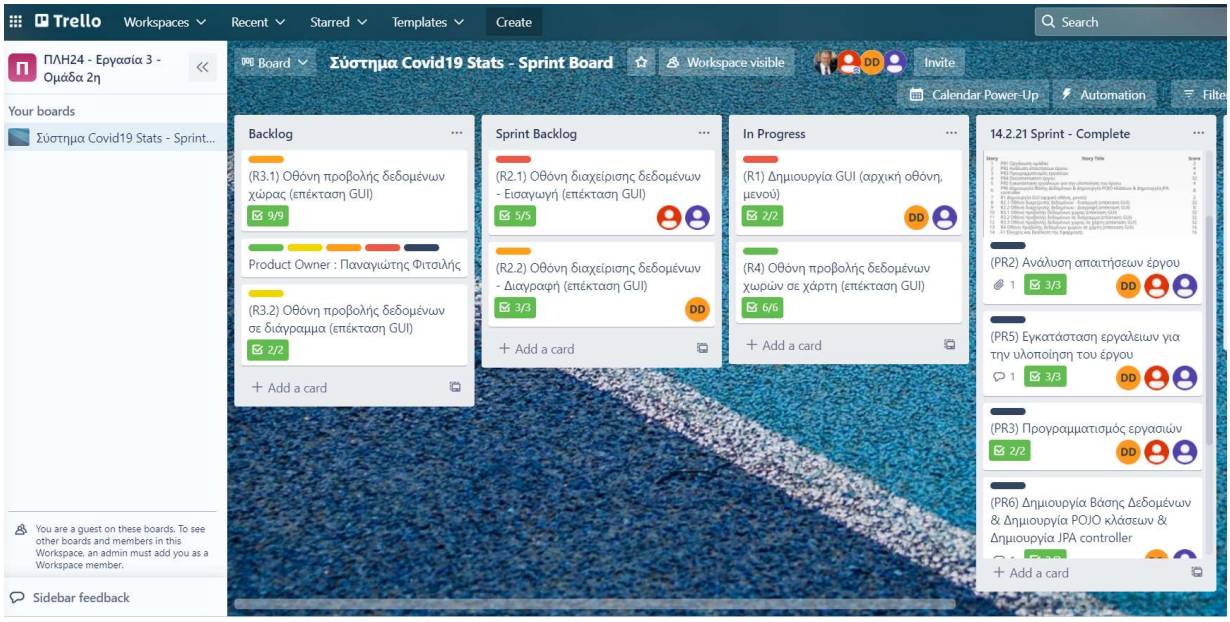
- Στην Εικόνα 4.19 παρουσιάζεται μια σειρά χαρακτηριστικών για κάθε στοιχείο του product backlog. Συνήθως η διαχείριση του product backlog αν και ξεκίνησε με τη χρήση post-it που η ομάδα ανάπτυξης κολλούσε σε ένα πίνακα ή τοίχο, σήμερα γίνεται με χρήση αυτοματοποιημένων συνεργατικών εργαλείων. Σε κάθε περίπτωση στο product backlog διατηρούμε ένα σύνολο στοιχείων τα πιο βασικά από τα οποία είναι:
- Η αρίθμηση των απαιτήσεων που συνήθως είναι ένα κωδικός ανά απαίτηση
- Ο τίτλος της κάθε απαίτησης
- Η περιγραφή της απαίτησης. Ο πιο διαδεδομένος τρόπος περιγραφής των απαιτήσεων στις ευέλικτες μεθόδους είναι οι ιστορίες χρηστών (user stories). Θα αναφερθούμε διεξοδικά σε αυτό στο κεφάλαιο 7.
- Το είδος της απαίτησης. Το είδος της απαίτησης μπορεί να είναι:
    - Ιστορία χρήστη/ Περίπτωση Χρήσης (use case)/απαίτηση σε απλό κείμενο
    - Εργασία που αφορά επίλυση σφάλματος (bug/maintenance)
    - Τεχνική εργασία που αφορά την εργασία της ομάδας (overhead), όπως για παράδειγμα την εγκατάσταση μιας νέας έκδοσης μια βάσης δεδομένων, ενός περιβάλλοντος ανάπτυξης, κ.λπ.
    - Μελέτη με σκοπό την απόκτηση γνώσης που η ομάδα ανάπτυξης χρειάζεται τεχνογνωσία.
- Την εκτίμηση της απαιτούμενης προσπάθειας. Η εκτίμηση της αναγκαίας προσπάθειας συνήθως γίνεται με χρήση αντίστοιχης μεθόδου (π.χ. planning poker). Για τον τρόπο εκτίμησης της προσπάθειας θα αναφερθούμε αναλυτικά στο κεφάλαιο 7.
- Την προτεραιότητα της απαίτησης. Η ιεράρχηση των απαιτήσεων συνήθως γίνεται με χρήση αντίστοιχης μεθόδου (π.χ. priority poker) και θα αναφερθούμε αναλυτικά σε αυτό στο κεφάλαιο 7.



Εικόνα 4.19: *Παράδειγμα παρουσίασης στοιχείων που διαχειριζόμαστε σε ένα Product Backlog*

- Η διαχείριση των απαιτήσεων μέσω του product backlog γίνεται συνήθως με τη χρήση αυτοματοποιημένων εργαλείων. Από τα πιο γνωστά εργαλεία που επιτρέπουν τη διαχείριση του product backlog είναι
- Τα εργαλεία Jira και Trello της εταιρείας Atlassian (https://www.atlassian.com/)
- Το εργαλείο Monday project management από την Monday.com
- Το εργαλείο Miro από την miro.com
- Το εργαλείο Zoho sprints (https://sprints.zoho.eu/)
- κ.λπ.

Στην Εικόνα 4.20 παρουσιάζεται η διαχείριση των απαιτήσεων του product backlog με τη χρήση του εργαλείου Trello.



Εικόνα 4.20: *Παράδειγμα διαχείριση απαιτήσεων σε ένα Product Backlog με χρήση σύγχρονων εργαλείων*

Ένα καλό product backlog έχει στοιχεία (Product Backlog Items - PBI) τα οποία, σύμφωνα με τους Mike Cohn και Roman Pichler έχουν χαρακτηριστικά Detailed (λεπτομερή), Emergent (Αναδυόμενα), Estimated (να μπορούν να εκτιμηθούν), Prioritized (και να μπορούν να ιεραρχηθούν). Από τα αρχικά αυτά προκύπτει το ακρώνυμο DEEP (Pichler, 2020). Ας δούμε πιο προσεκτικά καθένα από αυτά τα χαρακτηριστικά:

- Τα ανεκτέλεστα στοιχεία/απαιτήσεις προϊόντων διαφέρουν ως προς **το επίπεδο λεπτομέρειάς** τους. Αυτά για τα οποία θα εργαστούμε στο επόμενο sprint θα πρέπει να είναι πιο λεπτομερή, ενώ αυτά που θα συμπεριληφθούν σε ένα sprint αργότερα μπορεί να είναι λιγότερο λεπτομερή.
- Όπως αναφέρθηκε σε προηγούμενες παραγράφους, το ανεκτέλεστο προϊόν (product backlog) είναι ένα ζωντανό έγγραφο, που αλλάζει συνεχώς καθώς το προϊόν αναπτύσσεται ή συντηρείται. Καθώς η ομάδα και ο ιδιοκτήτης του προϊόντος μαθαίνουν περισσότερα για το προϊόν και την αγορά, ενδέχεται να προσθέσουν νέα στοιχεία, να απορρίψουν ορισμένα και να αλλάξουν άλλα. Ο **αναδυόμενος χαρακτήρας** των απαιτήσεων προϊόντος (emergent product requirements) δεν είναι μόνο αναμενόμενος, αλλά αποτελεί ένδειξη υγιούς και λειτουργικού ανεκτέλεστου προϊόντος.
- Την κατάλληλη στιγμή, κάθε ανεκτέλεστο προϊόν θα πρέπει να έχει μια **εκτίμηση μεγέθους** που αντιστοιχεί στην προσπάθεια που απαιτείται για την ολοκλήρωσή του. Ο ιδιοκτήτης του προϊόντος χρησιμοποιεί αυτές τις εκτιμήσεις για να βοηθήσει στον προσδιορισμό της προτεραιότητας ενός στοιχείου του product backlog. Ιδανικά, για τα στοιχεία που θα συμπεριληφθούν στο επόμενο(α) sprint(s) θα πρέπει να έχουμε ακριβείς εκτιμήσεις.
- Θα πρέπει να είμαστε σε θέση να μπορούμε να αποφασίσουμε για **την προτεραιότητα της κάθε απαίτησης** του product backlog. Στόχος μας είναι πάντα να αυξήσουμε την αξία του παραδιδόμενου λογισμικού για τον πελάτη και η δυνατότητα να δίνουμε προτεραιότητα είναι ένα σημαντικό εργαλείο για την επίτευξη αυτού του στόχου.

Οι Sedano, Ralph, και Péraire (2019) κατέγραψαν 13 πρακτικές και 6 εμπόδια που σχετίζονται με δημιουργία του product backlog. Οι 13 βασικές πρακτικές που σχετίζονται με τη δημιουργία ενός επιτυχημένου product backlog συνοψίζονται στα ακόλουθα:

- Η ύπαρξη ισορροπημένων ομάδων. Μια ισορροπημένη ομάδα «είναι μια αυτόνομη ομάδα ανθρώπων με ποικίλες δεξιότητες και οπτικές γωνίες που υποστηρίζουν ο ένας τον άλλον προς έναν κοινό στόχο» (Thomson, 2016). Με άλλα λόγια, μια ομάδα πρέπει να περιλαμβάνει μέλη με διαφορετικά υπόβαθρα, τα οποία συνδυάζουν επιχειρηματικές γνώσεις (ιδιοκτήτες προϊόντων), δεξιότητες σχεδιασμού (σχεδιαστές προϊόντων) και δεξιότητες ανάπτυξης (μηχανικοί/προγραμματιστές). Εάν ένας ρόλος από τους παραπάνω υπο-εκπροσωπείται ή υπερ-εκπροσωπείται, η ομάδα είναι ανισόρροπη με άμεση επίπτωση στην ανάπτυξη του προϊόντος.
- Οργάνωση των εργασιών της ομάδας ανάπτυξης σε δύο άξονες. Ο πρώτος άξονας (track) περιλαμβάνει την έρευνα για τις ιστορίες χρηστών, τις συζητήσεις με τους συμμετέχοντες, τη σχεδίαση της μακέτας διεπαφής χρήστη και τη σύνταξη ιστοριών χρηστών. Ο δεύτερος άξονας συνήθως περιλαμβάνει την κατασκευή, τη δοκιμή, την αρχιτεκτονική, την ανακατασκευή, την ανάπτυξη και τη συντήρηση του προϊόντος. Η τεχνική αυτή ονομάζεται dual track agile.
- Μοντελοποίηση χρηστών (personas), χαρτογράφηση συμμετεχόντων (stakeholders mapping), συνεντεύξεις και αντιστοίχιση συγγένειας (affinity mapping). Η μοντελοποίηση χρηστών είναι ένα σημαντικό εργαλείο για την εύρεση των ιστοριών χρηστών. Μια persona είναι «ένα πρόσωπο, είναι μια περιγραφή ενός δυνητικού χρήστη που βασίζεται σε δεδομένα από την έρευνα χρηστών» [17]. Μια persona αντιπροσωπεύει συνήθως μια ομάδα παρόμοιων χρηστών (ή πιθανών χρηστών) που περιγράφεται με μια εικόνα, κίνητρα, δημογραφικά στοιχεία, συμπεριφορές, ανάγκες και στόχους.
- Χρήση τεχνικών βελτίωσης σχεδιασμού, όπως στούντιο σχεδιασμού (design studio), μακέτες οθονών (sketching mockups), έλεγχος χρηστικότητας (usability testing), κ.λπ. Η χρήση εργαλείων

σχεδιασμού βοηθά στο να αποτυπώσουμε τις ανάγκες των χρηστών και να έχουμε ένα επιτυχημένο product backlog. Ο σχεδιασμός περιλαμβάνει εικόνες/μακέτες ιστοσελίδων ή οθόνες κινητών, επεξήγηση της αλληλεπίδρασης των χρηστών με την εφαρμογή.

- Οργάνωση συναντήσεων για την παρουσίαση ιστοριών χρηστών (user story showcase) όπου παρουσιάζονται οι ιστορίες χρηστών με σκοπό την καλύτερη κατανόηση αυτών από τους συμμετέχοντες, το κούρεμα του καταλόγου των απαιτήσεων (backlog grooming) με σκοπό το ξεκαθάρισμα του product backlog, και τη συνάντηση για την αποδοχή των ιστοριών χρηστών (accepting stories).

Αντίστοιχα τα έξι βασικότερα εμπόδια τα οποία προέκυψαν από τη μελέτη των Sedano, Ralph, και Péraire (2019) ήταν:

- Το πρόβλημα της προκατάληψης ή των διαφορετικών απόψεων στην περιγραφή του προβλήματος. Συνήθως στην τεχνολογία λογισμικού θεωρούμε ότι τα προβλήματα είναι γνωστά και η λύση τους δεδομένη. Στην πράξη, συνήθως ποτέ το πρόβλημα δεν ορίζεται μονοσήμαντα αφού διαφορετικού συμμετέχοντες έχουν διαφορετικές απόψεις για την περιγραφή και τις αιτίες που προκαλούν το πρόβλημα
- Αντίστοιχα υπάρχει και το πρόβλημα της προκατάληψης ή των διαφορετικών απόψεων στην περιγραφή της λύσης του προβλήματος. Η προκατάληψη ή η προκαταβολική παραγωγή λύσεων από τους διαφορετικούς συμμετέχοντες πολλές φορές οδηγεί σε λανθασμένες αποφάσεις. Πάντα η λύση στο επιχειρηματικό πρόβλημα πρέπει να προκύπτει από την «περίπλοκη» διαδικασία της μελέτης των συμμετεχόντων, δημιουργία δοκιμαστικών, πρωτότυπων του συστήματος, κ.λπ. Κατά κοινή παραδοχή πελάτες που έρχονται στις συναντήσεις με την ομάδα ανάπτυξης ή ιδιοκτήτες προϊόντων που έχουν διαμορφώσει πρόωρα πλαίσια προβλημάτων ή λύσεων, τα οποία δεν έχουν προκύψει από προσεκτική έρευνα και προβληματισμό, εμποδίζουν τον σωστό σχεδιασμό του προϊόντος.
- Πίεση για πρώιμη σύγκλιση: Ο δημιουργικός σχεδιασμός συνδέεται στενά με τη μελέτη πολλές φορές αποκλινουσών και τη διερεύνηση ανόμοιων ιδεών. Η ανεπαρκής διερεύνηση του χώρου της λύσης και του σχεδιασμού και η πρόωρη σύγκλιση σε μια πρώιμη λύση είναι επομένως μια απειλή για την καινοτομία αλλά και το βέλτιστο σχεδιασμό του προϊόντος.
- Η αμφισημία (ambiguity) στις απαιτήσεις του προβλήματος ή η συχνή αλλαγή κατεύθυνσης αποτελεί ένα σημαντικό πρόβλημα για τη δημιουργία του product backlog. Πολλές φορές η αμφισημία δημιουργεί στρες στην ομάδα έργου και μειώνει το ηθικό της ομάδας.
- Ο χρόνος που απαιτείται για συνεντεύξεις με χρήστες, ή την έρευνα, για τη σύνθεση των εννοιών καθώς και για τη δημιουργία πρωτότυπων είναι σημαντικός. Η πίεση χρόνου συχνά επιδεινώνεται από τους πελάτες που δεν κατανοούν τη διαδικασία σχεδιασμού και την αξία της ή πιστεύουν ότι έχουν ήδη σχεδιάσει το προϊόν.
- Η δύσκολη πρόσβαση της ομάδας έργου στους τελικούς χρήστες είτε λόγω φόρτου εργασίας είτε για άλλους λόγους.

4.3.2 Ο κατάλογος των απαιτήσεων του sprint (sprint backlog)

Στα ευέλικτα έργα, ένα sprint είναι μια επανάληψη ανάπτυξης λογισμικού σταθερής χρονικής διάρκειας κατά την οποία η ομάδα ανάπτυξης δημιουργεί μια συγκεκριμένη ομάδα λειτουργιών προϊόντος. Στο τέλος κάθε sprint, το λογισμικό που έχει δημιουργήσει η ομάδα ανάπτυξης θα πρέπει να λειτουργεί, να είναι έτοιμο για επίδειξη και ενδεχομένως να μπορεί να αποσταλεί στον πελάτη. Σε έναν έργο, αποτελεί καλή πρακτική τα sprint να έχουν την ίδια χρονική διάρκεια. Η διατήρηση σταθερής διάρκειας στα sprint βοηθά την ομάδα ανάπτυξης να μετρήσει την απόδοσή της και να προγραμματίσει καλύτερα σε κάθε νέο επόμενο sprint. Μια συνηθισμένη χρονική διάρκεια είναι μεταξύ μιας και τεσσάρων εβδομάδων. Οι τέσσερις εβδομάδες είναι ο μεγαλύτερος χρόνος που πρέπει να διαρκεί ένα sprint, αφού οι επαναλήψεις μεγαλύτερης διάρκειας κάνουν

τις αλλαγές πιο δύσκολες και μειώνουν την ευελιξία του έργου. Συνήθως, η διάρκεια του sprint είναι δύο εβδομάδες, μα όλο και πιο συχνά παρατηρούμε sprint που διαρκούν μόλις μια εβδομάδα. Τα sprint μιας εβδομάδας είναι ένας φυσικός κύκλος, αυτός της εργάσιμης εβδομάδας, από Δευτέρα έως Παρασκευή, που αποτρέπει την εργασία κατά τη διάρκεια του Σαββατοκύριακου. Ορισμένες ομάδες Scrum εργάζονται σε μονοήμερα sprint όπου οι προτεραιότητες αλλάζουν σε καθημερινή βάση. Σήμερα, οι ανάγκες της αγοράς και των πελατών αλλάζουν όλο και πιο γρήγορα και ο χρόνος ακόμη της μιας ημέρας μπορεί να είναι σημαντικός. Η λογική αυτή υλοποιείται από προσεγγίσεις όπως αυτή της DevOps που θα παρουσιαστεί αναλυτικά στο 6^{ο} κεφάλαιο. Ο εμπειρικός μας κανόνας για τον υπολογισμό της διάρκειας του sprint είναι ότι το sprint δεν θα πρέπει να είναι μεγαλύτερο από το χρονικό διάστημα στο οποίο δεν υπάρχουν αλλαγές προτεραιοτήτων στις ιστορίες χρηστών.

Στο σημείο αυτό θα πρέπει να διαφοροποιήσουμε την έννοια της αποδέσμευσης (release) από αυτή του sprint. Μια αποδέσμευση (release) είναι μια έκδοση του προϊόντος (version), η οποία μπορεί να διατεθεί στους τελικούς χρήστες και περιλαμβάνει ένα σύνολο χρήσιμων για τον χρήστη χαρακτηριστικών του προϊόντος. Μια αποδέσμευση δεν χρειάζεται να περιλαμβάνει όλες τις λειτουργίες που περιγράφονται στον οδικό χάρτη προϊόντος, αλλά θα πρέπει να έχει τη μορφή ενός ελάχιστου βιώσιμου προϊόντος (Minimum Viable Product – MVP). Όπως έχουμε ήδη αναφέρει, οι επιχειρήσεις συχνά απαιτούν τη δημιουργία πρώιμης αξίας, αφού σε ένα δυναμικό εμπορικό περιβάλλον δεν μπορούμε να περιμένουμε την ανάπτυξη ενός πλήρους προϊόντος λογισμικού. Συνεπώς η έννοια της αποδέσμευσης δεν ταυτίζεται με την έννοια του sprint, διότι μια αποδέσμευση φτάνει πάντα στον τελικό πελάτη, ενώ ένα sprint αν και πάντα παράγει αξία για τον τελικό πελάτη δεν αποδεσμεύεται πάντα. Για παράδειγμα, το αποτέλεσμα ενός sprint θα μπορούσε να αξιολογηθεί μόνο από τον ιδιοκτήτη του προϊόντος. Στην πράξη ένας τυπικός αριθμός sprint (2-3) θα οδηγήσει πάντα σε αποδέσμευση (release). Η λογική αυτή μπορεί σε ένα βαθμό να θεωρηθεί ξεπερασμένη αφού με φιλοσοφία DevOps προσδιορίζει ότι το λογισμικό θα πρέπει να παραδίδεται άμεσα στον πελάτη (continuous delivery) χωρίς καμία καθυστέρηση και μάλιστα πολλές φορές καθημερινά (Kim et al., 2016).

Ο σχεδιασμός του sprint περιλαμβάνει:

- Τη στοχοθεσία του sprint και τη συμφωνία με την ομάδα έργου και τον ιδιοκτήτη προϊόντος για αυτούς τους στόχους.
- Την επιλογή των ιστοριών χρηστών που υποστηρίζουν αυτούς τους στόχους.
- Το χωρισμό και ομαδοποίηση των ιστοριών χρηστών σε συγκεκριμένες εργασίες ανάπτυξης.
- Τη δημιουργία ενός ανεκτέλεστου sprint backlog το οποίο αποτελείται από τα ακόλουθα:
    - Τη λίστα των ιστοριών χρηστών στο sprint με σειρά προτεραιότητας.
    - Την εκτίμηση της σχετικής προσπάθειας για κάθε ιστορία χρήστη.
    - Τις εργασίες που είναι απαραίτητες για την ανάπτυξη κάθε ιστορίας χρήστη.
    - Την προσπάθεια, σε ώρες, για την ολοκλήρωση της κάθε εργασίας.

Οι εργασίες σε ευέλικτα έργα θα πρέπει να διαρκούν μία ημέρα ή και λιγότερο για να ολοκληρωθούν για δύο λόγους. Ο πρώτος λόγος είναι ψυχολογικός, αφού σύντομες εργασίες ολοκληρώνονται πιο εύκολα και έγκαιρα. Ο δεύτερος λόγος είναι ότι είναι εύκολο να παρακολουθούμε την πρόοδο των εργασιών διάρκειας μιας ημέρας και συνεπώς είναι εύκολο να παρακολουθήσουμε την πρόοδο του έργου γενικότερα. Έτσι, εάν ένα μέλος της ομάδας ανάπτυξης αναφέρει ότι εργάζεται στην ίδια εργασία για περισσότερες από μία ή δύο ημέρες, αυτό οφείλεται πιθανότητα στην ύπαρξη κάποιου εμπόδιου.

4.3.3 Η επαύξηση (increment)

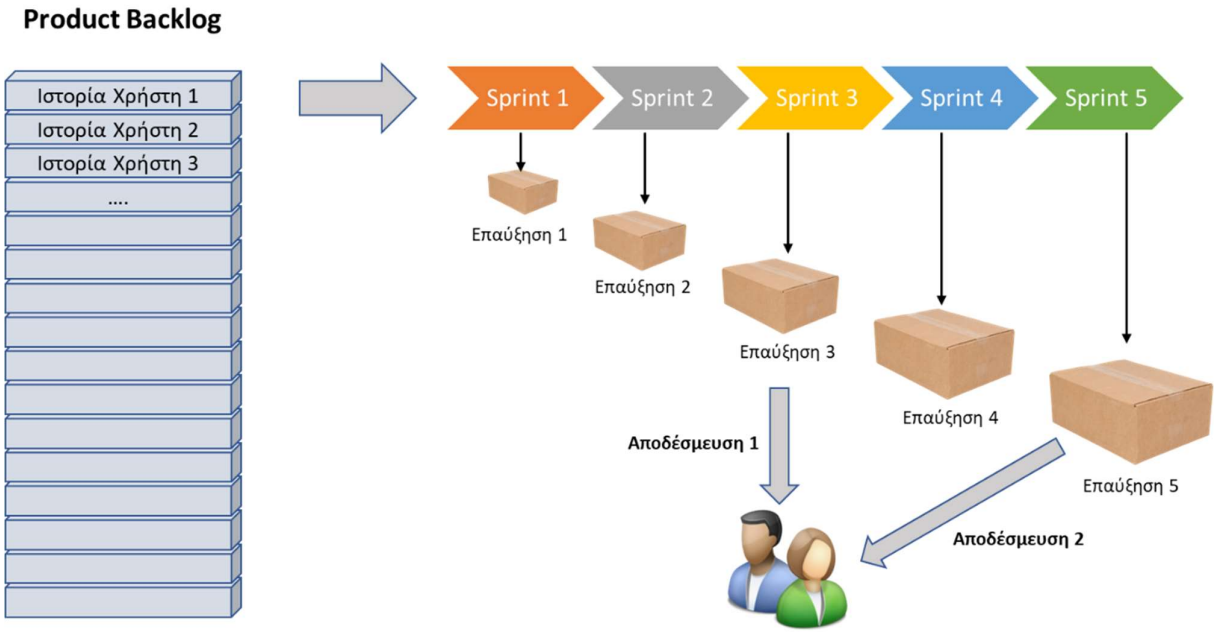
Σύμφωνα με τον οδηγό του Scrum η επαύξηση (increment) είναι ένα ξεκάθαρο βήμα προς την επίτευξη του στόχου του προϊόντος. Κάθε επαύξηση προστίθεται σε όλες τις προηγούμενες επαυξήσεις και έχει προσεκτικά επαληθευθεί, ώστε να διασφαλιστεί ότι όλες οι επαυξήσεις μαζί λειτουργούν σωστά. Η κάθε επαύξηση θα πρέπει να παρέχει αξία για τον πελάτη, και να είναι έτοιμη για χρήση.

Η επαύξηση είναι το άθροισμα όλων των PBI που ολοκληρώθηκαν κατά τη διάρκεια του τελευταίου sprint με τις επαυξήσεις όλων των προηγούμενων Sprint. Φανταστείτε να χτίζετε ένα σπίτι. Αρχικά το Product backlog περιλαμβάνει τα πάντα, τα θεμέλια, το σκελετό, τα υδραυλικά, τα ηλεκτρικά, τη μόνωση, τη στέγη κ.λπ.

Ας υποθέσουμε ότι στην παρούσα φάση έχουμε θεμελιώσει το σπίτι, έχουμε το σκελετό έχουμε χτίσει τους τοίχους και έχουμε φτιάξει και την οροφή. Στο τρέχον Sprint, οι προτεραιότητες είναι η εγκατάσταση της μπροστινής πόρτας, της πίσω πόρτας και των παραθύρων. Σύμφωνα με τον παραπάνω ορισμό η επαύξηση είναι το σπίτι στη μορφή που είναι σήμερα, δηλαδή αυτό που περιλαμβάνει τις πόρτες και τα παράθυρα.

Επομένως, κάθε επαύξηση προσθέτει κάτι επιπλέον, αλλά χρηστικό στο προϊόν που ήδη υπάρχει. Έτσι, η πρώτη επαύξηση είναι τα θεμέλια, η επόμενη επαύξηση είναι τα θεμέλια με το σκελετό, η επόμενη είναι τα θεμέλια, ο σκελετός και η τοιχοποιία, κ.λπ. Η έννοια της επαύξησης παρουσιάζεται γραφικά στην Εικόνα 4.21, από όπου μπορούμε να συμπεράνουμε ότι:

- όλα τα sprints παράγουν τουλάχιστον μία η περισσότερες επαυξήσεις,
- κάθε επαύξηση προσθέτει στις λειτουργίες του προϊόντος,
- κάθε επαύξηση είναι λειτουργούσα,
- κάθε επαύξηση είναι αυξητική, καθώς και
- δεν αποδεσμεύονται όλες οι επαυξήσεις καθώς κάποιες μπορεί να χρησιμοποιηθούν μόνο για έλεγχο σε σχέση με εξωτερικά συστήματα ή για επαλήθευση των απαιτήσεων.



Εικόνα 4.21: *Η έννοια της επαύξησης*

4.4 Οι τελετές στη διεργασία Scrum

4.4.1 Επανάληψη (sprint)

Όπως ήδη αναφέραμε, το sprint είναι ένας πλήρης κύκλος ανάπτυξης που περιλαμβάνει τον σχεδιασμό, την ανάλυση και την υλοποίηση συγκεκριμένων απαιτήσεων (sprint backlog). Η ομάδα ανάπτυξης οργανώνει τις εργασίες της, έτσι ώστε να παράγει μια νέα εκτελέσιμη και επαυξημένη εφαρμογή σε ένα χρονικό διάστημα που ποικίλλει από δύο έως τέσσερις εβδομάδες. Στον καθορισμό και υλοποίηση των στόχων μιας επανάληψης συμβάλλουν αποφασιστικά οι συνεδριάσεις για το σχεδιασμό των επαναλήψεων (sprint planning meetings), οι καθημερινές συνεδριάσεις scrum (daily scrum meetings), καθώς και οι συνεδριάσεις για την επανεξέταση της επανάληψης (sprint review meeting).

4.4.2 Συνεδρίαση για το σχεδιασμό της επανάληψης (sprint planning meeting)

Στην έναρξη κάθε sprint πραγματοποιείται το sprint planning meeting. Η συνεδρίαση χωρίζεται σε δύο μέρη τα οποία απαντούν στις παρακάτω ερωτήσεις (βλέπε Εικόνα 4.22):

- Τι θα παραδοθεί στον πελάτη με το τέλος του sprint;
- Πώς θα προγραμματιστεί η δουλειά ώστε να επιτευχθεί η παράδοση;

Το πρώτο μέρος του sprint planning meeting επικεντρώνεται στη διαπραγμάτευση για το ποια user stories θα υλοποιηθούν στο επόμενο sprint με βάση τις προτεραιότητες του ιδιοκτήτη προϊόντος και της ομάδας ανάπτυξης. Αυτές οι προτεραιότητες καθορίζουν και το στόχο του συγκεκριμένου sprint. Στη συνέχεια οι συμμετέχοντες στη συνάντηση συζητούν για τους στόχους και τις λειτουργίες των user stories που θα υλοποιηθούν. Πιο συγκεκριμένα, θα πρέπει να γίνουν τα ακόλουθα:

1. Να συζητηθεί και να τεθεί ο στόχος του sprint.
2. Να ελέγξουμε τις ιστορίες χρηστών από το product backlog και να δούμε ποιες υποστηρίζουν τον στόχο του sprint και να επανεξετάσουμε τις εκτιμήσεις για την απαιτούμενη προσπάθεια.
3. Εάν χρειάζεται, να δημιουργήσουμε νέες ιστορίες χρηστών για να συμπληρώσουμε τις πιθανές ελλείψεις που απαιτούνται για την επιτυχημένη ολοκλήρωση του στόχου του sprint.
4. Να προσδιορίσουμε το μέγεθος της προσπάθειας (effort) που μπορεί να αναλάβει η ομάδα έργου στο τρέχον sprint.

Το δεύτερο μέρος της συνεδρίασης εστιάζει στον λεπτομερή προγραμματισμό των εργασιών που πρέπει να γίνουν. Επίσης, τα μέλη της ομάδας αποφασίζουν τις ιστορίες χρηστών που θα υλοποιήσουν, διότι, όπως έχουμε ήδη αναφέρει, η ομάδα σε ένα Scrum αυτο-οργανώνεται. Για παράδειγμα, ο λεπτομερής προγραμματισμός των εργασιών (tasks) που γίνεται για μια ιστορία χρήση με τίτλο «Είσοδος στο λογαριασμό και προβολή στοιχείων» θα μπορούσε να περιλαμβάνει:

- Συγγραφή του μοναδιαίου ελέγχου (unit test).
- Δημιουργία οθόνης ελέγχου ταυτότητας χρήστη και κωδικού πρόσβασης.
- Δημιουργία οθόνης σφάλματος, ώστε ο χρήστης να εισαγάγει ξανά τα στοιχεία του σε περίπτωση λάθους.
- Δημιουργία οθόνης (μετά τη σύνδεση) που εμφανίζει τη λίστα των τραπεζικών λογαριασμών του πελάτη.

- Δημιουργία κλήσης στη βάση δεδομένων για την επαλήθευση του ονόματος χρήστη και του κωδικού πρόσβασης.
- Επανασυγγραφή κώδικα, ώστε να είναι κατάλληλος για κινητές συσκευές.
- Συγγραφή του ελέγχου ολοκλήρωσης (integration test).
- Ενημέρωση του αρχείου που περιέχει τη βοήθεια προς τον χρήστη.

Ένας βασικός περιορισμός που θα πρέπει να ικανοποιείται σε κάθε sprint είναι ότι ο συνολικός απαιτούμενος χρόνος για την υλοποίηση όλων των ιστοριών χρηστών ενός sprint δεν θα πρέπει να υπερβαίνει το συνολικό διαθέσιμο χρόνο της ομάδας έργου. Δηλαδή, αν η ομάδα έργου είναι 5 άτομα και το sprint έχει διάρκεια 3 εβδομάδες, ο διαθέσιμος χρόνος της ομάδας είναι 75 ανθρωποημέρες. Στην περίπτωση αυτή το άθροισμα της απαιτούμενης προσπάθειας των user stories για το συγκεκριμένο sprint δεν θα πρέπει να υπερβαίνει τις 70 ημέρες, καθότι θα πρέπει να υπολογίσουμε και τον αναγκαίο χρόνο για τις συναντήσεις που θα γίνουν κατά τη διάρκεια αυτής της χρονικής περιόδου.

Κατά το sprint planning δεν είναι απαραίτητο τα μέλη της ομάδας να επιλέξουν να υλοποιήσουν τις εργασίες που γνωρίζουν καλύτερα, γιατί οι συμμετέχοντες συνεργάζονται ανεξαρτήτως της ειδικότητας που κατέχει ο κάθε ένας και τα μέλη της ομάδας αλληλοβοηθιούνται. Με τον τρόπο αυτό τα άτομα της ομάδας έχουν τη δυνατότητα να διευρύνουν τις γνώσεις και τις ικανότητές τους σε διάφορους τομείς.

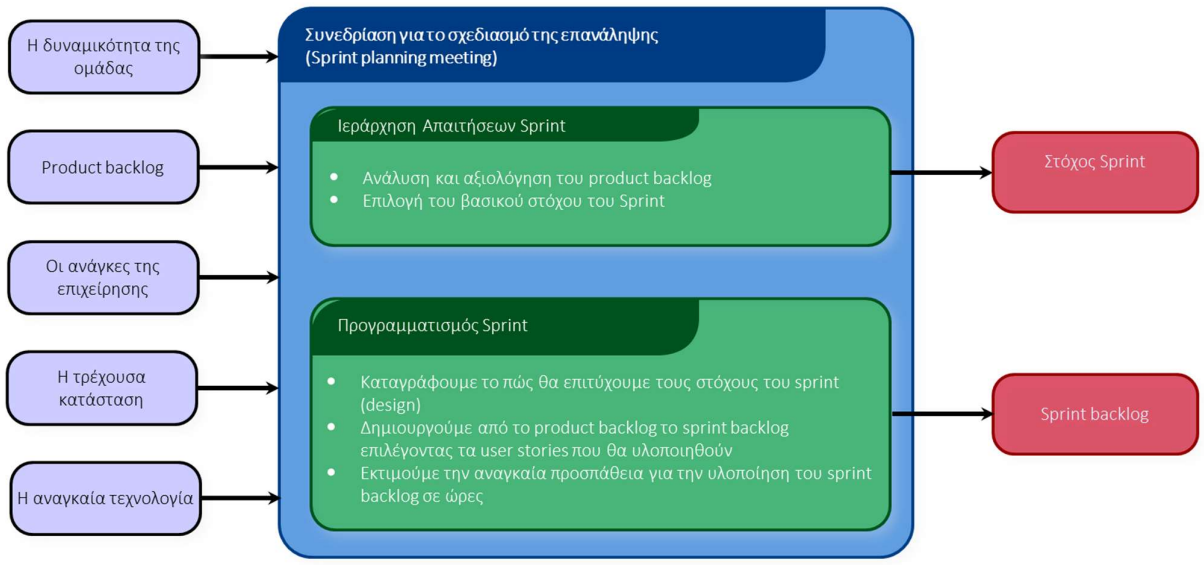
Επίσης θα πρέπει να δοθεί ιδιαίτερη προσοχή στη συνολική ποσότητα εργασίας που αναλαμβάνει η ομάδα έργου και ειδικά στα πρώτα sprints. Γενικά πάντα υπάρχει το θέμα της συνοχής της ομάδας καθώς και της καμπύλης μάθησης (learning curve) και τα προβλήματα αυτά είναι ιδιαίτερα εμφανή στα πρώτα sprints. Ένας εμπειρικός κανόνας είναι ότι η ομάδα στο 1^{ο} sprint δεν πρέπει να αναλαμβάνει έργο μεγαλύτερο του 25% της διαθέσιμης δυναμικότητας, στο 2^{ο} sprint θα πρέπει να στοχεύσουμε στο 50%, στο 3^{ο} sprint στο 75% και από το 4^{ο} sprint και μετά στο 100%.

Ακόμη και αν η ομάδα χρησιμοποιεί ειδικό λογισμικό για τη διαχείριση των sprint, καλό είναι να χρησιμοποιηθεί και ένας πιο φυσικός τρόπος για τη διαχείριση των δραστηριοτήτων του sprint. Ένας συνηθισμένος τρόπος για την παρακολούθηση της προόδου ενός sprint είναι η χρήση ενός πίνακα εργασιών (task board), στον οποίον καταγράφονται όλες οι ιστορίες χρηστών ενός sprint καθώς και η πρόοδός τους.

Η διάρκεια του sprint planning meeting εξαρτάται από τη διάρκεια του sprint. Έτσι αν το sprint έχει μεγάλη διάρκεια π.χ. 4 εβδομάδες η συνάντηση μπορεί να διαρκέσει μια ολόκληρη ημέρα. Ένας εμπειρικός κανόνας αναφέρει ότι:

Διάρκεια Sprint	**Διάρκεια συνάντησης**
1 εβδομάδα	2 ώρες
2 εβδομάδες	4 ώρες
3 εβδομάδες	6 ώρες
4 εβδομάδες	8 ώρες

Πίνακας 4.1: *Διάρκεια Sprint Planning Meeting .*



Εικόνα 4.22: *Συνεδρίαση για το σχεδιασμό της επανάληψης*

4.4.3 Το κούρεμα του product backlog (product backlog grooming)

Το κούρεμα του product backlog (product backlog grooming) ή αλλιώς η βελτίωση του backlog γίνεται όταν ο ιδιοκτήτης του προϊόντος ή τα μέλη της ομάδας έργου, ελέγχουν τα στοιχεία του backlog για να διασφαλίσουν ότι το backlog περιέχει τα κατάλληλα PBI. Αυτή η δραστηριότητα γίνεται σε τακτική βάση και μπορεί να είναι μια επίσημα προγραμματισμένη συνάντηση ή μια συνεχής δραστηριότητα που αναλαμβάνει ένα μέλος της ομάδας ανάπτυξης. Μερικές από τις δραστηριότητες που λαμβάνουν χώρα κατά τη διάρκεια της βελτίωσης του product backlog είναι (Cohn, 2015):

- κατάργηση ιστοριών χρηστών που δεν φαίνονται πλέον σχετικές ή κατάλληλες
- δημιουργία νέων ιστοριών χρηστών ως απάντηση σε ανάγκες που ανακαλύφθηκαν ή δημιουργήθηκαν πρόσφατα
- επαναξιολόγηση της προτεραιότητας των ιστοριών χρηστών
- εκτίμηση της απαιτούμενης προσπάθειας για ιστορίες χρηστών που δεν έχουν ακόμη εκτιμηθεί
- διόρθωση των εκτιμήσεων υπό το φως των νέων πληροφοριών που έχουμε διαθέσιμες
- διαχωρισμός ιστοριών χρηστών που είναι υψηλής προτεραιότητας αλλά είναι χονδροειδείς για να συμπεριληφθούν σε ένα επόμενο sprint.

4.4.4 Καθημερινή συνεδρίαση scrum (daily scrum meeting)

Αυτή η μικρής διάρκειας συνεδρίαση (περίπου δεκαπέντε λεπτών) οργανώνεται σε καθημερινή βάση για να ελέγχεται η πορεία και η απόδοση της ομάδας ανάπτυξης. Η συνάντηση πραγματοποιείται την ίδια ώρα και στο ίδιο μέρος κάθε ημέρα και ο σκοπός της είναι να βελτιωθεί η επικοινωνία και η διαδικασία λήψης αποφάσεων. Η συνάντηση είναι σημαντική όσον αφορά την αυτο-οργάνωση της ομάδας γιατί, αν το daily scrum meeting δεν πραγματοποιηθεί με επιτυχία, η ομάδα δεν θα γνωρίζει την τρέχουσα κατάσταση και ίσως μελλοντικά να αντιμετωπίσει δυσκολίες στην επίτευξη των στόχων της (Schwaber, 2008).

Η συνεδρίαση αυτή μπορεί να θεωρηθεί και ως συνεδρίαση σχεδιασμού, καθώς συζητιούνται τρία βασικά θέματα:

- Ποιο ήταν το αποτέλεσμα της χθεσινής εργασίας;

- Ποιος είναι ο στόχος για σήμερα;
- Εφόσον έχουν προκύψει κάποια προβλήματα ή τυχόν ελλείψεις, ποια είναι αυτά;

Στην καθημερινή συνεδρίαση scrum συμμετέχουν μόνο τα μέλη της ομάδας scrum και ο scrum master. Κατά τη διάρκεια της συνάντησης γίνεται και η ανανέωση του Task Board με τα νέα στοιχεία που έχουν προκύψει.

Με την πρώτη ματιά, ο σκοπός των τριών καθημερινών ερωτήσεων Scrum είναι σχετικά απλός και προφανής. Αυτές οι ερωτήσεις επιτρέπουν στην ομάδα να αξιολογήσει: (1) πώς απέδωσαν τις τελευταίες 24 ώρες και (2) πώς φαίνονται οι επόμενες 24 ώρες (Geekbot, 2020). Γενικότερα, οι 3 αυτές ερωτήσεις δίνουν πληροφορίες για τους τέσσερις κοινούς στόχους οποιουδήποτε ομαδικού ελέγχου:

- **Επιθεώρηση**: Πώς προχωρά η ομάδα προς τους στόχους της;
- **Διαφάνεια**: Είναι όλοι στην ομάδα συγχρονισμένοι και ενημερωμένοι σχετικά με τις εργασίες που έχουν ήδη γίνει και ποιες εργασίες πρέπει ακόμη να ολοκληρωθούν;
- **Προσαρμογή**: Χρειάζεται να προγραμματίσουμε εκ νέου σήμερα, με βάση τα γεγονότα που συνέβησαν την προηγούμενη εργάσιμη ημέρα;
- Ποιοι είναι οι λόγοι που **εμποδίζουν** την πρόοδο της ομάδας (blockers);

Στην πραγματικότητα οι τρεις αυτές ερωτήσεις παρέχουν και μη προφανείς πληροφορίες που δεν είναι εύκολα διαθέσιμες στις περισσότερες ομάδες.

Για παράδειγμα με την πρώτη ερώτηση μπορούμε να διακρίνουμε:

- **Οργανωτικές δυσλειτουργίες** (π.χ. πάρα πολλές συναντήσεις). Για παράδειγμα, είναι συνηθισμένο τα μέλη της ομάδας να απαντούν με το εξής: «Δεν έκανα πολλά χθες, καθώς το μεγαλύτερο μέρος της ημέρας μου αποτελούνταν από συσκέψεις». Αυτό θα μπορούσε να είναι ένα σημάδι ότι ο οργανισμός/επιχείρηση δεν λειτουργεί αποτελεσματικά, καθώς οι εργαζόμενοι δεν έχουν χρόνο να επικεντρωθούν πραγματικά στην εργασία τους – επομένως ίσως χρειάζεται μια οργανωτική αλλαγή για να αντιμετωπιστεί αυτό.
- Τρόπους για να ολοκληρώσουμε **το έργο πιο γρήγορα**, δηλαδή συμβουλές προερχόμενες από μέλη της ομάδας που ολοκλήρωσαν παρόμοιες εργασίες χθες. Για παράδειγμα, ένας προγραμματιστής μπορεί να εκτιμά ότι μια συγκεκριμένη εργασία απαιτεί 12 ώρες για να ολοκληρωθεί. Όμως κατά τη διάρκεια της συνάντησης, συνειδητοποιεί ότι χθες, κάποιο άλλο μέλος της ομάδας ολοκλήρωσε την ίδια ακριβώς εργασία σε πολύ λιγότερο χρόνο. Λογικά θα πρέπει να υιοθετήσει την ίδια μέθοδο.
- Να αποκτήσουμε **εσωτερική γνώση** σχετικά με τα χαρακτηριστικά των μελών της ομάδας έργου, όπως στην περίπτωση που ένα μέλος της ομάδας δίνει ιδιαίτερη σημασία στην ποιότητα. Για παράδειγμα, ας υποθέσουμε ότι ένας μηχανικός θα πρέπει να υλοποιήσει μια διόρθωση για ένα σφάλμα γρήγορα, - αλλά αντ' αυτού, κάνει ανακατασκευή του προϊόντος στο συγκεκριμένο σημείο ώστε να αποφύγει μελλοντικές ανεπιθύμητες καταστάσεις. Αυτό δείχνει ότι αυτός το μέλος της ομάδας δίνει αξία στην εργασία υψηλής ποιότητας και δεν καταφεύγει σε γρήγορες λύσεις.
- Αντιμετώπιση ασαφών εργασιών και ανεπαρκούς προγραμματισμού έργου.

Με βάση τη δεύτερη ερώτηση, δηλαδή ποιος είναι ο στόχος για σήμερα, μπορούμε να συμπεράνουμε:

- Ποιες εργασίες **δεν συμβάλλουν στον κύριο στόχο** της ομάδας, αφού μπορούμε να δούμε και να αναλύσουμε τα καθήκοντα της ομάδας για την ημέρα και να εξαλείψουμε τις εργασίες που δεν συμβάλλουν στον συνολικό στόχο του έργου. Ας υποθέσουμε ότι ο στόχος ενός πιεσμένου χρονικά έργου είναι να αυξήσει την ταχύτητα απόκρισης του ιστότοπου. Αν κατά τη διάρκεια της συνάντησης, κάποιος δηλώνει ότι θα εργαστεί για την αποσφαλμάτωση ενός άλλου προβλήματος

που δεν συμβάλλει στον γενικό στόχο, τότε θα πρέπει να αλλάξει η εργασία αυτού του μέλους ώστε να ευθυγραμμιστεί με τους στόχους της ομάδας.

- Να εντοπίσουμε ποια μέλη της ομάδας θα μπορούσαν **να προσφέρουν βοήθεια** σε συναδέλφους (δηλαδή, εάν ένας συνάδελφος εργάζεται πάνω σε ένα πρόβλημα που κάποιος στην ομάδα έχει ήδη αντιμετωπίσει στο παρελθόν)
- Να εντοπίσουμε ζητήματα διαχείρισης όπως **αλληλεπικαλύψεις, εξαρτήσεις** κ.λπ. Για παράδειγμα, μπορούμε πολύ εύκολα να εντοπίσουμε αν υπάρχουν μέλη της ομάδας που εργάζονται στην ίδια ιστορία χρήστη ενώ δεν απαιτείται, ή μια ιστορία χρήστη στην οποία δεν εργάζεται κανείς. Γνωρίζοντας το αντικείμενο της εργασίας του κάθε μέλους της ομάδας ανάπτυξης βοηθά, ώστε η εργασία και οι στόχοι να γίνονται πιο ξεκάθαροι και κατανοητοί.

Τέλος με βάση την τρίτη ερώτηση για τους λόγους που εμποδίζουν την πρόοδο της ομάδας, θα μπορούσαμε να χρησιμοποιήσουμε τη συζήτηση για να λάβουμε αποφάσεις που αφαιρούν το εμπόδιο. Το πρόβλημα είναι ότι η επίλυση του προβλήματος μπορεί να απαιτεί περισσότερο χρόνο από τα 15 λεπτά που είναι διαθέσιμα για αυτή τη συνάντηση.

Οι Stray, Moe και Aurum (2012) μελέτησαν τον τρόπο με τον οποίο αλληλοεπιδρούν τα μέλη της ομάδας μεταξύ τους στις καθημερινές συναντήσεις και εντόπισαν τα εξής:

- Η καθημερινή συνάντηση επικεντρώνεται κυρίως στην επικοινωνία και ειδικά στα εμπόδια και στα προβλήματα.
- Η μικρή διάρκεια της συνάντησης οδηγεί στη λήψη γρήγορων αποφάσεων, υπό πίεση χρόνου, χωρίς να εξεταστούν εναλλακτικές λύσεις.
- Η ομάδα δεν συζητά για πολύ χρόνο για το συντονισμό των εργασιών.
- Πολλές φορές τα άτομα της ομάδας δεν μιλούν αυτοβούλως αλλά ο scrum master τους ζητά να μιλήσουν.

4.4.5 Συνεδρίαση για την επανεξέταση του sprint (sprint review meeting)

Μετά την ολοκλήρωση ενός sprint, ο ιδιοκτήτης του προϊόντος μαζί με την ομάδα Scrum επανεξετάζουν την επανάληψη που μόλις τελείωσε. Η διαδικασία αυτή ονομάζεται sprint review, όπου η ομάδα παρουσιάζει τα αποτελέσματα του sprint και εξετάζεται κατά πόσο οι στόχοι που είχαν τεθεί στο sprint planning έχουν ολοκληρωθεί με επιτυχία. Μια βασική αρχή στο Scrum είναι ο έλεγχος και η προσαρμοστικότητα (inspect and adapt). Αυτό σημαίνει πως, αφού εξεταστεί το αποτέλεσμα που έχει προκύψει, τότε εντοπίζονται οι αλλαγές ή οι προσαρμογές, εφόσον υπάρχουν, που θα πρέπει να γίνουν στο σύστημα κατά τις επόμενες επαναλήψεις. Γίνεται μια λεπτομερής συζήτηση μεταξύ του product owner και της ομάδας ώστε να προσδιοριστεί η κατάσταση στην οποία βρίσκεται η όλη ανάπτυξη του συστήματος.

Επίσης, κατά τη διάρκεια της επανεξέτασης του sprint γίνεται μια παρουσίαση των αποτελεσμάτων στους χρήστες του προϊόντος. Οι συμμετέχοντες αξιολογούν την ανάπτυξη του έργου και λαμβάνουν αποφάσεις που αφορούν τις επόμενες λειτουργίες που θα πρέπει να υλοποιηθούν. Αυτή η συνεδρίαση είναι σημαντική για την πορεία του έργου καθώς μπορεί να επιφέρει νέα στοιχεία στον κατάλογο του product backlog αλλά και αναθεώρηση των υπαρχόντων. Επιπλέον, καθορίζονται οι στόχοι για το επόμενο sprint.

4.4.6 Η ανασκόπηση της επανάληψης (sprint retrospective)

Το sprint retrospective είναι μια διαδικασία που ακολουθεί την επανεξέταση του sprint. Αφορά την εξέταση και αναθεώρηση της διαδικασίας που ακολουθήθηκε κατά το τελευταίο sprint. Χρησιμοποιείται για να εντοπιστούν οι καθοριστικοί παράγοντες επιτυχίας και πιθανοί τομείς της διαδικασίας που μπορούν να βελτιωθούν. Στο στάδιο αυτό η ομάδα έχει τη δυνατότητα να συζητήσει τι δουλεύει και τι όχι και να καταλήξουν, τα μέλη της, σε τυχόν αλλαγές που μπορούν να εφαρμόσουν στο επόμενο sprint.

Στη διαδικασία λαμβάνουν μέρος ο scrum master και η ομάδα ανάπτυξης, ενώ η παρουσία του ιδιοκτήτη προϊόντος δεν είναι υποχρεωτική. Μια πρακτική για την οργάνωση της συνάντησης είναι να σχηματιστούν δύο στήλες σε έναν πίνακα. Στη μια στήλη τοποθετούνται οι παράγοντες επιτυχίας (what is working well) και στη δεύτερη στήλη οι πιθανοί τομείς βελτίωσης (what could work better). Τα μέλη της ομάδας προσθέτουν στον πίνακα ένα ή περισσότερα στοιχεία σε οποιαδήποτε από τις δύο στήλες. Έπειτα γίνεται καταμέτρηση και προκύπτουν τα δημοφιλέστερα προβλήματα ή παράγοντες επιτυχίας, αντίστοιχα. Τα ζητήματα που αφορούν βελτιώσεις στη διαδικασία κατατάσσονται με σειρά προτεραιότητας και υλοποιούνται στα επόμενα sprints. Ο Kniber (2015) προτείνει η ομάδα να επικεντρώνεται μόνο σε μία περιοχή βελτίωσης ανά sprint. Επίσης, είναι σημαντικό να διαχωρίζονται τα στοιχεία που αφορούν τη διαδικασία της scrum από τους υπόλοιπους παράγοντες.

Βιβλιογραφία/Αναφορές

Φιτσιλής, Π. (2018). Σύγχρονα πληροφοριακά συστήματα επιχειρήσεων (2η έκδοση). Broken Hill Publishers

Bass, J. M. (2014, August). Scrum master activities: process tailoring in large enterprise projects. In *2014 IEEE 9th international conference on global software engineering* (pp. 6-15). IEEE.

Beck K. (2005). Extreme Programming Explained: Embrace Change (2nd ed.). Addison Wesley.

Belling S. (2020) Agile Teams and Challenges. In: Succeeding with Agile Hybrids. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6461-4_6

Brooks Jr, F. P. (1995). *The mythical man-month: essays on software engineering*. Pearson Education.

Cohn, M. (2015). Product backlog refinement (grooming).

Dean, D. & Webb, C. (2011, January). Recovering from information overload. McKinsey Quarterly, https://www.mckinsey.com/business-functions/organization/our-insights/recovering-from-information-overload

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2012). A lightweight guide to the theory and practice of scrum. Ver, 2, 2012.

Geekbot (2020), Analyzing the 3 Daily Standup Questions: Common Pitfalls & Unique Ideas. https://geekbot.com/blog/daily-standup-questions/#h.ikcs5jvn1ovp

Goldratt, E. M. (1997). Critical chain. Great Barrington, MA: North River Press.

Greenleaf, R. K. (1998). *The power of servant-leadership: Essays*. Berrett-Koehler Publishers.

Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176-189.

Holtzhausen, N., & de Klerk, J. J. (2018). Servant leadership and the Scrum team's effectiveness. Leadership & Organization Development Journal.

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations*. IT Revolution.

Kniberg, H. (2015). *Scrum and XP from the Trenches*. Lulu.com.

Layton, M. C., & Ostermiller, S. J. (2017). Agile project management for dummies, 2nd edition. John Wiley & Sons.

Noll, J., Razzak, M. A., Bass, J. M., & Beecham, S. (2017, November). A study of the scrum master's role. In *International Conference on Product-Focused Software Process Improvement* (pp. 307-323). Springer, Cham.

Pichler, R. (2010). *Agile product management with scrum: Creating products that customers love*. Pearson Education India.

Pichler, R. (2020). *How to Lead in Product Management: Practices to Align Stakeholders, Guide Development Teams, and Create Value Together*. Pichler Consulting.

Rozovsky, J. (2015), "The five keys to a successful google team", available at: https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/.

Salas, E., Sims, D. E., & Burke, C. S. (2005). Is there a "big five" in teamwork?. *Small group research*, *36*(5), 555-599.

Sedano, T., Ralph, P., & Péraire, C. (2019, May). The product backlog. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (pp. 200-211). IEEE.

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum* (Vol. 1). Upper Saddle River: Prentice Hall.

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game.(2011). *Available: scrum. org*.

Stray, V. G., Moe, N. B., & Aurum, A. (2012, September). Investigating daily team meetings in agile software projects. In *2012 38th Euromicro Conference on Software Engineering and Advanced Applications* (pp. 274-281). IEEE.

Sutherland, J. (2001). Inventing and Reinventing SCRUM in five Companies. *Cutter IT journal*, *14*, 5-11.

Sutherland, J., & Schwaber, K. (2007). The scrum papers. *Nuts, Bolts and Origins of an Agile Process*.

Takeuchi, H., & Nonaka, I. (1986). The new product development game. *Harvard business review*, *64*(1), 137-146.

Thomson, J. (2016). Trust and balanced teams. https://medium.com/product-labs/trust-and-balanced-teams-919456ad57cf

Williams, L., & Kessler, R. R. (2003). *Pair programming illuminated*. Addison-Wesley Professional.

Van Dierendonck, D. (2011). Servant leadership: A review and synthesis. *Journal of management*, *37*(4), 1228-1261.

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Ποιοι είναι οι βασικοί ρόλοι της διεργασίας Scrum;

Απάντηση/Λύση

Οι βασικοί ρόλοι της διεργασίας Scrum είναι:

- ο διαχειριστής Scrum (θα αναφέρεται εφεξής ως Scrum master)
- ο ιδιοκτήτης του προϊόντος (Product Owner),
- η ομάδα Scrum (Scrum team),
- ο πελάτης (Customer),
- ο χρήστης (User) και
- η διοίκηση (Administration).

Κριτήριο αξιολόγησης 2

Ποιοι είναι τα βασικά τεχνουργήματα της διεργασίας Scrum;

Απάντηση/Λύση

Τα βασικά τεχνουργήματα της διεργασίας Scrum είναι:

- ο κατάλογος των απαιτήσεων του προϊόντος (product backlog)
- ο κατάλογος των απαιτήσεων του sprint (sprint backlog) και
- η επαύξηση

Κριτήριο αξιολόγησης 3

Ποιοι είναι τα βασικά γεγονότα της διεργασίας Scrum;

Απάντηση/Λύση

Τα βασικά γεγονότα της διεργασίας Scrum είναι τα ακόλουθα:

- Το sprint
- Ο σχεδιασμός του sprint (sprint planning)
- Η καθημερινή συνάντηση (daily scrum)
- Η ανασκόπηση[5] του sprint (sprint review)

[5] Η λέξη review μεταφράζεται από την αγγλική ως ανασκόπηση, ενώ η λέξη retrospective ως επισκόπηση. Η επισκόπηση ως εργασία είναι η γενική, ανακεφαλαιωτική, βασική εξέταση του φαινομένου υπό μελέτη π.χ. η μελέτη της βιβλιογραφίας που γίνεται από τον ερευνητή προκειμένου να επιλέξει ένα ερευνητικό θέμα. Αντίθετα η ανασκόπηση είναι πιο συγκεκριμένη και περισσότερο εις βάθος, για τις ανάγκες του συγκεκριμένου sprint.

- Η επισκόπηση του sprint (sprint retrospective)

Κριτήριο αξιολόγησης 4

Ποιες είναι οι προϋποθέσεις για τη σωστή λειτουργία μια ομάδας;

Απάντηση/Λύση

Οι Salas και λοιποί (Salas et al., 2005) υποστηρίζουν ότι οι ομάδες απαιτούν για τη λειτουργία τους ένα σύνθετο μείγμα παραγόντων που περιλαμβάνουν την οργανωτική υποστήριξη, ατομικές δεξιότητες, καθώς και δεξιότητες ομαδικής εργασίας. Τα πέντε βασικά συστατικά που εντόπισαν στην έρευνά τους είναι: ηγεσία ομάδας, αμοιβαία παρακολούθηση της απόδοσης (mutual performance monitoring), πνεύμα αλληλοϋποστήριξης (backup behavior), προσαρμοστικότητα (adaptability) και προσανατολισμός ομάδας. Κάθε ένα από τα πέντε αυτά χαρακτηριστικά είναι προαπαιτούμενο για την αποτελεσματικότητα της ομάδας, αλλά και κάθε στοιχείο μπορεί να εκδηλωθεί διαφορετικά ανάλογα με το είδος της εργασίας και τους περιορισμούς ή τις ανάγκες της ομάδας.

Κριτήριο αξιολόγησης 5

Δώστε τον ορισμό ενός κοινού νοητικού μοντέλου και το πώς αυτό αναπτύσσεται στο Scrum.

Απάντηση/Λύση

Ένα κοινό νοητικό μοντέλο οδηγεί σε μια κατάσταση κατά την οποία η γνώση που έχει κάθε μέλος μιας ομάδας σχετικά με τις επερχόμενες ενέργειες της ομάδας είναι τουλάχιστον παρόμοια με τη γνώση των άλλων μελών της ομάδας για τις ίδιες ενέργειες. Στο Scrum τα κοινά νοητικά μοντέλα υποστηρίζονται μέσω της συμμετοχής του ιδιοκτήτη προϊόντος, την εστίαση στο όραμα του έργου και τον προγραμματισμό που γίνεται με τη χρήση του product backlog και την επισκόπηση του sprint. Η καθημερινή συνάντηση είναι επίσης σημαντική για την κατανόηση των καθηκόντων των μελών της ομάδας.

Κριτήριο αξιολόγησης 6

Είστε ο ιδιοκτήτης ενός προϊόντος. Παρουσιάστε το όραμα ενός γνωστού προϊόντος λογισμικού χρησιμοποιώντας τις ερωτήσεις της Εικόνας 4.13.

Απάντηση/Λύση

Η δημιουργία του οράματος ενός προϊόντος απαιτεί μια σύντομη περιγραφή του πελάτη στόχου, των βασικών πλεονεκτημάτων του προϊόντος έναντι του ανταγωνισμού καθώς και του μότο.

Θα δώσουμε ένα παράδειγμα χρησιμοποιώντας ένα ηλεκτρονικό ημερολόγιο για ένα φανταστικό προϊόν που ονομάζεται eCalendar:

Για τον πελάτη επιχειρηματία **που χρειάζεται** ένα ευέλικτο ηλεκτρονικό ημερολόγιο **το προϊόν** eCalendar **προσφέρει** αυτοματισμούς που επιλύουν με το βέλτιστο τρόπο το πρόβλημα του προγραμματισμού των συναντήσεων με χρήση τεχνητής νοημοσύνης και φτιάχνοντας ένα μοναδικό ατομικό προφίλ.

Το προϊόν είναι διαφορετικό από τα προϊόντα του ανταγωνισμού που απλά καταγράφουν την ώρα της συνάντησης.

Το μότο είναι "Δημιουργούμε χρόνο για εσάς".

Κριτήριο αξιολόγησης 7

Ποιες είναι οι βασικές δραστηριότητες ενός ιδιοκτήτη προϊόντος;

Απάντηση/Λύση

Ο ιδιοκτήτης του προϊόντος έχει τις παρακάτω δραστηριότητες:

- Ορίζει τις απαιτήσεις του προϊόντος χρησιμοποιώντας ιστορίες χρηστών.
- Κάνει ιεράρχηση των απαιτήσεων ανάλογα με την επιχειρηματική αξία.
- Αποφασίζει την ημερομηνία κυκλοφορίας του προϊόντος καθώς και τα χαρακτηριστικά που θα συμπεριληφθούν σε κάθε αποδέσμευση.
- Είναι υπεύθυνος για την κερδοφορία του προϊόντος και το ROI.
- Αποδέχεται ή απορρίπτει τα αποτελέσματα της εργασίας του κάθε sprint.

Κριτήριο αξιολόγησης 8

Ποια είναι τα βασικά λάθη στην εφαρμογή του ρόλου του ιδιοκτήτη του προϊόντος;

Απάντηση/Λύση

Τα πιο συχνά λάθη στην εφαρμογή του ρόλου είναι τα ακόλουθα:

- Ο ιδιοκτήτης προϊόντος δεν έχει ικανή επιρροή μέσα στην επιχείρηση που αναπτύσσει το προϊόν.
- Ο ιδιοκτήτης προϊόντος που έχει υπερβολικό φόρτο εργασίας και δεν μπορεί να ανταποκριθεί στις υποχρεώσεις του.
- Ο χωρισμός του ρόλου και η ανάθεση καθηκόντων σε περισσότερα του ενός άτομα.
- Η δημιουργία επιτροπής ιδιοκτητών προϊόντος που υποκαθιστά τον ατομικό ρόλο.
- Ο απομακρυσμένος ιδιοκτήτης προϊόντος που βρίσκεται σε απόσταση από την ομάδα έργου.

Κριτήριο αξιολόγησης 9

Μελετήστε το άρθρο του Van Dierendonck[6] με τίτλο «Servant Leadership: A Review and Synthesis» που αναφέρεται στη ηγεσία-υπηρέτης και περιγράψτε τα έξι βασικά χαρακτηριστικά που ορίζουν τη συμπεριφορά του ηγέτη-υπηρέτη.

Απάντηση/Λύση

Τα έξι βασικά χαρακτηριστικά που χαρακτηρίζουν τη συμπεριφορά της ηγεσίας-υπηρέτης όπως τη βιώνουν τα μέλη της ομάδας είναι:

- οι ηγέτες αυτού του είδους ενδυναμώνουν και αναπτύσσουν τις ικανότητες των μελών της ομάδας,

[6] Το άρθρο αναφέρεται στη βιβλιογραφία του κεφαλαίου και μπορείτε να το ανακτήσετε στο διαδίκτυο χρησιμοποιώντας την υπηρεσία scholar.google.com

- δείχνουν ταπεινότητα,
- είναι αυθεντικοί,
- αποδέχονται τους ανθρώπους γι' αυτό που είναι,
- δίνουν κατεύθυνση στην ομάδα και
- εργάζονται για το κοινό καλό.

Τα χαρακτηριστικά αυτά παρουσιάζονται αναλυτικά στις σελίδες 4-7 του άρθρου και στην παράγραφο με τίτλο «Key Characteristics of Servant Leadership».

Κριτήριο αξιολόγησης 10

Αναφέρατε τις πιο σημαντικές μεθόδους ιεράρχησης των απαιτήσεων.

Απάντηση/Λύση

Η ιεράρχηση των απαιτήσεων γίνεται χρησιμοποιώντας μεθόδους όπως:

- Η μέθοδος MoSCoW
- Ταξινόμηση (ranking)
- Analytical Hierarchical Process (AHP)
- Το μοντέλο Kano
- Μέθοδος στάθμισης με βάρη
- Η μέθοδος των 100 δολαρίων
- Κ.λπ.

Κριτήριο αξιολόγησης 11

Ποια είναι η μικρότερη δομική μονάδα που μπορεί να μπει σε ένα κατάλογο απαιτήσεων προϊόντος (product backlog);

Απάντηση/Λύση

Τα epics, που αποτελούν ένα σύνολο λειτουργιών που σχετίζονται με ένα χαρακτηριστικό. Το epic είναι η μεγαλύτερη μονάδα λειτουργικότητας που μπορούμε να συμπεριλάβουμε σε ένα sprint, διότι είναι αρκετά μικρή και αναλυτική ώστε να μπορεί να προγραμματιστεί η ανάπτυξή της με συγκεκριμένο τρόπο.

Κριτήριο αξιολόγησης 12

Δώστε ένα αναλυτικό παράδειγμα όπου να γίνεται εμφανής η διαφορά μεταξύ θέματος (theme), (feature), (epic), (user story).

Απάντηση/Λύση

Στον επόμενο πίνακα παρουσιάζουμε ενδεικτικά παραδείγματα για την κάθε περίπτωση.

Επίπεδο απαίτησης	Παράδειγμα
Θέμα – Theme	Να παρουσιάσουμε τα δεδομένα του τραπεζικού λογαριασμού ενός χρήστη σε μια κινητή συσκευή.
Χαρακτηριστικό – Feature	Να παρουσιάσουμε τα υπόλοιπα των λογαριασμών ενός χρήστη. Να παρουσιάσουμε τη λίστα με πρόσφατες αναλήψεις ή αγορές. Να παρουσιάσουμε τη λίστα με πρόσφατες καταθέσεις. Να παρουσιάσουμε τις επερχόμενες αυτόματες πληρωμές λογαριασμών. Να παρουσιάσουμε τις ειδοποιήσεις του λογαριασμού.
Επικό – Epic (αντιστοιχούν στην ανάλυση του χαρακτηριστικού «Να δούμε τα υπόλοιπα των λογαριασμών»).	Να παρουσιάσουμε το υπόλοιπο του τρέχοντος λογαριασμού. Να παρουσιάσουμε το υπόλοιπο του λογαριασμού ταμιευτηρίου. Να παρουσιάσουμε το υπόλοιπο δανείου. Να παρουσιάσουμε το υπόλοιπο του επενδυτικού λογαριασμού.
Ιστορία χρήστη – User Stories (αντιστοιχούν στο epic «Να δούμε το υπόλοιπο του τρέχοντος λογαριασμού»).	Να παρουσιάσουμε, αφού προηγηθεί σύνδεση με ασφάλεια, μια λίστα με τους λογαριασμούς του χρήστη. Ο χρήστης να επιλέξει για να δείτε τον λογαριασμό όψεως. Ο χρήστης να δει το υπόλοιπο του λογαριασμού μετά από ανάληψη. Ο χρήστης να δει το υπόλοιπο του λογαριασμού μετά από κατάθεση. Ο χρήστης να δει το υπόλοιπο του λογαριασμού στο τέλος της ημέρας. Ο χρήστης να δει το υπόλοιπο του λογαριασμού. Ο χρήστης να αποσυνδεθεί από την εφαρμογή για κινητά.

Κριτήριο αξιολόγησης 13

Με τη χρήση ενός εκ των πολλών εργαλείων διαχείρισης product backlog (π.χ. Trello.com) εισάγετε τις απαιτήσεις του παρακάτω πίνακα, ώστε να δημιουργήσετε ένα στοιχειώδες product backlog.

PRODUCT BACKLOG	
Α/Α	ΠΕΡΙΓΡΑΦΗ ΑΠΑΙΤΗΣΗΣ
1	Επικοινωνία και επιλογή εργαλείων επικοινωνίας
2	Δοκιμή εργαλείων και Σύνδεση με Βιβλιοθήκες
3	Σύνδεση, Δημιουργία βάσης και Δημιουργία πινάκων
4	Δημιουργία Οθονών και Δημιουργία Menu
5	Λίστα επιλογής κατηγορίας δεδομένων, Πλήκτρα εισαγωγή χωρών - Εισαγωγή δεδομένων, Κλήση API και Ανάκτηση δεδομένων από Json, Buiding της ΒΔ με το GUI και Εισαγωγή δεδομένων στη ΒΔ
6	Διαγραφή δεδομένων, Διαγραφή χωρών, Μήνυμα επιβεβαίωσης Διαγραφής
7	Διασύνδεση με τη ΒΔ, Επιλογή από λίστα χώρας, Ανάκτηση δεδομένων σε πίνακα (3 σελίδες/κατηγορία), Πεδία Ημερομηνίας από-εως, Πλήκτρο Προβολή σε χάρτη, Πλήκτρο Προβολή σε Διάγραμμα, Διαγραφή Χώρας, Binding της ΒΔ με το GUI και Διαγραφή δεδομένων
8	Εμφάνιση Δεδομένων σε Διάγραμμα , Επιλογή μιας καμπύλης από τρείς/ εμφάνιση σωρευτικών δεδομένων
9	Εμφάνιση των Δεδομένων της Επιλεχθήσας Χώρας σε Χάρτη
10	Πεδίο βασικής επιλογής χώρας,Πεδίο πολλαπλής επιλογής χωρών, Πεδία ημερομηνίας από-εώς, Εμφάνιση στο χάρτη με Mark, Pop-up με εμφάνιση δεδομένων, Binding της ΒΔ με το GUI

Απάντηση/Λύση

Μετά την επιλογή του κατάλληλου εργαλείου (π.χ. Trello) και τη δημιουργία του αντίστοιχου λογαριασμού θα πρέπει να δημιουργηθεί το product backlog. Το Trello είναι ένα εργαλείο που βοηθά την ομάδα στο σχεδιασμό, στον έλεγχο και στην υλοποίηση του έργου, εντός του χρονοδιαγράμματος που έχει ορίσει ο product owner. Τα μέλη της ομάδας μπορούν να συνεργάζονται και να ενημερώνονται σε πραγματικό χρόνο μεταξύ τους διαδικτυακά.

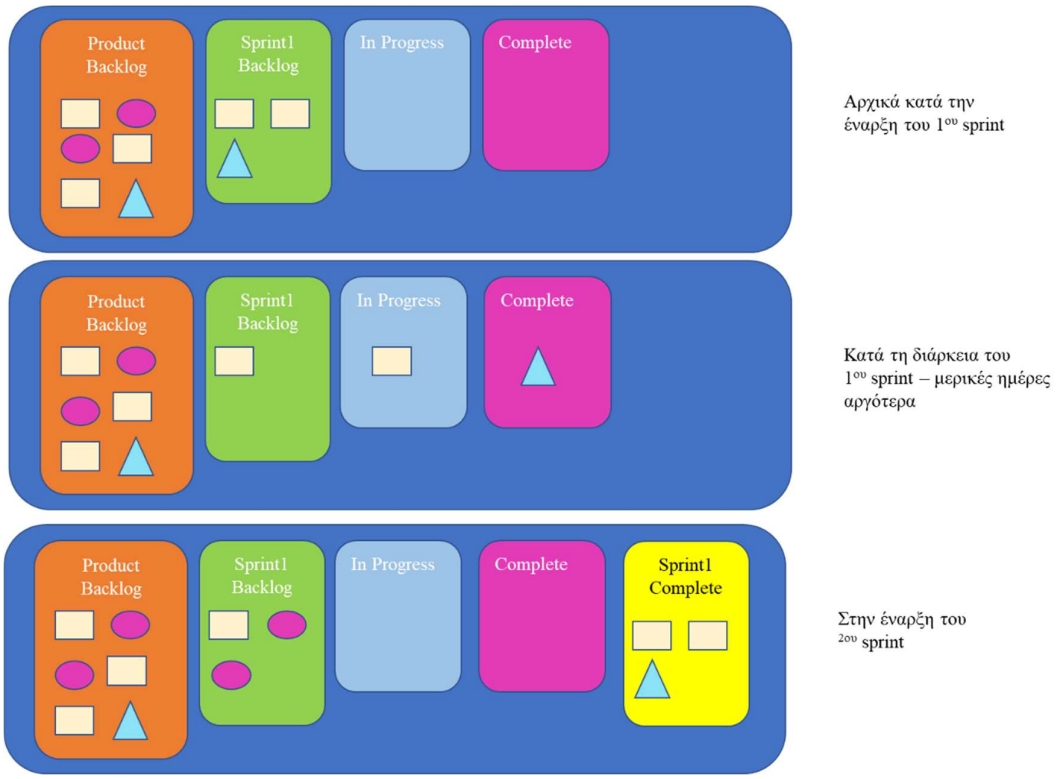
Το μέλος της ομάδας που αναλαμβάνει το ρόλο του Scrum master θα πρέπει να δημιουργήσει το Agile Board με τίτλο το όνομα του project π.χ. «Project_example». Επειδή το εργαλείο Trello είναι κατά βάση εργαλείο συνεργασίας γενικού σκοπού θα πρέπει να χρησιμοποιηθεί το αντίστοιχο Trello template. Στη συνέχεια θα πρέπει να προσκαλέσει σε αυτό τα μέλη της ομάδας έργου ώστε να μπορούν αφενός να έχουν πρόσβαση στο περιβάλλον συνεργασίας αφετέρου και να μπορούν να αναλάβουν εργασίες.

Η οργάνωση του board θα πρέπει να γίνει με τέτοιο τρόπο ώστε να υπάρχει άμεση εποπτεία των εκκρεμών και των ολοκληρωμένων απαιτήσεων. Συνήθως, δημιουργούμε αρχικά 4 βασικές λίστες:

- Το Product Backlog που περιέχει όλες τις απαιτήσεις του έργου.
- Το Sprint Backlog που περιέχει τις απαιτήσεις που έχουν συμπεριληφθεί στο τρέχον sprint.
- Τη λίστα In Progress που περιλαμβάνει τις απαιτήσεις του Sprint των οποίων οι εργασίες είναι σε εξέλιξη.
- Τη λίστα Complete η οποία περιέχει τις απαιτήσεις του sprint που έχουν ολοκληρωθεί.

Στο τέλος του Sprint μετονομάζουμε τη λίστα Complete σε Sprint1_Complete και αρχικοποιούμε ξανά τις λίστες Sprint Backlog, In Progress και Complete ώστε η λίστα Sprint Backlog να περιέχει τις απαιτήσεις του 2ου Sprint, ενώ οι λίστες να είναι In Progress και Complete να είναι κενές.

Η συνολική ροή της εργασίας παρουσιάζεται στην επόμενη Εικόνα.



Κριτήριο αξιολόγησης 14

Εξηγήστε τη διαφορά μεταξύ sprint, επαύξησης (increment), και αποδέσμευσης (release).

Απάντηση/Λύση

Η καρδιά του Scrum είναι το sprint, μια επανάληψη με χρονική διάρκεια ενός μήνα ή μικρότερη κατά τη διάρκεια του οποίου δημιουργείται μια ολοκληρωμένη (done) επαύξηση του προϊόντος, που μπορεί να χρησιμοποιηθεί και είναι δυνητικά απελευθερώσιμη (releasable).

Σύμφωνα με τον οδηγό του Scrum, στο τέλος ενός Sprint, η νέα επαύξηση πρέπει να είναι ολοκληρωμένη (Done), που σημαίνει ότι πρέπει να είναι σε κατάσταση χρήσης και να πληροί τον ορισμό της ομάδας Scrum για τα κριτήρια ολοκλήρωσης.

Το αν θα απελευθερωθεί η επαύξηση η όχι είναι στην αρμοδιότητα του product owner να αποφασίσει που σημαίνει ότι δεν είναι υποχρεωτικό να απελευθερωθούν όλες.

Κριτήριο αξιολόγησης 15

Ποιοι είναι οι στόχοι της ανασκόπησης του sprint;

Απάντηση/Λύση

Κατά τη διάρκεια της ανασκόπησης του sprint, η ομάδα συναντάται με τους επιχειρηματικούς χρήστες και τους πελάτες για να αναθεωρήσουν τι έχουν κάνει και να συζητήσουν για τους στόχους του επόμενου sprint. Θα εξετάσουν την τρέχουσα επαύξηση, η οποία συνήθως περιλαμβάνει την επίδειξη του λειτουργικού λογισμικού που κατασκεύασαν. Θα συζητήσουν επίσης το ανεκτέλεστο backlog και θα το ενημερώσουν για να δείξουν τα στοιχεία στα οποία πιθανότατα θα εργαστούν στο επόμενο sprint. Η ανασκόπηση του sprint δεν είναι για να δούμε τι συνέβη ή για να κάνουμε βελτιώσεις - γι' αυτό είναι σκοπός της συνάντησης για το Sprint Retrospective, αλλά για να αξιολογήσουμε την τρέχουσα επαύξηση και να συζητήσουμε για το επόμενο sprint. Κατά τη διάρκεια της ανασκόπησης του sprint δεν αποφασίζουμε για τα PBI που θα συμπεριλάβει το επόμενο sprint. Αυτό γίνεται στη συνάντηση για τον προγραμματισμό του sprint (sprint planning meeting).

Κριτήριο αξιολόγησης 16

Ποιες είναι οι βασικές ερωτήσεις που πρέπει να απαντηθούν από τα μέλη της ομάδας έργου κατά τη διάρκεια της καθημερινής συνάντησης Scrum;

Απάντηση/Λύση

Οι βασικές ερωτήσεις που θα πρέπει να απαντηθούν κατά τη διάρκεια της καθημερινής συνάντησης Scrum είναι:

- Ποιο ήταν το αποτέλεσμα της χθεσινής εργασίας;
- Ποιος είναι ο στόχος για σήμερα;
- Εφόσον έχουν προκύψει κάποια προβλήματα ή τυχόν ελλείψεις, ποια είναι αυτά;

Εισαγωγή στη λιτή διοίκηση - Η μέθοδος Kanban

Let the flow manage the processes, and
not let management manage the flow

Taiichi Ohno

Σύνοψη
Το πέμπτο κεφάλαιο παρουσιάζει την έννοια της *λιτής διοίκησης και του συστήματος Kanban. Η λιτή διοίκηση είναι ένα σύστημα παραγωγής το οποίο στοχεύει στην ελάττωση των περιττών δραστηριοτήτων που λαμβάνουν χώρα στις διαδικασίες παραγωγής και στη μείωση του κόστους παραγωγής με ταυτόχρονη αύξηση της ευελιξίας. Στη λιτή διοίκηση εστιαζόμαστε μόνο στις δραστηριότητες που παράγουν αξία, η ομαλή ροή των υλικών χωρίς ενδιάμεση αποθεματοποίηση, στον έλεγχο της παραγωγής μέσω της έλξης της από τη ζήτηση, και στη διαρκή αναζήτηση της τελειότητας μέσω δράσεων συνεχούς βελτίωσης.*
Επίσης θα παρουσιαστεί η μέθοδος Kanban η οποία αποτελεί μια συλλογή από καλές πρακτικές, βασίζεται στην οπτικοποίηση της ροής εργασιών που εκκρεμούν, βρίσκονται σε επεξεργασία, ή έχουν ολοκληρωθεί με σκοπό την καλύτερη οργάνωση της εργασίας.

5 Εισαγωγή στη λιτή διοίκηση – Η μέθοδος Kanban

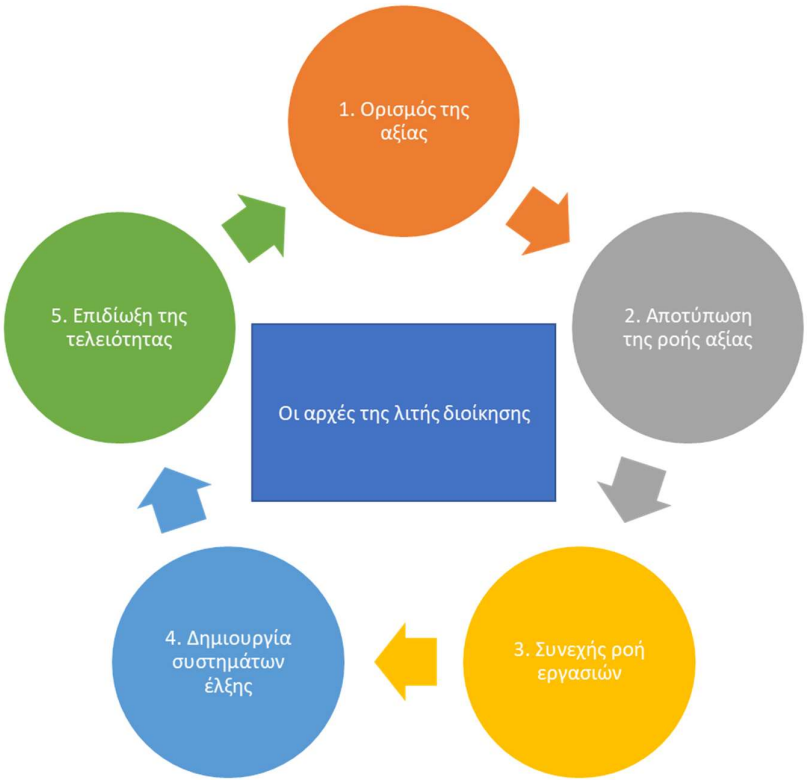
Ο γενικός όρος της **λιτής διοίκησης (lean management)** πρωτοεμφανίστηκε το 1988 και αποδίδεται στον Krafcik (1988) ως αποτέλεσμα της έρευνας που πραγματοποιήθηκε από το Ινστιτούτο Τεχνολογίας της Μασαχουσέτης σχετικά με τη σύγκριση της βιομηχανικής παραγωγής των αυτοκινήτων της Ιαπωνίας και του Δυτικού Κόσμου. Η βασική αντίληψη της λιτής παραγωγής είναι ότι αποτελεί μία φιλοσοφία όπου όταν εφαρμοστεί, μειώνει τον χρόνο απόκρισης από την παραγγελία του πελάτη μέχρι την παράδοση του τελικού προϊόντος μέσω εξάλειψης της σπατάλης κατά την παραγωγική διαδικασία. Η λιτή διοίκηση είναι ένα ευέλικτο σύστημα παραγωγής που χρησιμοποιεί λιγότερους πόρους σε σχέση με τα συμβατικά συστήματα. Στοχεύει σε υψηλότερη παραγωγικότητα, χαμηλότερο κόστος παραγωγής, μικρότερους χρόνους κύκλου και αυξημένη ποιότητα.

Αναπτύχθηκε κυρίως από την αυτοκινητοβιομηχανία Toyota (Toyota Production System – TPS) (Ohno, & Bodek, 2019), ξεκινώντας από τα μέσα του προηγούμενου αιώνα, μετά τον Β΄ Παγκόσμιο Πόλεμο, για να αντιμετωπίσει την έλλειψη πόρων που υπήρχε στην Ιαπωνία εξαιτίας των καταστροφών του πολέμου. Έγινε ευρύτερα γνωστή σε όλο τον κόσμο από το βιβλίο των Womack, Jones και Roos *The machine that changed the world*, στο οποίο περιγραφόταν ο τρόπος λειτουργίας και τα πλεονεκτήματα της ιαπωνικής αυτοκινητοβιομηχανίας (Womack et al. 1990).

Για να μπορέσουμε να κατανοήσουμε τη φιλοσοφία της λιτής διοίκησης πρέπει να αναφερθούμε στις αρχές της λιτής διοίκησης όπως αυτές διατυπώθηκαν από τους Womack και Jones (1996). Η λιτή διοίκηση στηρίζεται στις εξής πέντε αρχές:

- **Αξία – Value.** Είναι ο καθορισμός της αξίας από τη μεριά του πελάτη, δηλαδή η αναγνώριση της αξίας που η υπηρεσία/προϊόν παρέχει στους πελάτες. Η αξία δημιουργείται από τον προμηθευτή της υπηρεσίας ή του προϊόντος, αλλά ο τελικός πελάτης είναι αυτός που θα καθορίσει την αξία του προϊόντος που προμηθεύεται. Ο πελάτης ζητάει προϊόντα και υπηρεσίες που θα ικανοποιήσουν τις ανάγκες του, χωρίς περιττές διαδικασίες, καθυστερήσεις, ελαττώματα κ.λπ.

- **Αποτύπωση Ροής αξίας - Value Stream Mapping**– Η αποτύπωση της ροής αξίας ώστε να επιτευχθεί η προκαθορισμένη αξία. Είναι όλα τα βήματα και οι ενέργειες που απαιτούνται, από την αγορά των πρώτων υλών έως την τελική παράδοση του προϊόντος στον πελάτη ή από την παραγγελία της υπηρεσίας έως τη χρήση της. Σκοπός είναι η εξάλειψη της σπατάλης, ο οποίος αποτελεί τον βασικό στόχο της λιτής διοίκησης, που μπορεί να επιτευχθεί μόνο με την πλήρη κατανόηση των βημάτων παραγωγής και των σημείων όπου παράγεται προστιθέμενη αξία για τον πελάτη. Οτιδήποτε περιττό που δεν προσθέτει αξία για τον πελάτη πρέπει να απαλείφεται. Σύμφωνα με τους Womack και Jones (2005), η αποτύπωση της ροής της αξίας σχεδόν πάντα καταλήγει στους παρακάτω τρεις τύπους ενεργειών:
    - Αυτές που προσθέτουν αξία.
    - Αυτές που δεν προσθέτουν αξία αλλά είναι απαραίτητες.
    - Αυτές που είναι απολύτως περιττή σπατάλη (muda στα ιαπωνικά) και θα πρέπει να εξαλειφθούν.
- **Ροή – Flow**. Στη λιτή διοίκηση τα βήματα της ροής αξίας πρέπει να εκτελούνται με τέτοιο τρόπο ώστε η ροή των εργασιών να είναι συνεχής, χωρίς καθυστερήσεις και διαλλείματα. Εάν σε κάποιο σημείο η ροή εργασιών διακοπεί, τότε δημιουργείται αναμονή, η οποία είναι μη παραγωγικός χρόνος και συνεπώς θεωρείται σπατάλη. Αυτή η απαίτηση της λιτής διοίκησης επιφέρει μεγάλες μειώσεις στους χρόνους παραγωγής, σε σχέση με παραδοσιακές διεργασίες που οργανώνουν την παραγωγή σε στάδια. Ένα σημαντικό εμπόδιο για τη δημιουργία ομαλής ροής εργασίας είναι τα σημεία συμφόρησης (bottlenecks) που υπάρχουν σε κάθε διεργασία. Τα σημεία συμφόρησης μπορεί να δημιουργηθούν από έλλειψη χωρητικότητας σε ένα συγκεκριμένο στάδιο παραγωγής, από αναμονή για πρώτες ύλες ή υπηρεσίες που παρέχουν εξωτερικοί προμηθευτές κ.λπ.
- **Σύστημα έλξης – Pull**. Σε ένα σύστημα έλξης τα προϊόντα δημιουργούνται μετά από την απαίτηση του πελάτη και όχι ως πρόβλεψη παραγωγής προϊόντων. Στόχος είναι η ελαχιστοποίηση των αποθεμάτων.
- **Τελειότητα – Perfection.** Αποτελεί θεμελιώδη αρχή της λιτής διοίκησης. Με τη λέξη τελειότητα δεν εννοούμε αποκλειστικά την ποιότητα του προϊόντος, αλλά και την εξάλειψη της σπατάλης και τη συνεχή βελτιστοποίηση των διεργασιών παραγωγής, των ικανοτήτων των εργαζομένων αλλά και όλης της επιχείρησης γενικότερα. Αποτελεί τη φιλοσοφία της επιχείρησης για συνεχή βελτίωση και απόκτηση ανταγωνιστικού πλεονεκτήματος σε σχέση με τους ανταγωνιστές.



Εικόνα 5.1: *Οι αρχές της λιτής διοίκησης*

Συνεπώς με τη λιτή διοίκηση:

- Μειώνουμε τον χρόνο εκτέλεσης των διεργασιών
- Βελτιώνουμε τον χρόνο παράδοσης των προϊόντων ή των υπηρεσιών
- Μειώνουμε ή εξαλείφουμε την πιθανότητα δημιουργίας ελαττωμάτων
- Μειώνουμε τα επίπεδα των αποθεμάτων απογραφής και
- Βελτιστοποιούμε τη χρήση των πόρων.

Όπως αναφέραμε, βασικό ρόλο στη λιτή διοίκηση αποτελεί η έννοια της δημιουργίας αξίας (value creation). Ανάλογα με τον τύπο των επιχειρηματικών διεργασιών και τον βιομηχανικό κλάδο, ο ορισμός της «δημιουργίας αξίας» είναι διαφορετικός, αφού σχετίζεται άμεσα με την αντίληψη του πελάτη σχετικά με το προϊόν ή τις υπηρεσίες, για τις οποίες είναι διατεθειμένος να πληρώσει.

Θα θυμίσουμε ότι μια παραγωγική διεργασία είναι μια σειρά βημάτων/δραστηριοτήτων, η οποία μετατρέπει τις εισροές σε αποτελέσματα (προϊόντα ή υπηρεσίες) χρησιμοποιώντας πόρους. Σε μια διεργασία, οι δραστηριότητες αυτές μπορούν να ταξινομηθούν σε τρεις διαφορετικούς τύπους. Αυτοί είναι:

- Δραστηριότητες μη-προστιθέμενης αξίας: Αυτές οι δραστηριότητες δεν προσθέτουν αξία στα προϊόντα που παράγουμε. Πολλές φορές αποτελούν τα περιττά βήματα, δηλαδή αυτά που θα πρέπει να εξαλείψουμε. Συνήθως ένας πελάτης δεν πληρώνει πρόθυμα τις δαπάνες που σχετίζονται με αυτές τις δραστηριότητες. Επιπλέον, εάν δοθεί έμφαση σε αυτές, δημιουργείται δυσαρέσκεια στους πελάτες.
- Δραστηριότητες προστιθέμενης αξίας: Αυτές οι δραστηριότητες προσθέτουν αξία στη διαδικασία και είναι απαραίτητες για την παραγωγή των προϊόντων ή των υπηρεσιών.

- Βοηθητικές δραστηριότητες στις δραστηριότητες προστιθέμενης αξίας: Αυτές οι δραστηριότητες δεν προσθέτουν αξία στον πελάτη. Είναι όμως απαραίτητες για τη συνεχή λειτουργία των διεργασιών.

Σε οποιαδήποτε παραγωγική διεργασία, σχεδόν το 80% - 85% των δραστηριοτήτων είναι δραστηριότητες μη προστιθέμενης αξίας. Ο στόχος της λιτής στρατηγικής προσέγγισης είναι να εντοπίσει αυτές τις διεργασίες και με τη χρήση εργαλείων να τις εξαλείψει ή να τις μειώσει.

5.1 Οι αρχές της λιτής διοίκησης στην ανάπτυξη λογισμικού

Η λιτή διοίκηση στην ανάπτυξη λογισμικού συνοψίζεται σε επτά βασικές αρχές. Οι βασικές αυτές αρχές είναι οι ακόλουθες (Poppendieck, M., & Poppendieck, T. 2003).:

- **Η εξάλειψη της σπατάλης (waste management)**. Τα περιττά είναι οτιδήποτε δεν προσθέτει αξία σε ένα προϊόν ή αξία με τον τρόπο που την αντιλαμβάνεται ο πελάτης. Στη λιτή φιλοσοφία, η έννοια των περιττών είναι κομβικής σημασίας και μπορεί να έχει διαφορετικές μορφές όπως θα αναλυθεί σε επόμενες παραγράφους.

- **Ενίσχυση της μάθησης**. Η ανάπτυξη νέων προϊόντων είναι μια πράξη καινοτομίας, ενώ η παραγωγή είναι μια άσκηση μείωσης της διακύμανσης και ενίσχυσης της συμμόρφωσης με τις προδιαγραφές. Για το λόγο αυτό η λιτή προσέγγιση στην ανάπτυξη και η λιτή προσέγγιση στην παραγωγή απαιτούν εφαρμογή διαφορετικών πρακτικών. Η ανάπτυξη προϊόντων ή λογισμικού είναι, λοιπόν, μια παρόμοια διαδικασία με τη διαδικασία μάθησης με την πρόσθετη πρόκληση ότι τα προϊόντα ή το λογισμικό είναι συνήθως πιο περίπλοκα. Συνεπώς, η καλύτερη προσέγγιση για τη βελτίωση ενός περιβάλλοντος ανάπτυξης είναι η ενίσχυση της μάθησης.

- **Αποφασίζουμε όσο το δυνατόν αργότερα**. Η καθυστερημένη λήψη αποφάσεων είναι μια τακτική αποτελεσματική, σε τομείς που εμπεριέχουν αβεβαιότητα. Για παράδειγμα, σε μια ευμετάβλητη αγορά, η διατήρηση όλων των επιλογών σχεδιασμού ανοιχτών, είναι μια χρήσιμη τακτική διότι δεν μας δεσμεύει νωρίς και μας επιτρέπει να αποφασίσουμε όταν όλα τα δεδομένα γίνουν γνωστά.

- **Παραδίδουμε το προϊόν όσο το δυνατόν γρηγορότερα.** Η ταχύτητα παράδοσης είναι πάντα σημαντική ιδιαίτερα σε περιπτώσεις που η αγορά είναι ευμετάβλητη και επιπλέον δίνει στον πελάτη τη δυνατότητα να συμμετάσχει άμεσα στην ανάπτυξη του προϊόντος ανατροφοδοτώντας την ομάδα ανάπτυξης με σχόλια. Οι μικροί χρονικά κύκλοι ανάπτυξης διασφαλίζουν ότι το τελικό προϊόν είναι αυτό που πραγματικά χρειάζονται οι πελάτες.

- **Ενδυνάμωση της ομάδας παραγωγής**. Είναι δεδομένο ότι κανείς δεν καταλαβαίνει καλύτερα τις λεπτομέρειες τις εργασίας από τους ανθρώπους που κάνουν πραγματικά τη δουλειά. Η συμμετοχή των μελών της ομάδας έργου στη λήψη των αποφάσεων είναι θεμελιώδης για την επίτευξη του βέλτιστου αποτελέσματος.

- **Δημιουργία ακεραιότητας στο υπό ανάπτυξη σύστημα**. Ένα σύστημα θεωρείται ότι έχει ακεραιότητα όταν ένας χρήστης είναι ικανοποιημένος από τη λειτουργία του. Επίσης το μερίδιο αγοράς είναι ένα ενδεικτικό μέτρο της αντιληπτής ακεραιότητας για τα προϊόντα, επειδή μετρά την αντίληψη των πελατών με την πάροδο του χρόνου. Εννοιολογική ακεραιότητα (conceptual integrity) σημαίνει ότι τα κεντρικά στοιχεία του συστήματος συνεργάζονται ως ένα ομαλό, συνεκτικό σύνολο.

- **Η εστίαση στη συνολική εικόνα του συστήματος**. Η ακεραιότητα σε πολύπλοκα συστήματα απαιτεί βαθιά εξειδίκευση σε πολλούς διαφορετικούς τομείς. Ένα από τα πιο δυσεπίλυτα προβλήματα με την ανάπτυξη προϊόντων είναι ότι οι ειδικοί σε οποιονδήποτε τομέα (π.χ. βάση δεδομένων ή GUI) έχουν την τάση να μεγιστοποιούν την απόδοση του μέρους του προϊόντος που αντιπροσωπεύει τη δική τους ειδικότητα, αντί να εστιάζονται στη συνολική απόδοση του συστήματος.

Ομοιότητες και διαφορές μεταξύ ευέλικτης και της λιτής προσέγγισης

Ένα βασικό ερώτημα που προκύπτει είναι ποιες είναι οι ομοιότητες και οι διαφορές μεταξύ της ευέλικτης και της λιτής προσέγγισης. Είναι γεγονός ότι και οι δύο μέθοδοι εστιάζονται στον πελάτη και έχουν ως στόχο την ταχεία ανάπτυξη προϊόντων και υπηρεσιών υψηλής ποιότητας. Ωστόσο, η κατανόηση των εννοιών δεν είναι ξεκάθαρη για πολλούς και μπορεί να προκαλέσει σύγχυση. Η σαφής κατανόηση των εννοιών μπορεί να διασφαλίσει ότι οι ομάδες υιοθετούν πλήρως τις αρχές και τις συνοδευτικές πρακτικές της κάθε φορά επιλεγμένης μεθοδολογίας τους, έτσι ώστε να μπορούν να δημιουργήσουν μια σταθερή βάση για την εφαρμογή τους. Αλλά πρώτα, ας προσπαθήσουμε να δούμε τις ομοιότητες και να κατανοήσουμε τις διαφορές μεταξύ τους.

Οι βασικές ομοιότητες των προσεγγίσεων είναι ότι τόσο η λιτή όσο και η ευέλικτη προσέγγιση εστιάζονται στην:

- εστίαση στον πελάτη,
- συνεχή βελτίωση της διεργασίας ανάπτυξης,
- συνεχή παράδοση του προϊόντος/λογισμικού στον πελάτη,

Αντίστοιχα οι διαφορές συνοψίζονται στα παρακάτω:

- η ευέλικτη προσέγγιση δεν εστιάζεται στην εξάλειψη των περιττών η οποία αποτελεί τη βασική προτεραιότητα της λιτής προσέγγισης,
- στη λιτή διοίκηση η εστίαση στον πελάτη επιτυγχάνεται μέσω του εντοπισμού της αξίας του προϊόντος καθώς και με την αντιληπτή ποιότητα του προϊόντος, σε αντίθεση με τις ευέλικτες μεθόδους που η εστίαση στον πελάτη γίνεται κατά κύριο λόγο μέσω της συμμετοχής του πελάτη στην ανάπτυξη του προϊόντος,
- στη λιτή διοίκηση στόχος είναι η σταθερή ροή εργασίας, ενώ στην ευέλικτη διοίκηση η ανάπτυξη γίνεται μέσω επαναλήψεων (sprint),
- η συνεχής βελτίωση στη λιτή διοίκηση επιτυγχάνεται με την εξάλειψη των περιττών σε αντίθεση με την ευέλικτη προσέγγιση όπου η βελτίωση επιτυγχάνεται με τις συνεχείς ανασκοπήσεις στο τέλος κάθε επανάληψης αλλά και στο τέλος του έργου, καθώς και
- ο στόχος μιας ευέλικτης μεθόδου είναι να παραδώσει στον πελάτη ένα προϊόν που τον ικανοποιεί, σε αντίθεση με τη λιτή διοίκηση όπου στόχος είναι να παραδοθεί ένα προϊόν το οποίο να έχει τα ελάχιστα περιττά χαρακτηριστικά.

	Ευέλικτη προσέγγιση	**Λιτή προσέγγιση**
Μεθοδολογία	Εστιάζεται στη συνεχή βελτίωση	Εστιάζεται στην εξάλειψη των περιττών
Αξία στον πελάτη	Ο πελάτης συμμετέχει στην ανάπτυξη των προϊόντων και κάνει ιεράρχηση των απαιτήσεων	Δίνει έμφαση στην ιεράρχηση της αξίας σύμφωνα με τις απαιτήσεις του πελάτη και στην τελική ποιότητα του προϊόντος
Παράδοση προϊόντος	Η ανάπτυξη του προϊόντος και η παράδοση αυτού γίνεται σε περιοδικούς κύκλους (επαναλήψεις)	Δίνεται έμφαση στη βέλτιστη χρήση των πόρων και στη συνεχή ροή εργασίας
Συνεχής βελτίωση	Συνεχής αξιολόγηση και ανατροφοδότηση με σχόλια από τον πελάτη	Συνεχής βελτίωση μέσω της εξάλειψης των περιττών
Στόχος	Ο τελικός στόχος είναι η ικανοποίηση του πελάτη	Ο τελικός στόχος είναι η εξάλειψη όλων εκείνων των παραγόντων που δεν προσθέτουν αξία στο τελικό προϊόν

Πίνακας 5.1: *Σύγκριση ευέλικτης και λιτής προσέγγισης*

5.2 Η εξάλειψη της σπατάλης (waste management)

Όπως προαναφέρθηκε, η λιτή διοίκηση σχεδιάστηκε και εφαρμόστηκε από την εταιρεία Toyota στο σύστημα παραγωγής της, το οποίο είναι γνωστό και ως Toyota Production System. Σύμφωνα με το σύστημα αυτό, εντοπίζονται συνολικά οι παρακάτω μορφές σπατάλης:

Ως σπατάλη μπορεί να οριστεί οποιαδήποτε διεργασία η οποία δεν προσδίδει την αναμενόμενη προστιθέμενη αξία στο παραγόμενο προϊόν ή παραγόμενη υπηρεσία, καθυστερεί την ροή της διεργασίας και καταναλώνει περισσότερους από τους αναγκαίους πόρους (υλικούς ή/και άυλους). Ένα χαρακτηριστικό παράδειγμα της σπατάλης είναι τα έτοιμα προϊόντα τα οποία παραμένουν στο ενδιάμεσο στάδιο μιας παραγωγικής διεργασίας και δεν προχωρούν σε επόμενο στάδιο της επεξεργασίας διότι είναι ελαττωματικά.

Οι οκτώ διαφορετικές μορφές σπατάλης παρουσιάζονται στην Εικόνα 5.2.

Εικόνα 5.2: *Οι οκτώ διαφορετικές μορφές της σπατάλης*

Ενδιαφέρον παρουσιάζει να δούμε πως οι γενικές κατηγορίες σπατάλης αντιστοιχούν στην σπατάλη στην περίπτωση των έργων ανάπτυξης λογισμικού. Ο Πίνακας 5.2 παρουσιάζει συγκριτικά τις κατηγορίες σπατάλης στη γενική περίπτωση και στην ανάπτυξη λογισμικού.

Σπατάλη στην κατασκευή προϊόντων	**Σπατάλη στην ανάπτυξη λογισμικού**
Αποθέματα (Inventory)	Ημιτελής δουλειά (Partially Done Work)
Υπερπαραγωγή (Over-production)	Υπερβάλλουσα λειτουργικότητα (Extra Features)
Αναμονές (Waiting)	Καθυστερήσεις-Αναμονές (Delays)

Σπατάλη στην κατασκευή προϊόντων	Σπατάλη στην ανάπτυξη λογισμικού
Μεταφορές (Transportation) και κινήσεις στην εργονομία παραγωγής (motion)	Μεταφορά εργασίας (Handoffs) και εναλλαγή εργασιών (task switching)
Ελαττωματικά προϊόντα (Defects)	Ελαττωματικός κώδικας (Defects)
Αναποτελεσματικοί μέθοδοι εργασίας (extra processes)	Αναποτελεσματικός κύκλος ζωής λογισμικού και διεργασίες
Υποαπασχόληση/υπεραπασχόληση (Underutilization/Overutilization)	Υποαπασχόληση/υπεραπασχόληση (Underutilization/Overutilization)

Πίνακας 5.2: *Η σπατάλη στην κατασκευή των προϊόντων και στο λογισμικό*

Το μερικώς ολοκληρωμένο λογισμικό (Partially Done Work) έχει την τάση να γίνεται παρωχημένο και εμποδίζει την ανάπτυξη άλλων εξελίξεων που μπορεί να χρειαστεί να γίνουν. Επιπλέον, εφόσον δεν έχει ενσωματωθεί στο παραγωγικό περιβάλλον δεν είναι ακόμη γνωστά τα πιθανά του προβλήματα και η πραγματική του αξία για τον πελάτη. Τέλος, από οικονομικής απόψεως, δημιουργεί οικονομικούς κινδύνους διότι δεν μπορεί να θεωρηθεί ως μια ολοκληρωμένη επένδυση.

Μπορεί να φαίνεται καλή ιδέα ή ανώδυνο να έχει το λογισμικό κάποια επιπλέον χαρακτηριστικά ή νέες τεχνικές δυνατότητες - υπερβάλλουσα λειτουργικότητα, αλλά κατ' ουσίαν αποτελεί μια σημαντική σπατάλη. Ο λόγος είναι ότι κάθε κομμάτι κώδικα απαιτεί σχεδιασμό, ανάπτυξη, έλεγχο και ενσωμάτωση καθ' όλη τη διάρκεια ζωής του συστήματος. Κάθε κομμάτι κώδικα αυξάνει την πολυπλοκότητα και αποτελεί ένα πιθανό σημείο αποτυχίας. Συνεπώς, εάν ο κώδικας δεν είναι απαραίτητος άμεσα, η ανάπτυξή του αποτελεί σπατάλη.

Μια από τις μεγαλύτερες σπατάλες στην ανάπτυξη λογισμικού είναι συνήθως η αναμονή για διάφορα γεγονότα. Παραδείγματα σπατάλης, της κατηγορίας «αναμονή» είναι οι καθυστερήσεις στην έναρξη ενός έργου, στη στελέχωσή του, η ανάγκη για τεκμηρίωση απαιτήσεων, οι αναθεωρήσεις και οι εγκρίσεις, καθώς και οι έλεγχοι λογισμικού. Οι καθυστερήσεις αυτές είναι πολύ συνηθισμένες και μπορεί να φαίνεται αντιφατικό ότι αποτελούν σπατάλη. Όμως οι καθυστερήσεις αυτές εμποδίζουν τον πελάτη να πραγματοποιήσει την αναμενόμενη «αξία» που επιθυμεί το συντομότερο δυνατό.

Το μέγεθος της σπατάλης που προκαλείται από ένα ελάττωμα (defect) είναι το γινόμενο των συνεπειών του ελαττώματος επί τον χρόνο που δεν ανιχνεύεται. Ένα κρίσιμο ελάττωμα που ανιχνεύεται μέσα σε τρία λεπτά προκαλεί μικρότερη σπατάλη σε σχέση με ένα μικρό ελάττωμα που δεν ανακαλύπτεται για εβδομάδες. Ο τρόπος για να μειώσουμε τον αντίκτυπο των ελαττωμάτων είναι να τα ανιχνεύουμε αμέσως μόλις εμφανιστούν. Ο τρόπος μείωσης της σπατάλης λόγω ελαττωμάτων είναι η άμεση δοκιμή, η συχνή ενσωμάτωση και η απελευθέρωση του λογισμικού στην παραγωγή το συντομότερο δυνατό.

Η ανάπτυξη λογισμικού απαιτεί βαθιά συγκέντρωση από τον προγραμματιστή/αναλυτή. Η εναλλαγή εργασιών (task switching) εκτός του να αποσπά τον εργαζόμενο για το αντίστοιχο χρονικό διάστημα, μπορεί να οδηγήσει και σε υποδεέστερο ποιοτικά προϊόν. Επιπλέον η εναλλαγή εργασιών καθυστερεί την παράδοση του προϊόντος στον πελάτη. Μερικοί τρόποι που θα μπορούσαν να βοηθήσουν στην επίλυση του προβλήματος είναι οι ακόλουθοι (Poppendieck, & Poppendieck, 2007):

- Εάν πρέπει να υλοποιήσουμε ταυτόχρονα πολλά έργα θα πρέπει να ελαχιστοποιήσουμε τον αριθμό των εναλλαγών.
- Εάν δεν είναι εφικτό να εξαλείψουμε τις εναλλαγές, αναθέτουμε την εργασία αυτή κάθε εβδομάδα σε ένα διαφορετικό μέλος της ομάδας, ώστε να ελαχιστοποιήσουμε την όχληση της ομάδας.
- Εξαλείφουμε τις μη αναγκαίες εργασίες και ό,τι γενικότερα δεν προσφέρει αξία.
- Εξασφαλίζουμε ότι όλες οι γνώσεις που είναι απαραίτητες για την ολοκλήρωση της ανατεθείσας εργασίας υπάρχουν στο κατάλληλο προσωπικό. Αυτό μπορεί να αποτρέψει την ανάγκη εναλλαγής εργασιών αποτρέποντας τα εμπόδια που προκαλούνται από την έλλειψη πληροφοριών ή γνώσης.

Η μεταφορά εργασίας και η παράδοση ημιτελούς δουλειάς σε ένα άλλο εργαζόμενο αναφέρεται με τον αγγλικό όρο handoff. Το φαινόμενο του handoff δημιουργεί σπατάλη αφού η μεταφορά της εργασίας από ένα εργαζόμενο στον επόμενο οδηγεί σε απώλεια γνώσης. Κυρίως αναφερόμαστε στη άρρητη γνώση (tacit knowledge), που είναι εκείνη η γνώση η οποία δεν μπορεί να καταγραφεί εύκολα και παραμένει κτήμα εκείνου που τη δημιούργησε. Οι Poppendieck αναφέρουν ότι κάθε φορά που η εργασία μεταφέρεται σε ένα επόμενο άτομο μεταφέρεται μόνο το 50% της γνώσης που είναι διαθέσιμη. Αυτό σημαίνει ότι:

- Μόνο το 25% της γνώσης είναι διαθέσιμο μετά από δύο μεταφορές
- Μόνο το 12% της γνώσης είναι διαθέσιμο μετά από τρεις μεταφορές
- Μόνο το 6% της γνώσης είναι διαθέσιμο μετά από τέσσερις μεταφορές
- Κ.λπ.

Μέθοδοι αποφυγής αυτής της διαρροής γνώσης θα μπορούσαν να είναι οι παρακάτω :

- Μείωση του αριθμού των handoffs. Είναι αποδοτικότερο να ολοκληρώνεται η δουλειά από εκείνον που την έχει ξεκινήσει.
- Βελτίωση της προσωπικής επικοινωνίας, ώστε να γίνεται η μετάδοση της άρρητης γνώσης ευκολότερα.
- Στη γραπτή επικοινωνία, εκτός των τυπικών και λεπτομερών τεκμηρίων, θα μπορούσαν να προστεθούν λιγότερο τυπικά τεκμήρια που θα περιλαμβάνουν τα πιο σημαντικά σημεία. Π.χ. κάποιες συμβουλές για τι χρειάζεται να ξέρει κανείς σε μορφή video

Οι αναποτελεσματικοί μέθοδοι εργασίας και η γραφειοκρατία είναι δύο βασικές αιτίες δημιουργίας σπατάλης. Είναι βέβαιο ότι η γραφειοκρατία καταναλώνει πόρους και επιβραδύνει τον χρόνο απόκρισης. Από την άλλη πλευρά πολλές διαδικασίες ανάπτυξης λογισμικού απαιτούν τη δημιουργία εγγράφων για την επικοινωνία με τον πελάτη ή για την ιχνηλασιμότητα του λογισμικού ή για τη λήψη έγκρισης για μια αλλαγή. Το βασικό ερώτημα είναι κατά πόσο οι μέθοδοι εργασίας που χρησιμοποιούμε προσθέτουν αξία για τον πελάτη.

Επίσης, οι δραστηριότητες διαχείρισης δεν προσθέτουν άμεσα αξία σε ένα προϊόν, αλλά προκαλούν σπατάλη. Για παράδειγμα, μια διαδικασία ιεράρχησης των απαιτήσεων ενός έργου ή ένα σύστημα απελευθέρωσης λογισμικού (software releases). Η ελαχιστοποίηση της σπατάλης απαιτεί η ποσότητα της ημιτελούς εργασίας να είναι η ελάχιστη, απαίτηση που είναι σε αντίθεση με τη λογική της διαχείρισης των αποδεσμεύσεων, που απαιτεί την απελευθέρωση του λογισμικού σε τακτά χρονικά διαστήματα (π.χ. κάθε εξάμηνο).

5.3 Η αποτύπωση της ροής αξίας

Ένα βασικό εργαλείο για την κατανόηση της σπατάλης είναι η αποτύπωση της ροής αξίας (Value Stream Mapping - VSM). Η αποτύπωση της ροής αξίας, γνωστή και ως «αποτύπωση ροής υλικού και πληροφοριών», είναι μια μέθοδος της λιτής διοίκησης που αποσκοπεί στην ανάλυση της τρέχουσας κατάστασης και τον σχεδιασμό μιας μελλοντικής κατάστασης για τις εργασίες/βήματα που απαιτούνται για την κατασκευή ενός προϊόντος ή την παροχή μιας υπηρεσίας από την έναρξη της συγκεκριμένης διαδικασίας και μέχρι να φτάσει στον πελάτη. Είναι ένα οπτικό βοήθημα που εμφανίζει όλα τα κρίσιμα βήματα σε μια συγκεκριμένη διαδικασία ενώ ταυτόχρονα ποσοτικοποιεί τον χρόνο και τον όγκο που απαιτείται σε κάθε βήμα (Rother & Shook, 2003). Η αποτύπωση της ροής αξίας γίνεται με τη δημιουργία του αντίστοιχου διαγράμματος το οποίο αποτελεί μια εξειδικευμένη μορφή διαγράμματος ροής.

Υπό την οπτική γωνία του πελάτη προσπαθούμε να μεγιστοποιήσουμε την παρεχόμενη αξία. Η αξία μπορεί να είναι:

- Η τιμή του προϊόντος

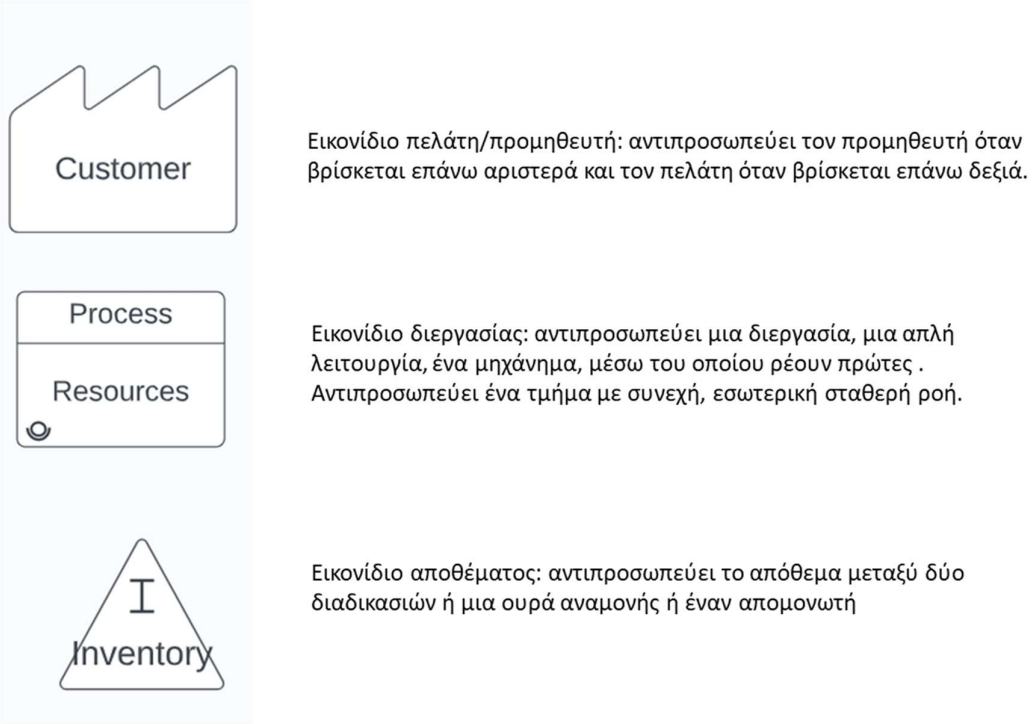
- Η ποιότητα
- Η αξιοπιστία παράδοσης ή λειτουργίας
- Η ταχεία ανταπόκριση στις μεταβαλλόμενες ανάγκες
- Κ.λπ.

Αντίστοιχα, η ροή αναφέρεται σε υλικά σε ένα σύστημα παραγωγής, σε σχεδιαστικά έγγραφα σε μια υπηρεσία σχεδιασμού, σε λογισμικό σε μια ομάδα ανάπτυξης λογισμικού, κ.λπ.

Η ροή αξίας πάντα ξεκινάει και τελειώνει με τον πελάτη. Σε ένα περιβάλλον ανάπτυξης (κώδικα/προϊόντος) το ρολόι ξεκινάει τη στιγμή που ο πελάτης κάνει ένα αίτημα/δίνει μια παραγγελία για ένα νέο χαρακτηριστικό του συστήματος). Αντίστοιχα, το ρολόι σταματά όταν το αίτημα του πελάτη έχει επιλυθεί επιτυχώς (για παράδειγμα το λογισμικό έχει μπει σε παραγωγική λειτουργία). Η αποτύπωση της ροής αξίας είναι ένα διαγνωστικό εργαλείο εύρεσης της σπατάλης και μας βοηθά να απαντήσουμε σε δύο βασικά ερωτήματα:

- Ποιος είναι ο πραγματικός χρόνος που απαιτείται για την επεξεργασία του αιτήματος του πελάτη από τη στιγμή που αυτό δημιουργείται έως τη στιγμή που αυτό μπαίνει σε παραγωγική λειτουργία. Ο χρόνος αυτός στα παραγωγικά συστήματα ονομάζεται ρυθμός παραγωγής (takt time). Ο ρυθμός παραγωγής είναι ο χρόνος κύκλου που απαιτείται σε ένα σύστημα παραγωγής για να ταιριάξει τον ρυθμό της παραγωγής με αυτόν της ζήτησης. Μερικές φορές λέγεται ότι είναι ο παλμός ενός λιτού συστήματος παραγωγής.
- Ποια είναι η αποδοτικότητα αυτής της ροής εργασίας, δηλαδή το ποσοστό του χρόνου που παράγει αξία σε σχέση με τον συνολικό χρόνο επεξεργασίας

Ένα διάγραμμα ροής αξίας χρησιμοποιεί πληθώρα συμβόλων για την απεικόνιση της παραγωγικής διεργασίας. Όμως τα δύο πιο βασικά είναι αυτά που απεικονίζουν τη διεργασία ή το βήμα αυτής καθώς και μια ουρά αναμονής ή σημείο αποθήκευσης (inventory). Στον Εικόνα 5.3 παρουσιάζονται ενδεικτικά κάποια από τα βασικά σύμβολα.

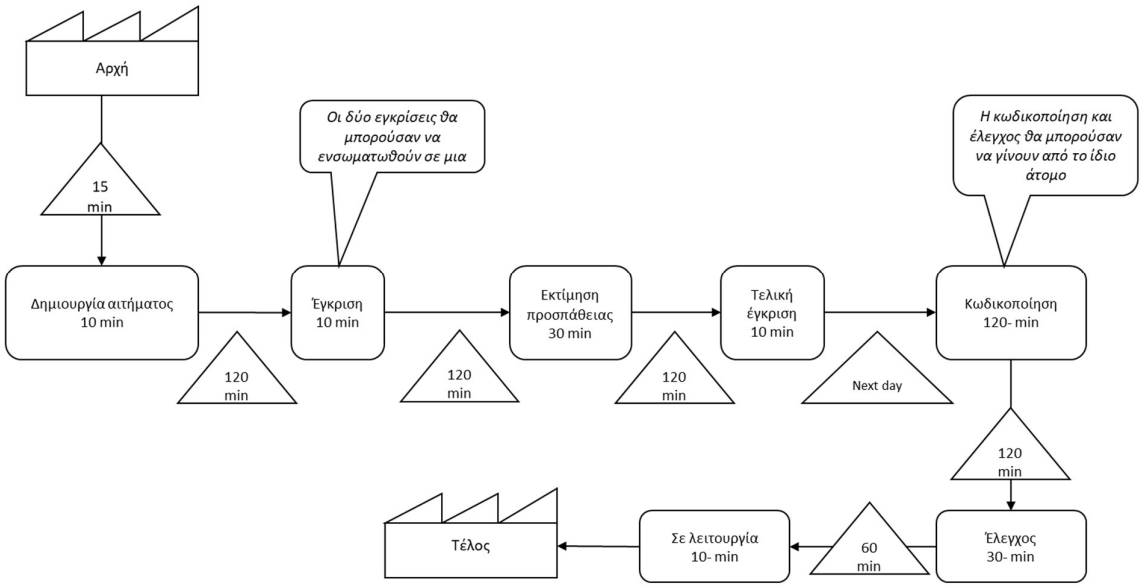


Εικόνα 5.3: *Τα βασικά σύμβολα του διαγράμματος ροής αξίας*

Τα τέσσερα βασικά βήματα για τη δημιουργία ενός διαγράμματος ροής αξίας είναι τα παρακάτω:

- **Επιλογή ροής αξίας** προς διερεύνηση. Η συνιστώμενη πρακτική είναι να απεικονίζεται μια διαδικασία ή ένα είδος διαδικασιών και όχι ένα μεμονωμένο γεγονός. Π.χ. θα μπορούσε να απεικονιστεί η ροή αξίας της δημιουργίας μιας νέας λειτουργίας σε υπάρχον σύστημα, ή η ανάπτυξη ενός νέου προϊόντος.
- **Επιλογή γεγονότων έναρξης – λήξης**. Η επιλογή των γεγονότων έναρξης και λήξης μπορεί να αλλάζει σημαντικά την οπτική γωνία. Για παράδειγμα, όταν θέλουμε να αναπτύξουμε ένα νέο προϊόν, αν επιλέξουμε ως γεγονός έναρξης τη στιγμή που ξενικά η ανάπτυξη αυτού, θα αφήσουμε εκτός της ανάλυσης της ροής αξίας όλα τα βήματα που αφορούν την αξιολόγησή του.
- **Εύρεση/Ανάδειξη του ιδιοκτήτη της ροής αξίας.** Η ροή αξίας για να σχεδιαστεί σωστά θα πρέπει να υπάρχει ένας υπεύθυνος/ιδιοκτήτης ο οποίος να μπορεί να αποφασίσει, να σχεδιάσει το έργο, κ.λπ. Ο ιδιοκτήτης είναι αυτός που αποφασίζει ποιες ροές αξίας θα αποτυπωθούν, τα προβλήματα που υπάρχουν και πώς η επίλυση αυτών επηρεάζει την αξία.
- Το επόμενο βήμα είναι η **συλλογή των αναγκαίων στοιχείων**, στατιστικών κατά κύριο λόγο, ώστε να αποτυπώσουμε την τρέχουσα κατάσταση.
- Αφού έχουμε κατανοήσει τη λειτουργία της επιχείρησης και των περιοχών που θέλουμε να αναλύσουμε καταγράφουμε την **τρέχουσα κατάσταση** και δημιουργούμε το αντίστοιχο διάγραμμα ροής αξίας. Στο σημείο αυτό θα πρέπει να αξιολογήσουμε την τρέχουσα κατάσταση, και να ενθαρρύνουμε την ομάδα ανάπτυξης ώστε να κάνει προτάσεις για μείωση της σπατάλης.
- Με βάση τα ευρήματα της αξιολόγησης κατασκευάζουμε το διάγραμμα ροής αξίας της **μελλοντικής κατάστασης.**

Στη συνέχεια παραθέτουμε ένα απλό παράδειγμα για πληρέστερη κατανόηση.

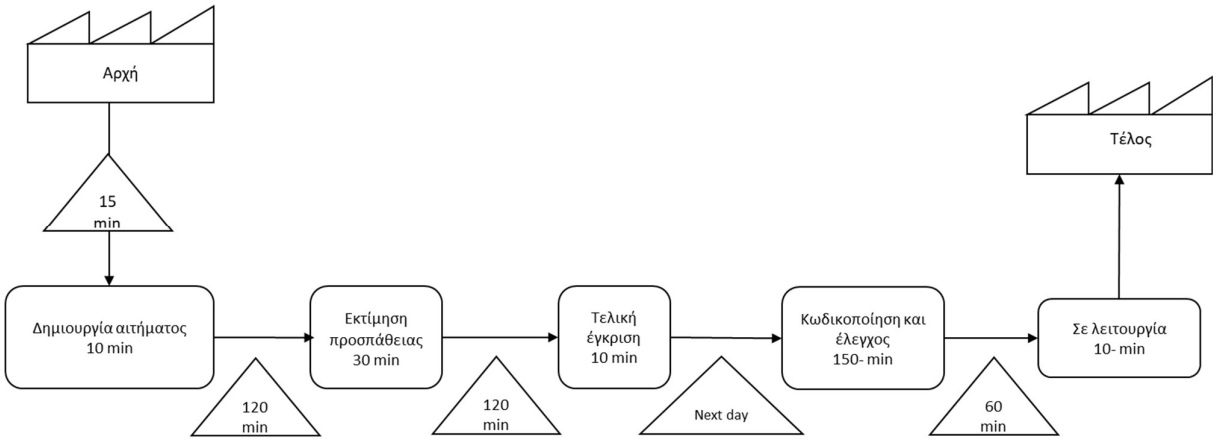


Εικόνα 5.4: *Παράδειγμα διαγράμματος ροής αξίας τρέχουσας κατάστασης*

Στην Εικόνα 5.4 παρατηρούμε ότι ο χρόνος επεξεργασίας (cycle time) ανέρχεται σε 220 min, ο χρόνος αναμονής σε 540 min ενώ η κωδικοποίηση ξεκινά πάντα την επόμενη ημέρα.

Θα μπορούσαμε να μειώσουμε αισθητά τους χρόνους αναμονής αν αφαιρούσαμε ένα από τα δύο βήματα που απαιτούνται για τις εγκρίσεις και αν πάντα η κωδικοποίηση και ο έλεγχος γινόταν από το ίδιο πρόσωπο.

Αν κάνουμε τις αναγκαίες αλλαγές προκύπτει η Εικόνα 5.5, στην οποία παρουσιάζονται μειωμένοι χρόνοι επεξεργασίας, αφού οι εργασίες δεν προσέθεταν αξία για τον πελάτη (δύο εγκρίσεις) αλλά και βελτιωμένους χρόνους αναμονής (Καραγιάννη, 2020).



Εικόνα 5.5: *Παράδειγμα διαγράμματος ροής αξίας μελλοντικής κατάστασης*

5.4 Η ενίσχυση της μάθησης

Σε ένα έργο ανάπτυξης λογισμικού, η ενίσχυση της μάθησης είναι η διαδικασία με την οποία αυξάνουμε την ικανότητα της ομάδας ανάπτυξης να μαθαίνει γρήγορα και αποτελεσματικά. Το πιο σημαντικό αντικείμενο μάθησης, σε ένα έργο ανάπτυξης λογισμικού, είναι η κατανόηση των αναγκών των χρηστών, αφού η αποτυχία κατανόησης των αναγκών του χρήστη μπορεί εύκολα να προκαλέσει την αποτυχία ενός έργου.

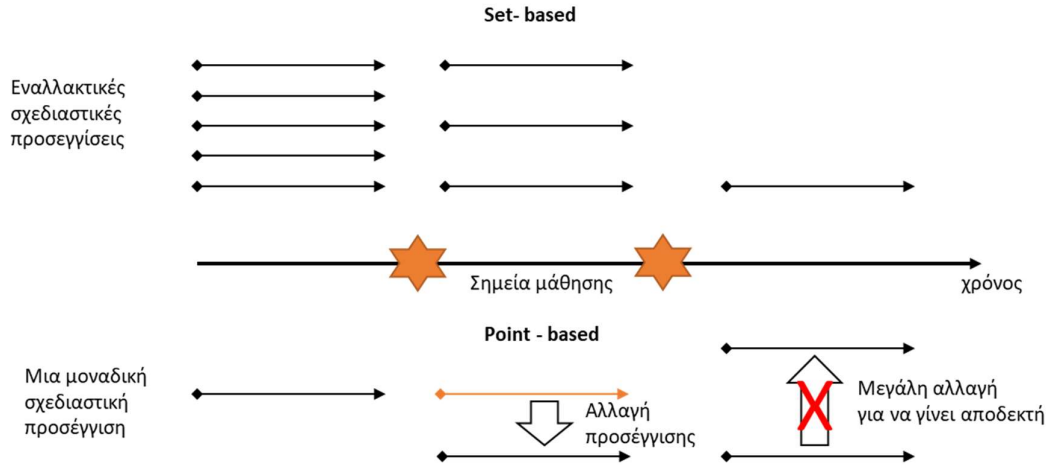
Στην ευέλικτη και λιτή ανάπτυξη λογισμικού, ο καλύτερος τρόπος για να επιτευχθεί αυτό είναι οι επαναληπτικοί κύκλοι ανάπτυξης. Σύμφωνα με τα λεγόμενα του Poppendieck: "Μια επανάληψη είναι μια χρήσιμη αύξηση λογισμικού που σχεδιάζεται, προγραμματίζεται, δοκιμάζεται, ενσωματώνεται και παραδίδεται σε σύντομο, σταθερό χρονικό πλαίσιο". Οι επαναλήψεις αυξάνουν τα επίπεδα ανατροφοδότησης μεταξύ της ομάδας ανάπτυξης και των χρηστών και είναι χρήσιμες για την επισήμανση προβλημάτων.

Οι επαναλήψεις επιτρέπουν την ανατροφοδότηση όλων των συμμετεχόντων με περιοδικό τρόπο. Η ανατροφοδότηση μπορεί να έχει πολλές μορφές όπως για παράδειγμα επίδειξη του υπ’ ανάπτυξη συστήματος ή επίδειξη ενός πρωτότυπου συστήματος.

Επίσης, όταν αντιμετωπίζουμε ένα δύσκολο πρόβλημα, είναι καλή πρακτική συχνά να σκεφτόμαστε στον χώρο λύσης αντί για συγκεκριμένες λύσεις. Η προσέγγιση αυτή ονομάζεται χώρος λύσεων ανάπτυξης (Set Based Development – SBD). Το Set-Based Development (SBD) είναι μια πρακτική που στοχεύει να έχουμε τις επιλογές σχεδιασμού ανοικτές για όσο το δυνατόν περισσότερο κατά τη διάρκεια της διαδικασίας ανάπτυξης. Αντί να επιλέγει εκ των προτέρων μια συγκεκριμένη λύση, το SBD εντοπίζει και διερευνά ταυτόχρονα πολλές επιλογές, εξαλείφοντας με την πάροδο του χρόνου τις χειρότερες επιλογές. Ενισχύει την ευελιξία στη διαδικασία σχεδιασμού δεσμεύοντας την ομάδα σε μια τεχνική λύση μόνο μετά από επικύρωση παραδοχών και υποθέσεων, γεγονός που παράγει καλύτερα τελικά αποτελέσματα. Υπό αυτή την οπτική γωνία, η ανάπτυξη ενός συστήματος μπορεί να περιγραφεί ως μια διαδικασία συνεχούς μετατροπής της αβεβαιότητας σε γνώση. Ανεξάρτητα από το πόσο καλά έχει αρχικά καθοριστεί και σχεδιαστεί ένα σύστημα, οι πραγματικές ανάγκες των πελατών και οι διαθέσιμες τεχνολογικές επιλογές είναι αβέβαιες όσο και εξελισσόμενες. Επομένως, η κατανόηση του τρόπου με τον οποίο ένα σύστημα πρέπει να υλοποιηθεί είναι κάτι δυναμικό που εξελίσσεται με την πάροδο του χρόνου.

Αντίθετα, ο σχεδιασμός που βασίζεται σε ένα σημείο (point based design) δεσμεύει την ομάδα σε μια συγκεκριμένη αρχιτεκτονική χρησιμοποιώντας τις διαθέσιμες ή τις παλαιότερες απαιτήσεις. Αυτή η προσέγγιση, συχνά οδηγεί σε εσφαλμένες υποθέσεις και λάθη που απαιτούν σημαντικές αλλαγές.

Η διαφορά αυτών των δύο προσεγγίσεων παρουσιάζεται στην Εικόνα 5.6.

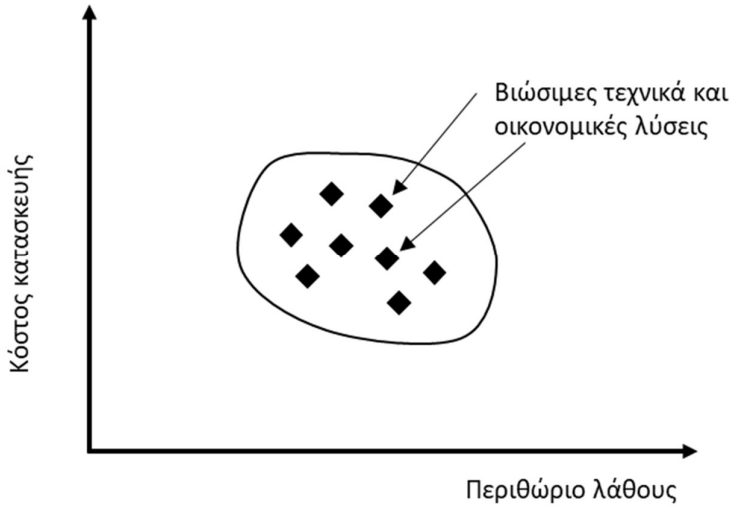


Εικόνα 5.6: *Η φιλοσοφία των πολλών και της μια σχεδιαστικής προσέγγισης*

Διαφορετικές επιλογές σχεδίασης έχουν διαφορετικές οικονομικές συνέπειες. Επομένως, η χρήση της φιλοσοφίας του SBD απαιτεί γνώση των μακροοικονομικών στόχων και των πλεονεκτημάτων που προσφέρει το υπ' ανάπτυξη σύστημα. Μερικοί από τους οικονομικούς παράγοντες που επηρεάζουν την τελική επιλογή είναι (Ward & Sobek II, 2014; Oosterwal, 2010):

- Κόστος ανάπτυξης
- Κόστος κατασκευής
- Απόδοση και αξιοπιστία
- Κόστος υποστήριξης
- Χρόνος ανάπτυξης
- Τεχνικοί κίνδυνοι

Αυτοί οι δείκτες βοηθούν να καταδειχθεί ποιες επιλογές σχεδίασης παρέχουν τα μεγαλύτερα οφέλη.



Εικόνα 5.7: *Οι βιώσιμες τεχνικά και οικονομικά λύσεις*

5.5 Η καθυστερημένη λήψη αποφάσεων

Μία από τις κοινές συζητήσεις στην ανάπτυξη λογισμικού αφορά την εφαρμογή προγνωστικών (predictive processes) ή προσαρμοστικών διεργασιών (adaptive processes). Στο προγνωστικό παράδειγμα η ανάπτυξη λογισμικού θα πρέπει να προσδιορίζεται λεπτομερώς πριν από την υλοποίηση, γιατί εάν δεν ικανοποιηθούν οι απαιτήσεις και ο σχεδιασμός σωστά, η υλοποίηση των τυχών αλλαγών θα κοστίσει πολύ αργότερα. Το προγνωστικό παράδειγμα, όπως αυτό του κύκλου ζωής καταρράκτη, μπορεί να λειτουργήσει καλά σε έναν κόσμο σταθερό και προβλέψιμο. Ωστόσο, όταν υπάρχει αβεβαιότητα σχετικά με τις απαιτήσεις των πελατών, ή για την εξέλιξη της τεχνολογίας, τότε πιο κατάλληλη είναι μια προσαρμοστική προσέγγιση, όπως οι ευέλικτες προσεγγίσεις.

Συνεπώς, οι ευέλικτες διαδικασίες ανάπτυξης λογισμικού μπορούν να θεωρηθούν ότι δημιουργούν επιλογές που επιτρέπουν την καθυστέρηση λήψης αποφάσεων, έως ότου οι ανάγκες των πελατών γίνουν πιο ξεκάθαρες και οι εξελισσόμενες τεχνολογίες να ωριμάσουν.

Το 1988 ο Harold Thimbleby δημοσίευσε μια εργασία στο IEEE Software με τίτλο «Καθυστερώντας τη δέσμευση - Delaying Commitment». Στην εργασία αυτή σημειώνει ότι όταν οι μηχανικοί λογισμικού έρχονται αντιμέτωποι με μια νέα κατάσταση, θα πρέπει να καθυστερήσουν τη λήψη οριστικών αποφάσεων και να διερευνήσουν την κατάσταση, καθώς έχει διαπιστωθεί ότι η τεχνική πολύ συχνά οδηγεί σε νέες προσεγγίσεις και ιδέες. Συνήθως μηχανικοί λογισμικού με μικρή εμπειρία έχουν την τάση να παίρνουν αποφάσεις γρήγορα που αρκετά συχνά είναι λάθος. Ο Thimbleby σημειώνει ότι η πρόωρη δέσμευση στον σχεδιασμό αποτελεί μια κακή πρακτική που περιορίζει τη μάθηση, και τη χρησιμότητα του προϊόντος ενώ αυξάνει το κόστος της αλλαγής.

Μια έννοια που σχετίζεται με την καθυστερημένη λήψη αποφάσεων είναι αυτή της τελευταίας υπεύθυνης στιγμής (Last Responsible Moment - LRM), που ορίζεται ως το χρονικό σημείο όπου το κόστος της μη λήψης μιας απόφασης γίνεται μεγαλύτερο από το κόστος λήψης μιας απόφασης ή το σημείο κατά το οποίο η αποτυχία λήψης μιας απόφασης εξαλείφει μια σημαντική εναλλακτική λύση. Εάν η απόφαση δεν ληφθεί μέχρι την τελευταία υπεύθυνη στιγμή, τότε οι αποφάσεις λαμβάνονται εξ ορισμού, κάτι που γενικά δεν είναι μια καλή προσέγγιση για τη λήψη αποφάσεων.

Δεν θα πρέπει να συγχέουμε την αναβλητικότητα με τη λήψη αποφάσεων την τελευταία υπεύθυνη στιγμή. Στην πραγματικότητα, η καθυστέρηση λήψης αποφάσεων είναι δύσκολη στην εφαρμογή της, διότι συνδέονται με την ψυχολογική πίεση απώλειας εναλλακτικών λύσεων. Ακολουθούν ορισμένες τακτικές για τη λήψη αποφάσεων την τελευταία υπεύθυνη στιγμή (Poppendieck & Poppendieck, 2007):

- Μοιραστείτε τη σχεδίαση του συστήματος ακόμη και ημιτελή. Η ιδέα ότι η σχεδίαση πρέπει να είναι ολοκληρωμένη πριν δοθεί στον πελάτη είναι αντιπαραγωγική, σύμφωνα με τις αρχές της λιτής σχεδίασης. Είναι αντιπαραγωγική καθώς αυξάνει τον απαιτούμενο χρόνο για ανατροφοδότηση από τον πελάτη και προκαλεί τη λήψη μη αναστρέψιμων αποφάσεων πολύ νωρίτερα από ό,τι χρειάζεται. Ο καλός σχεδιασμός είναι μια διαδικασία ανακάλυψης, που γίνεται μέσω σύντομων, επαναλαμβανόμενων εξερευνητικών κύκλων.
- Οργανώστε την άμεση συνεργασία μεταξύ εργαζομένων. Η αποδέσμευση ελλιπούς πληροφορίας αποσκοπεί στη βελτίωση του σχεδιασμού καθώς προχωρά η ανάπτυξη. Αυτό απαιτεί ότι τα άτομα που κατανοούν τις λεπτομέρειες λειτουργίας του συστήματος για να παρέχει αξία πρέπει να επικοινωνούν απευθείας με άτομα που κατανοούν τις λεπτομέρειες της τεχνικής δομής του συστήματος.
- Ο Thimbleby παρατηρεί ότι η διαφορά μεταξύ άπειρων και πεπειραμένων μηχανικών λογισμικού είναι ότι οι έμπειροι μηχανικοί ξέρουν πώς να καθυστερούν τις τεχνικές δεσμεύσεις και πώς να κρύβουν τα λάθη τους για όσο το δυνατόν περισσότερο. Οι έμπειροι μηχανικοί επιδιορθώνουν τα λάθη τους προτού προκαλέσουν προβλήματα. Οι άπειροι προσπαθούν να κάνουν τα πάντα σωστά την πρώτη φορά και έτσι υπερφορτώνουν το σύστημα με χαρακτηριστικά που τους δεσμεύουν σε αποφάσεις. Ο Thimbleby συνιστά ορισμένες τακτικές για την καθυστέρηση της δέσμευσης στην ανάπτυξη λογισμικού, οι οποίες θα μπορούσαν να συνοψιστούν ως υποστήριξη της αντικειμενοστρεφούς σχεδίασης και της ανάπτυξης βασισμένης σε συστατικά (component-based development):
- Χρήση αρθρωμάτων - modules: Η απόκρυψη πληροφορίας (information hiding), ή γενικότερα η απόκρυψη της συμπεριφοράς των αντικειμένων, είναι μια θεμελιώδης αρχή της αντικειμενοστρεφούς ανάπτυξης. Η απόκρυψη πληροφορίας και συμπεριφοράς επιτρέπει την καθυστέρηση δέσμευσης έως ότου να σταθεροποιηθούν οι απαιτήσεις των πελατών.
- Χρήση διεπαφών (interfaces): Είναι πολύ καλή και ιδιαίτερα δημοφιλής τεχνική να διαχωρίζουμε τη διεπαφή από την υλοποίησή της καθώς οι πελάτες δεν πρέπει να εξαρτώνται από τις αποφάσεις υλοποίησης.
- Χρήση παραμέτρων: Μετατρέψτε τις σταθερές μεταβλητές σε παραμέτρους. Χρησιμοποιείστε παραμετρικά, όπου είναι εφικτό βάσεις δεδομένων, ενδιάμεσο λογισμικό (middleware) τρίτων, κ.λπ.. Χρησιμοποιήστε τις δυνατότητες αυτές μέσα σε αρθρώματα (modules) μέσω διεπαφών, μειώνοντας την εξάρτηση του συστήματος από συγκεκριμένες υλοποιήσεις.
- Χρησιμοποιήστε αφαιρέσεις (abstraction): Η αφαίρεση και η δέσμευση είναι αντίστροφες διεργασίες. Αναβάλετε τη δέσμευση σε συγκεκριμένες αναπαραστάσεις, με την προϋπόθεση ότι η αφαίρεση που υιοθετήσατε εξυπηρετεί άμεσες σχεδιαστικές ανάγκες.
- Αποφύγετε τον ακολουθιακό προγραμματισμό (sequential programming): Χρησιμοποιήστε δηλωτικό προγραμματισμό (declarative programming) αντί για διαδικαστικό προγραμματισμό (procedural programming). Έτσι ανταλλάσσετε την αξία της απόδοσης με αυτή της ευελιξίας. Ορίστε τους αλγόριθμούς σας με τέτοιο τρόπο ώστε να μην εξαρτώνται από μια συγκεκριμένη σειρά εκτέλεσης.
- Δώστε προσοχή στη χρήση προσαρμοσμένων εργαλείων: Η χρήση συγκεκριμένων εργαλείων απαιτεί συχνά να δεσμευτούμε από πολύ νωρίς σε λεπτομέρειες υλοποίησης.
- Αποφυγή της επανάληψης: Η αρχή αυτή είναι γνωστή ως η αρχή Don't Repeat Yourself (DRY) ή Once And Only Once (OAOO). Εάν κάθε λειτουργία του λογισμικού υλοποιείται μόνο σε ένα σημείο στον κώδικα, θα υπάρχει μόνο ένα μέρος για αλλαγή όταν η λειτουργία αυτή πρέπει να αλλάξει και δεν θα υπάρχουν ασυνέπειες (Thomas & Hunt, 2019).

- Διαχωρισμός των εννοιών (separate concerns): Κάθε άρθρωμα πρέπει να έχει μια σαφώς καθορισμένη ευθύνη. Αυτό σημαίνει ότι κάθε κλάση θα έχει μόνο έναν λόγο για να αλλάξει (Martin, 2003).
- Ενθυλάκωση παραλλαγής (encapsulate variation): Η ενθυλάκωση αποτελεί βασική αρχή της αντικειμενοστρεφούς προσέγγισης, σύμφωνα με την οποία μία κλάση αποκρύπτει την εσωτερική της δομή και πολυπλοκότητα και παρουσιάζει μια απλοποιημένη εξωτερική όψη, η οποία και είναι διαθέσιμη για την αλληλεπίδραση με το περιβάλλον. Αυτό που είναι πιθανό να αλλάξει πρέπει να βρίσκεται μέσα στην κλάση, ενώ οι διεπαφές πρέπει να είναι σταθερές. Αυτή η στρατηγική, φυσικά, εξαρτάται από τη βαθιά κατανόηση του πεδίου προβλήματος (problem domain) ώστε να γνωρίζουμε ποιες πτυχές θα είναι σταθερές και ποιες μεταβλητές (Γερογιάννης et al., 2006).
- Αναβολή υλοποίησης μελλοντικών δυνατοτήτων: Θα πρέπει να υλοποιήσουμε μόνο τον απλούστερο κώδικα που ικανοποιεί τις άμεσες ανάγκες αντί να υλοποιήσουμε δυνατότητες που πιθανόν να χρειαστούμε στο μέλλον. Αφενός, στο μέλλον θα γνωρίζουμε καλύτερα τι χρειαζόμαστε και αφετέρου ο απλός κώδικας είναι ευκολότερο να επεκταθεί, εάν χρειαστεί. Ο Beck περιγράφει αυτή την αρχή με το ακρώνυμο YAGNI (You Aren't Going to Need It).

5.6 Η ταχεία παράδοση

Η ταχεία παράδοση είναι μια επιχειρησιακή πρακτική που παρέχει στην επιχείρηση ισχυρό ανταγωνιστικό πλεονέκτημα. Η ταχεία παράδοση σημαίνει ότι η επιχείρηση:

- Παραδίδει γρηγορότερα στους πελάτες.
- Δεσμεύει λιγότερους πόρους κατά τη διάρκεια των εργασιών.
- Δημιουργεί μικρότερα αποθέματα.
- Μειώνει τον κίνδυνο του έργου.

Η αρχή της ταχείας παράδοσης είναι συμπληρωματική της αρχής της καθυστερημένης λήψης αποφάσεων αφού όσο πιο γρήγορα μπορούμε να παραδώσουμε, τόσο περισσότερο μπορούμε να καθυστερήσουμε τις αποφάσεις.

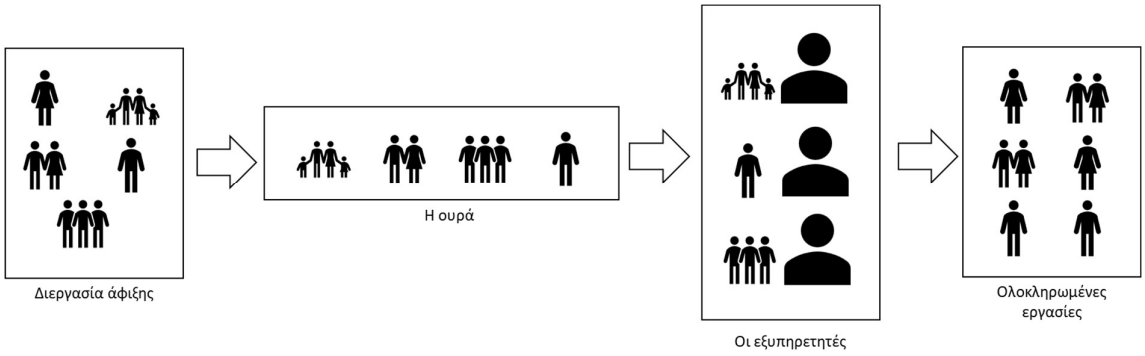
Η γρήγορη παράδοση δεν είναι κάτι που επιτυγχάνεται εύκολα αλλά απαιτεί την εφαρμογή ουρών αναμονής (queueing theory), τη χρήση μοντέλων για τον υπολογισμό του κόστους καθυστέρησης (cost of delay) καθώς και την υλοποίηση συστημάτων έλξης (pull systems), όπως το σύστημα καρτών Kanban.

5.6.1 Ουρές αναμονής

Ο σκοπός της θεωρίας των ουρών αναμονής είναι να διασφαλίσει ότι οι εργαζόμενοι δεν είναι υπερφορτωμένοι με εργασίες, ώστε να έχουν χρόνο να κάνουν τα πράγματα με τον σωστό τρόπο και αφετέρου ο χρόνος αναμονής των πελατών ελαχιστοποιείται έχοντας ως συνέπεια την αύξηση της ικανοποίησης. Η ουρά είναι μια λίστα εργασιών, δυνατοτήτων ή στοιχείων υποχρεώσεων για μια ομάδα ή για έναν μεμονωμένο εργαζόμενο. Στη θεωρία ουρών, οι βασικές έννοιες που συνήθως βρίσκουμε είναι οι ακόλουθες (βλέπε Εικόνα 5.8):

- Η **ουρά** (queue) περιέχει τις εργασίες που περιμένουν να εκτελεστούν (π.χ. πελάτες προς εξυπηρέτηση, ιστορίες χρηστών προς ανάπτυξη, σφάλματα προς διόρθωση).
- Ο **διακομιστής** (server) είναι ο πόρος (π.χ. εργαζόμενος, μηχανή) που εκτελεί την εργασία που βρίσκεται στην αρχή της ουράς.
- Το μοτίβο με το οποίο φτάνουν οι νέες εργασίες ονομάζεται **διεργασία άφιξης** (arrival process). Η διεργασία άφιξης είναι συνήθως απρόβλεπτη, δηλαδή δεν υπάρχει μοτίβο ή προβλεψιμότητα στον τρόπο με τον οποίο φτάνουν οι εργασίες.

- Ο χρόνος που χρειάζεται ο διακομιστής για να ολοκληρώσει την εργασία ονομάζεται **χρόνος εξυπηρέτησης** (process time) και είναι τυχαία μεταβλητή.
- Η σειρά εκτέλεσης των εργασιών συνήθως καθορίζεται από τον σχετικό αλγόριθμο που ορίζει την **πολιτική δρομολόγησης εργασιών της ουράς** (queue discipline), όπως για παράδειγμα First In - First Out (FIFO), Last In First Out (LIFO), First In Still Here, Round Robin, κ.λπ. (Bhat, 2008).



Εικόνα 5.8: *Μια ουρά αναμονής*

Το βασικό χαρακτηριστικό μιας ουράς είναι ο χρόνος κύκλου (Cycle Time – CT)- δηλαδή ο μέσος χρόνος που χρειάζεται η κατασκευή ενός προϊόντος ή η υλοποίηση μιας ιστορίας χρήστη για να φτάσει από το ένα άκρο της διεργασίας στο άλλο. Γενικότερα, σύμφωνα με το νόμο του Little, σε ένα σταθερό σύστημα, ο μέσος αριθμός πελατών που βρίσκονται στην ουρά είναι ίσος με τον μέσο ρυθμό που φτάνουν οι πελάτες στο σύστημα πολλαπλασιαζόμενο με τον μέσο χρόνο που ο πελάτης μένει μέσα στο σύστημα.

$$L = \lambda W$$

Όπου:

L είναι ο μέσος αριθμός πελατών στην ουρά ή το μέγεθος της ουράς. Είναι ισοδύναμο με το Work In Progress (WIP).

λ είναι ο μέσος αριθμός άφιξης πελατών ανά μονάδα χρόνου. Είναι ισοδύναμο με την ταχύτητα παραγωγής (throughput) και υπολογίζεται προσδιορίζοντας πόσα είδη παράγονται, διαιρώντας τα με τον χρόνο που χρειάστηκε για την παραγωγή τους.

W είναι ο μέσος χρόνος αναμονής για ένα πελάτη στο σύστημα. Είναι ισοδύναμο με τον χρόνο κύκλου (Cycle Time – CT), δηλαδή τον χρόνο που απαιτείται για την εξυπηρέτηση ενός πελάτη ή τον χρόνο παραγωγής ενός προϊόντος.

Παράδειγμα

Έστω ένας μηχανικός λογισμικού που εκτελεί script ελέγχου. Κάθε ώρα υπάρχουν κατά μέσο όρο 10 script ελέγχου. Απαιτείται χρόνος 30 λεπτών για την επεξεργασία του καθενός από αυτά. Αυτό σημαίνει ότι κάθε δεδομένη στιγμή υπάρχουν 5 script ελέγχου σε αναμονή.

λ= 10 script ελέγχου ανά ώρα

W = 0,5 ώρες

$L = \lambda * W = 10 * \ 0{,}5 = 5 \ script \ ελέγχου$

Σε ένα παραγωγικό σύστημα, το ρολόι του κύκλου ξεκινά όταν κάτι μπαίνει σε μια ουρά και συνεχίζει να χτυπά ενώ περιμένει στην ουρά, για κάποια ενδιάμεση επεξεργασία και μέχρι να ολοκληρωθούν όλα τα βήματα της επεξεργασίας.

Υπάρχουν δύο τρόποι για να μειωθεί ο χρόνος κύκλου.

- Ο πρώτος είναι να δούμε τον τρόπο με τον οποίο έρχονται τα αιτήματα για νέα λειτουργικότητα ή για διόρθωση σφαλμάτων, και
- ο δεύτερος τρόπος είναι να δούμε τον τρόπο επεξεργασίας των αιτημάτων.

Σε ορισμένα συστήματα, δεν είναι δυνατό να επηρεαστεί ο ρυθμός άφιξης των αιτημάτων καθώς εξαρτάται αποκλειστικά από το εξωτερικό περιβάλλον. Όμως μπορούν να θεσπιστούν τιμολογιακές πολιτικές για την εξομάλυνση της εισερχόμενης ζήτησης ώστε να έχουμε σταθερή ζήτηση (steady ready of arrival).

Για παράδειγμα, μια τηλεφωνική εταιρεία που προσφέρει πολύ χαμηλές χρεώσεις για τη νύχτα και το Σαββατοκύριακο το κάνει αυτό για να εξομαλύνει τη ζήτηση αιχμής. Αντίστοιχα, οι αεροπορικές εταιρείες χρησιμοποιούν μεταβλητές τιμές για την εξομάλυνση των κρατήσεων στις πτήσεις. Το αποτέλεσμα είναι ότι αν τα εισερχόμενα αιτήματα κατανεμηθούν με τέτοιο τρόπο ώστε να ταιριάζει με τη δυναμικότητα του συστήματος, οι ουρές, και επομένως οι χρόνοι κύκλου, συντομεύονται.

Ένας εύκολος τρόπος για να ελέγξουμε τον ρυθμό άφιξης των εργασιών είναι να απελευθερώνουμε συχνά νέες εκδόσεις (new software releases). Αντίθετα, αν οι απελευθερώσεις είναι μεγάλες σε μέγεθος, τότε η ουρά είναι μεγάλη και αντίστοιχα ο χρόνος κύκλου. Στην περίπτωση των ευέλικτων μεθόδων ανάπτυξης λογισμικού μπορούμε να πούμε ότι η ουρά αναμονής είναι ο κατάλογος απαιτήσεων (product backlog). Ένας ακόμη τρόπος για να ελέγξουμε το μέγεθος της ουράς εργασιών είναι η ιεράρχηση αυτών δίνοντας προτεραιότητα σε κάθε εργασία.

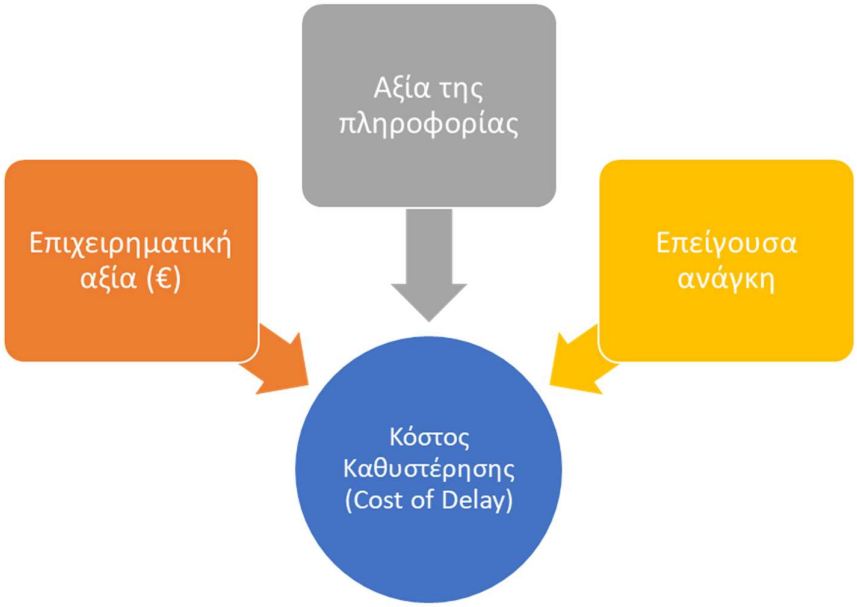
5.6.2 Κόστος καθυστέρησης

Το κόστος καθυστέρησης (Cost of Delay - CoD) είναι ένα πλαίσιο ιεράρχησης που βοηθά μια επιχείρηση να ποσοτικοποιήσει την οικονομική αξία της ολοκλήρωσης ενός έργου νωρίτερα σε σχέση με τη συμβατική ημερομηνία λήξης. Το κόστος καθυστέρησης μπορεί να χρησιμοποιηθεί σε διάφορες περιπτώσεις όπως:

- Στην ανάπτυξη προϊόντων – και αφορά το χρηματικό ποσό που θα χαθεί εάν καθυστερήσει η κυκλοφορία του νέου προϊόντος.
- Στην ανάπτυξη λογισμικού – και αφορά το χρηματικό ποσό που θα χαθεί εάν καθυστερήσει η ανάπτυξη μιας καινοτόμου λειτουργίας/χαρακτηριστικού που θα δώσει ανταγωνιστικό πλεονέκτημα.
- Στη λειτουργία μιας ψηφιακής υπηρεσίας– και αφορά το χρηματικό ποσό που θα χαθεί από τη μη παροχή της υπηρεσίας.

Το κόστος καθυστέρησης από την υλοποίηση ενός χαρακτηριστικού είναι η αξία που θα μπορούσε να δημιουργηθεί σε μια δεδομένη χρονική περίοδο, εάν το χαρακτηριστικό αυτό του προϊόντος ήταν άμεσα διαθέσιμο. Περιλαμβάνει (Arnold & Yüce, 2013):

- την επιχειρηματική αξία του χαρακτηριστικού,
- την αξία της πληροφορίας που ανακαλύπτουμε κατά τη διαδικασία ανάπτυξης και
- την αξιολόγηση του τρόπου με τον οποίο η επιχειρηματική αξία και η αξία της πληροφορίας γίνονται λιγότερο χρήσιμες-πολύτιμες με την πάροδο του χρόνου, ή με απλά λόγια αν το χαρακτηριστικό αποτελεί επείγουσα ανάγκη (βλέπε Εικόνα 5.9).



Εικόνα 5.9: *Οι παράγοντες που επηρεάζουν το κόστος καθυστέρησης*

Ο υπολογισμός του κόστους καθυστέρησης γίνεται με την ανάπτυξη μοντέλων κερδών-ζημιών (Profit and Losses model – P&L) όπου υπολογίζουμε το P&L για τουλάχιστον δύο σενάρια (Poppendieck & Poppendieck, 2003):

- το σενάριο χωρίς την εισαγωγή του νέου καινοτόμου χαρακτηριστικού του λογισμικού (cost of delay) και
- το σενάριο που περιλαμβάνει την εισαγωγή του νέου καινοτόμου χαρακτηριστικού του λογισμικού. Η διαφορά που προκύπτει μεταξύ των δύο σεναρίων είναι το κόστος καθυστέρησης (cost of delay)

Για να υπολογίσουμε με απλό τρόπο κόστος της καθυστέρησης, πρέπει να προβλέψουμε το πιθανό κέρδος ανά μονάδα χρόνου (π.χ. εβδομάδα). Για παράδειγμα, έστω ότι έχουμε μια εταιρεία SaaS (Software as a Service) που σκοπεύει να θέσει σε λειτουργία μια νέα πρόσθετη λειτουργία που αναμένεται ότι θα αποφέρει σε πωλήσεις επιπλέον 20.000 € ανά εβδομάδα. Στην περίπτωση αυτή, κάθε εβδομάδα που αναβάλλουμε την κυκλοφορία της νέας λειτουργίας κοστίζει στην επιχείρηση αυτό το ποσό. Για να δώσουμε μια απλή εξήγηση του τρόπου υπολογισμού του κόστους της καθυστέρησης, φανταστείτε ότι έχουμε προγραμματίσει τρία έργα ανάπτυξης λογισμικού για το εγγύς μέλλον. Το αποτέλεσμα των έργων θα δώσει αξία στους πελάτες της επιχείρησης και κέρδος στην επιχείρηση ανάπτυξης λογισμικού. Το ζητούμενο είναι να υπολογίσουμε τη σωστή σειρά παράδοσης των έργων, αφού η παράδοση των έργων με τη σειρά Β, Α και Γ, έχει διαφορετική κερδοφορία από την παράδοση των έργων με τη σειρά Α, Β και Γ.

Για να επιλέξουμε το πλέον κερδοφόρο σχέδιο δράσης, θα πρέπει να υπολογίσουμε την τιμή CD3 (Cost of Delay Divided by Duration) για κάθε ένα από τα έργα. Με απλά λόγια - το κόστος της καθυστέρησης διαιρεμένο με τη διάρκεια του έργου. Έτσι για τον υπολογισμό έχουμε (Goljan, 2021; Kanbanize.com):

- Υπολογίζουμε το εβδομαδιαίο κέρδος/αξία του αναμενόμενου έργου.
- Εκτιμούμε τον απαιτούμενο χρόνο για την υλοποίηση του έργου.
- Διαιρούμε το κέρδος με την εκτιμώμενη διάρκεια του έργου.

	Διάρκεια έργου (εβδομάδες)	Εκτιμώμενο κέρδος	CD3
Project 1	2	5.000	5.000/2=**2.500**
Project 2	4	30.000	30.000/4=**7.500**
Project 3	8	50.000	50.000/8=**6.250**
Σύνολο	14	85.000	N/A

Πίνακας 5.3: *Υπολογισμός CD3*

Τα έργα με υψηλότερο CD3 είναι πιο σημαντικά από οικονομικής άποψης για την επιχείρηση, καθώς η απόδοση της επένδυσης αυτών θα είναι ταχύτερη. Όπως ήδη αναφέρθηκε η σειρά εκτέλεσης των έργων επηρεάζει την κερδοφορία της επιχείρησης. Στην Εικόνα 5.10 παρουσιάζονται κάποια πιθανά εναλλακτικά σενάρια. Κάθε σενάριο που παρουσιάζεται στην Εικόνα 5.10 έχει διαφορετική κερδοφορία για την επιχείρηση. Συνεπώς θα πρέπει να υπολογίσουμε το κάθε σενάριο ξεχωριστά.

Σενάριο 1 – Με βάση τη διάρκεια του έργου

Εάν δώσουμε προτεραιότητα βάσει της διάρκειας του έργου, τότε το CoD για τις δύο πρώτες εβδομάδες είναι 10.000 €. Μετά την ολοκλήρωση του συντομότερου έργου, δηλαδή του πρώτου έργου, για τις επόμενες τέσσερις εβδομάδες εκτελούμε το δεύτερο έργο. Λαμβάνοντας υπόψη ότι το δεύτερο έργο διαρκεί τέσσερις εβδομάδες και θα παραδοθεί την έκτη εβδομάδα, το συνολικό κόστος της καθυστέρησης γίνεται 180.000 €. Όταν ολοκληρωθεί και το δεύτερο έργο, για τις οκτώ εβδομάδες που απαιτούνται για να τελειώσει το τρίτο έργο το κόστος καθυστέρησης είναι 700.000€. Συνεπώς, το πρώτο σενάριο έχει συνολικό κόστος καθυστέρησης 890.000€ (βλέπε Πίνακα 5.4).

Εβδομάδες	Όλα τα έργα ταυτόχρονα	Το συντομότερο πρώτα	Αυτό με τη μεγαλύτερη αξία	Υψηλότερο CD3
1	X	Project 1	Project 3	Project 2
2	X	Project 1	Project 3	Project 2
3	X	Project 2	Project 3	Project 2
4	X	Project 2	Project 3	Project 2
5	X	Project 2	Project 3	Project 3
6	X	Project 2	Project 3	Project 3
7	X	Project 3	Project 3	Project 3
8	X	Project 3	Project 3	Project 3
9	X	Project 3	Project 2	Project 3
10	X	Project 3	Project 2	Project 3
11	X	Project 3	Project 2	Project 3
12	X	Project 3	Project 2	Project 3
13	X	Project 3	Project 1	Project 1
14	X	Project 3	Project 1	Project 1

Εικόνα 5.10: *Εναλλακτικά σενάρια χρονοδρομολόγησης έργων*

	Εβδομάδες	Εβδομαδιαίο κέρδος	Κόστος καθυστέρησης (CoD)
Project 1	2	5.000	2x5.000=**10.000**
Project 2	4	30.000	(4+2)x30.000=**180.000**
Project 3	8	50.000	(8+4+2)x50.000=**700.000**
		Σύνολο	**890.000**

Πίνακας 5.4: *Υπολογισμός CoD για ιεράρχηση έργων με βάση τη διάρκεια*

Σενάριο 2 – Με βάση την αξία του έργου

Εάν δώσουμε προτεραιότητα με βάση την αξία του έργου, το κόστος καθυστέρησης για τις πρώτες οκτώ εβδομάδες είναι 400.000 €. Για τις επόμενες τέσσερις εβδομάδες είναι 360.000 €, ενώ για τις τελευταίες δύο εβδομάδες είναι 70.000€. Το σύνολο ανέρχεται σε 830.000 € (βλέπε Πίνακα 5.5).

	Εβδομάδες	Εβδομαδιαίο Κέρδος	Κόστος καθυστέρησης (CoD)
Project 3	8	50.000	8x50.000=**400.000**
Project 2	4	30.000	(4+8)x30.000=**360.000**
Project 1	2	5.000	(4+8+2)x5.000=**70.000**
		Σύνολο	**830.000**

Πίνακας 5.5: *Υπολογισμός CoD για ιεράρχηση έργων με βάση την αξία*

Με αντίστοιχο τρόπο υπολογίζουμε και τα άλλα δύο σενάρια που είναι αυτά με τη χρήση του κριτηρίου CD3 και της ταυτόχρονης εκτέλεσης των έργων. Ο Πίνακας 5.6 παρουσιάζει συγκριτικά και τα τέσσερα σενάρια όπου προφανώς το μικρότερο κόστος εμφανίζεται όταν η ιεράρχηση γίνει με τη χρήση του κριτηρίου CD3.

Κριτήριο	**Κόστος καθυστέρησης (CoD)**
Παράλληλη εκτέλεση	1.190.000
Με βάση τη διάρκεια	890.000
Με βάση τη μεγαλύτερη αξία	830.000
Με βάση το CD3	790.000

Πίνακας 5.6: *Συγκριτικός πίνακας υπολογισμού CoD*

Για να κατανοήσουμε τον αντίκτυπο που έχει μια καθυστέρηση στην παράδοση ενός έργου και για να μειώσουμε το κόστος της καθυστέρησης, πρέπει να επισημάνουμε ότι υπάρχουν διαφορετικοί τύποι CoD που εμφανίζονται σε διαφορετικές καταστάσεις:

- Το κόστος καθυστέρησης που αυξάνεται γραμμικά με τον χρόνο (standard CoD).
- Το κόστος καθυστέρησης που συνδέεται με ένα ορόσημο (fixed date CoD). Σε αυτό τον τύπο, το κόστος καθυστέρησης είναι πολύ χαμηλό ή ανύπαρκτο μέχρι μια συγκεκριμένη ημερομηνία - ορόσημο. Μετά το πέρας αυτής της ημερομηνίας το κόστος αυξάνεται εκθετικά. Για παράδειγμα, εάν η ομάδα έργου αποτύχει να ανταποκριθεί σε ένα κρίσιμο πρόβλημα εντός ενός συγκεκριμένου χρονικού πλαισίου, ο πάροχος υπηρεσιών θα πρέπει να αποζημιώσει τον πελάτη.
- Το κόστος καθυστέρησης που υπαγορεύεται από τον επείγοντα χαρακτήρα του έργου. Στην περίπτωση αυτή το CoD αυξάνεται γρήγορα σε πολύ σύντομο χρονικό διάστημα και στη συνέχεια παραμένει σχετικά σταθερό.

5.7 Η μέθοδος Kanban

Η λέξη kanban είναι μια ιαπωνική λέξη και σημαίνει "ένας πίνακας με κάρτες". Η λογική της οπτικοποίησης της εργασίας με κάρτες σε ένα πίνακα χρησιμοποιήθηκε από την εταιρεία Toyota από τη δεκαετία του 1950 ως μια μέθοδο χρονοδρομολόγησης εργασιών. Στο σύστημα παραγωγής Toyota (Toyota Production System), ή στην προσέγγιση Kanban, (Ohno 1988), στόχοι είναι η ελαχιστοποίηση του αποθέματος στο εργοστάσιο και η διασφάλιση ότι το απαιτούμενο απόθεμα παραδίδεται στο σημείο όπου θα καταναλωθεί/χρησιμοποιηθεί, ακριβώς τη στιγμή που χρειάζεται (βλέπε Εικόνα 5.11). Ο Ohno βασίστηκε σε μεθόδους που χρησιμοποιούσαν οι αμερικανικές αλυσίδες παντοπωλείων, τις οποίες είχε παρατηρήσει να στοιβάζουν προϊόντα τα ράφια, ανάλογα με τις ανάγκες, ακριβώς τη στιγμή που οι καταναλωτές ήταν έτοιμοι να αγοράσουν τα προϊόντα. Γενικά, η προσέγγιση του Ohno είναι γνωστή ως παραγωγή Just-In-Time (JIT). Με το πέρασμα του χρόνου η μέθοδος μεταφέρθηκε, από τον κλάδο της παραγωγής αυτοκινήτων και σε άλλους κλάδους, όπως τον κλάδο της ανάπτυξης λογισμικού. Έτσι σήμερα, η λέξη Kanban (με κεφαλαίο Κ) αναφέρεται στη μέθοδο Kanban όπως αυτή έχει περιγραφεί από τον Anderson το 2003.

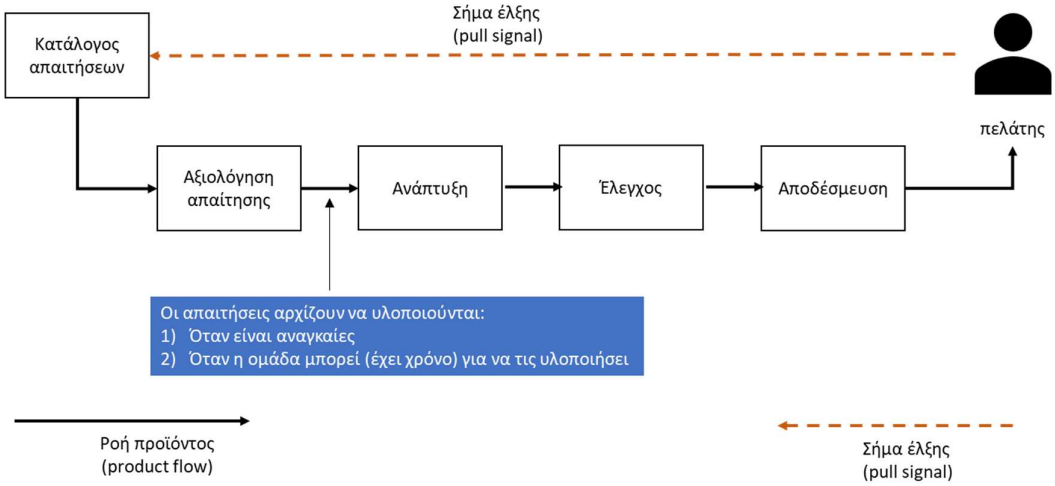
Εικόνα 5.11: *Το σύστημα kanban της εταιρείας Toyota. Πηγή: https://www.toyota-global.com*

5.7.1 Τα συστήματα έλξης (pull systems)

Συνήθως, η παραγωγή προϊόντων βασίζεται στη λογική ότι θα παραχθεί η μέγιστη δυνατή ποσότητα προϊόντων με βάση την παραγωγική δυνατότητα της επιχείρησης με σκοπό να επιτύχουμε οικονομίες κλίμακας και τη μέγιστη δυνατή παραγωγικότητα. Η λογική αυτή στη θεωρία των συστημάτων παραγωγής περιγράφεται ως συστήματα ώθησης (push systems). Η επιχείρηση που υιοθετεί αυτό το σύστημα παραγωγής δεν γνωρίζει τις πραγματικές ανάγκες των πελατών, αλλά με βάση ένα μοντέλο πρόβλεψης της ζήτησης, παράγει προϊόντα και αναμένει ότι θα ζητηθούν όσα τελικά προϊόντα παράγονται. Κάθε ημιέτοιμο προϊόν τροφοδοτεί έναν επόμενο σταθμό επεξεργασίας και καταλήγει να μετουσιωθεί σε τελικό προϊόν. Η προσέγγιση αυτή εμπεριέχει σημαντικό ρίσκο, καθώς ενδέχεται να μην πωληθούν όλα τα παραχθέντα προϊόντα ή να απορριφθούν αποθηκευμένες ποσότητες έτοιμων προϊόντων, τα οποία κατέστησαν παρωχημένα.

Αντίθετα, ένα σύστημα έλξης (pull system) λειτουργεί αντίστροφα, καθώς η παραγωγή ξεκινά κατόπιν πραγματικής ζήτησης, δηλαδή κατόπιν της παραγγελίας των πελατών (βλέπε Εικόνα 5.12). Ως εκ τούτου εκμηδενίζεται η σπατάλη πόρων για υπερπαραγωγή ή αποθήκευση πλεοναζόντων προϊόντων. Αντίθετα, το σύστημα έλξης μειονεκτεί στην ενδεχόμενη περίπτωση όπου ο οργανισμός δεν έχει διαθέσιμες πρώτες ύλες και πόρους για την παραγωγή ενός προϊόντος που θα ζητήσει ένας πελάτης σε οποιαδήποτε στιγμή. Επίσης, σε αντίθεση με τα συστήματα ώθησης, στα οποία τα μέλη της ομάδας εργάζονται σε διαφορετικές εργασίες, σ' ένα σύστημα έλξης τα μέλη της ομάδας εστιάζονται σε ένα μόνο αντικείμενο εργασίας. Αυτή η προσέγγιση επιτρέπει στην επιχείρηση:

- να προσαρμόζεται γρήγορα στις αλλαγές που μπορεί να προκύψουν στη ζήτηση,
- να σχεδιάζει τη βέλτιστη δυναμικότητα της ομάδας,
- να κατασκευάζει και να παραδίδει τα προϊόντα ή τα αντικείμενα εργασίας πολύ πιο γρήγορα,
- να μειώνει τη σπατάλη πόρων,
- να αυξήσει την παραγωγικότητα,
- να βελτιστοποιεί τη ροή εργασίας,
- κ.λπ.



Εικόνα 5.12: *Ένα σύστημα έλξης*

Μια σημαντική διαφορά με ένα σύστημα ώθησης είναι ότι, οι απαιτήσεις των πελατών αρχίζουν να υλοποιούνται μόνο εφόσον πληρούνται οι δύο παρακάτω συνθήκες:

1) όταν είναι αναγκαία μετά από αίτημα του πελάτη
2) όταν η ομάδα ανάπτυξης μπορεί (έχει χρόνο) να υλοποιήσει το λογισμικό.

5.7.2 Οι θεμελιώδεις αξίες της μεθόδου Kanban

Ο Anderson (2003) έχει παρουσιάσει τη μέθοδο Kanban ως μια προσέγγιση για τη σταδιακή, εξελικτική αλλαγή συστημάτων σε οργανισμούς έντασης γνώσης. Οι βασικές αρχές της μεθόδου μπορούν να ενταχθούν σε δύο κατηγορίες. Αυτές είναι (βλέπε Πίνακα 5.7):

- αρχές που αφορούν τη διαχείριση της αλλαγής (change management principles) και
- αρχές που αφορούν την παροχή υπηρεσιών (service delivery principles)

Αρχές διαχείρισης αλλαγής	Αρχές παροχής υπηρεσιών
Ξεκινήστε με αυτό που κάνετε τώρα	Εστίαση στις ανάγκες και τις προσδοκίες του πελάτη
Επιδιώξτε τη σταδιακή, εξελικτική αλλαγή	Διαχειριστείτε την εργασία, όχι τους εργαζομένους
Ενθαρρύνετε τις πράξεις ηγεσίας σε όλα τα επίπεδα	Ελέγχετε τακτικά το δίκτυο των υπηρεσιών

Πίνακας 5.7: *Αρχές μεθόδου Kanban*

- **Ξεκινήστε με αυτό που κάνετε τώρα.** Η μέθοδος Kanban δεν ζητάει από μια ομάδα να αλλάξει τη διαδικασία της ριζικά, αλλά βασίζεται στην ιδέα ότι μπορεί να εξελίξει την τρέχουσα διαδικασία. Δεν υπάρχει η έννοια της σαρωτικής, μηχανικής αλλαγής σε μια νέα διαδικασία ή στυλ εργασίας. Η μέθοδος Kanban υιοθετεί ευελιξία χρήσης χρησιμοποιώντας τις υπάρχουσες ροές εργασίας, και τα υπάρχοντα συστήματα και διεργασίες χωρίς να διαταράσσεται ότι ήδη υπάρχει και λειτουργεί αποτελεσματικά.
- **Επιδιώξτε τη σταδιακή, εξελικτική αλλαγή.** Η μέθοδος Kanban έχει σχεδιαστεί με τέτοιο τρόπο ώστε να συναντά την ελάχιστη αντίσταση. Ενθαρρύνει τις συνεχείς, μικρές, σταδιακές και εξελικτικές αλλαγές στις τρέχουσες διεργασίες με τη χρήση ερωτηματολογίων, αναφορών ανατροφοδότησης (feedback forms), κ.λπ. Γενικά, οι σαρωτικές αλλαγές δεν συνιστώνται διότι συνήθως δημιουργούν αντίδραση στους εργαζόμενους λόγω φόβου ή αβεβαιότητας.
- **Ενθαρρύνετε τις πράξεις ηγεσίας σε όλα τα επίπεδα.** Η ηγεσία ως τρόπος σκέψης και ενέργειας, σε όλα τα επίπεδα, απορρέει από τις καθημερινές γνώσεις των ατόμων που ενεργούν για τη βελτίωση του τρόπου εργασίας τους. Ακόμη και πολύ μικρές αλλαγές μπορούν να καλλιεργήσουν κουλτούρα και νοοτροπία συνεχούς βελτίωσης (Kaizen), με σκοπό την επίτευξη βέλτιστης απόδοσης σε επίπεδο ομάδας/τμήματος/επιχείρησης.
- **Εστίαση στις ανάγκες και τις προσδοκίες του πελάτη.** Η δημιουργία αξίας για τον πελάτη πρέπει πάντα να βρίσκεται στο επίκεντρο κάθε επιχείρησης. Η κατανόηση των αναγκών και των προσδοκιών των πελατών μας δίνει την δυνατότητα να δώσουμε έμφαση στην ποιότητα των παρεχόμενων υπηρεσιών και στην αξία που αυτές δημιουργούν.
- **Διαχειριστείτε την εργασία, όχι τους εργαζομένους.** Η διαχείριση των εργασιών στο πλαίσιο του δικτύου των υπηρεσιών, διασφαλίζει την ενδυνάμωση των εργαζομένων και την αυτοοργάνωση αυτών γύρω από την εργασία. Έτσι έχουμε τη δυνατότητα να εστιάσουμε στα επιθυμητά αποτελέσματα χωρίς τον «θόρυβο» που δημιουργείται από τη μικροδιαχείριση των εργαζομένων που παρέχουν τις υπηρεσίες.
- **Ελέγχετε τακτικά το δίκτυο των υπηρεσιών.** Η ανάπτυξης της υπηρεσιο-κεντρικής προσέγγισης σε μια επιχείρηση απαιτεί τη συνεχή αξιολόγηση για την ανάπτυξη της κουλτούρας εξυπηρέτησης πελατών. Μέσω των τακτικών αναθεωρήσεων σε συνεργάτες της επιχείρησης ή σε εργαζομένους του δικτύου παροχής υπηρεσιών και της αξιολόγησης των εφαρμοζόμενων πολιτικών εργασίας, η μέθοδος Kanban ενθαρρύνει τη βελτίωση των προσφερόμενων υπηρεσιών.

Εκτός από τις αρχές της μεθόδου Kanban, ο Anderson (2003) περιγράφει κάποια ενδιαφέροντα σημεία που είναι σημαντικά για την πραγματική κατανόηση της μεθόδου Kanban.

- Ο κύριος λόγος για να υιοθετήσει κανείς τη μέθοδο Kanban, είναι η βελτίωση των υφιστάμενων διεργασιών παράδοσης λογισμικού. Στόχος βέβαια δεν είναι να αναμορφωθεί ολοκληρωτικά η διεργασία ανάπτυξης, αλλά με βάση το υφιστάμενο περιβάλλον, το οποίο ήδη λειτουργεί (αν και όχι τέλεια) να βελτιωθεί όπου είναι εφικτό. Σε γενικές γραμμές γίνεται η παραδοχή πως είναι

καλύτερα να βελτιστοποιηθεί μια υπάρχουσα διαδικασία επειδή είναι ευκολότερο, ταχύτερο, και θα προκαλέσει λιγότερη αντίσταση παρά να εισάγουμε μια «ριζοσπαστική» αλλαγή.

- Οι διεργασίες πρέπει να προσαρμοστούν για κάθε συγκεκριμένη περίπτωση/επιχείρηση, επειδή οι άνθρωποι είναι μοναδικοί, οι ομάδες είναι μοναδικές, οι οργανώσεις είναι μοναδικές, και η κατάσταση στην οποία βρίσκεται μια ομάδα είναι μοναδική.
- Οι συχνές αποδεσμεύσεις του λογισμικού παρέχουν την απαραίτητη ορατότητα για την πρόοδο της ομάδας και έτσι αναπτύσσεται εμπιστοσύνη ανάμεσα στα ενδιαφερόμενα μέρη.
- Στη μέθοδο Kanban η ομάδα ανάπτυξης πρέπει να συνεργάζεται με όλους τους συμμετέχοντες, με στόχο την ενεργή συμμετοχή όλων των ενδιαφερόμενων φορέων, και όχι μόνο με έναν αντιπρόσωπο των ενδιαφερόμενων φορέων, όπως για παράδειγμα έναν αναλυτή διεργασιών της επιχείρησης.
- Οι ομάδες Kanban στοχεύουν σε μια μακροπρόθεσμη σχέση παροχής υπηρεσιών με την επιχείρηση. Μια ομάδα Kanban δεν δημιουργείται για ένα συγκεκριμένο έργο και στη συνέχεια διαλύεται, αλλά αντίθετα ο σκοπός είναι να δημιουργηθεί μια μακροχρόνια σχέση.
- Το αληθινό μέτρο της επιτυχίας είναι η επιχειρηματική αξία του προϊόντος που παραδίδεται και όχι απλώς η λειτουργικότητα του λογισμικού.

Γενικότερα, ένα σημαντικό πλεονέκτημα της μεθόδου Kanban είναι ότι παρέχει απλούς μηχανισμούς για τις ομάδες, ξεκαθαρίζοντας τα θέματα που επηρεάζουν την απόδοσή τους και τις παρακινεί να επικεντρωθούν στην αντιμετώπιση του πραγματικού προβλήματος, και όχι στη διεργασία. Επίσης, παρουσιάζει στην ομάδα τα στοιχεία εργασίας που έχουν πρόβλημα, με σαφή τρόπο και επειδή τα στοιχεία αυτά δεν είναι εύκολο να αγνοηθούν, τα μέλη της ομάδας που έχουν ελεύθερο χρόνο παρακινούνται να αντιμετωπίσουν ένα πρόβλημα γρήγορα για να αφαιρεθεί η κάρτα από τον πίνακα. Επιπλέον, η μέθοδος Kanban παρέχει διαφάνεια τόσο για την εργασία και τη διεργασία, δείχνοντας πώς μια εργασία περνά από μία ομάδα σε μια άλλη ή από ένα εργαζόμενο σε έναν άλλο, βελτιώνοντας έτσι την κατανόηση και την ομαδική εργασία. Αυτό βοηθά τις ομάδες να οργανωθούν πιο αποτελεσματικά και παρέχει στα ανώτερα στελέχη την απαραίτητη πληροφορία που τους επιτρέπει να διαχειριστούν πιο αποτελεσματικά το έργο. Τέλος, ο συνδυασμός της βελτιωμένης ροής και της βελτιωμένης ποιότητας έχει ως αποτέλεσμα τη μείωση του χρόνου αντίδρασης της ομάδας. Αυτό με τη σειρά του οδηγεί στη βελτίωση της απόδοσης και της προβλεψιμότητας. Η βελτιωμένη απόδοση και προβλεψιμότητα δίνουν τη δυνατότητα στους διαχειριστές να αντιμετωπίσουν την ανάπτυξη λογισμικού σαν μια επιχείρηση με στόχους και όχι σαν ένα αδιαφανές τέλμα από ανεκπλήρωτες υποσχέσεις.

Παρότι η Kanban είναι και αυτή μια ευέλικτη μέθοδος, υπάρχουν αρκετές διαφορές ανάμεσα σε αυτήν και τις υπόλοιπες μεθόδους (Γερογιάννης, 2013; Rehkopf, n.d.).

α) Ομοιότητες

- Είναι λιτές (lean) και ευέλικτες (agile)
- Και οι δύο χρησιμοποιούν χρονοπρογραμματισμό με βάση την έλξη (pull system)
- Περιορίζουν τις εργασίες σε εξέλιξη (WIP limit)
- Βασίζονται στη διαφάνεια, με στόχο τη βελτίωση των διαδικασιών
- Εστιάζουν στη γρήγορη και συχνή παράδοση λογισμικού προς απελευθέρωση
- Λειτουργούν με βάση ομάδες που αυτό-οργανώνονται
- Προϋποθέτουν τον καταμερισμό εργασιών
- Επιτρέπουν την εργασία σε πολλαπλά προϊόντα ταυτόχρονα

β) Διαφορές

Ο Πίνακας 5.8 παρουσιάζει τις διαφορές μεταξύ της μεθόδου Scrum και Kanban.

Scrum	**Kanban**
Απαραίτητες οι επαναλήψεις σε ίσα και ορισμένα χρονικά διαστήματα.	Οι επαναλήψεις σε ίσα και ορισμένα χρονικά διαστήματα είναι προαιρετικές. Μπορεί να οριστεί διαφορετικός ρυθμός για τον προγραμματισμό, την έκδοση και τη βελτίωση της διαδικασίας. Η διαδικασία μπορεί να οδηγείται από γεγονότα αντί από χρονικές περιόδους.
Η ομάδα δεσμεύεται ότι θα εκτελέσει ένα συγκεκριμένο φόρτο εργασίας σε κάθε επανάληψη.	Η δέσμευση είναι προαιρετική.
Χρησιμοποιεί την ταχύτητα ανάπτυξης (velocity) ως προεπιλεγμένη μονάδα μέτρησης για τον προγραμματισμό και τη βελτίωση της διαδικασίας.	Χρησιμοποιεί τον χρόνο παράδοσης (lead time) και τον χρόνο κύκλου (cycle time) ως προεπιλεγμένη μονάδα μέτρησης για τον προγραμματισμό και τη βελτίωση της διαδικασίας.
Οι εργασίες πρέπει να διασπώνται σε μικρότερα τμήματα, με κύριο στόχο την ολοκλήρωσή τους σε μία επανάληψη (sprint).	Δεν υπάρχουν περιορισμοί στο μέγεθος των εργασιών, αλλά υπάρχουν πρακτικές που οδηγούν στην κατάτμηση των μεγάλων εργασιών.
Η κατασκευή του διαγράμματος burn down είναι αναγκαία.	Δεν απαιτούνται διαγράμματα συγκεκριμένου τύπου.
Οι εργασίες σε εξέλιξη περιορίζονται έμμεσα (ανά sprint).	Οι υπό ανάπτυξη εργασίες περιορίζονται άμεσα (ανά κατάσταση/στήλη ή ανά ομάδα ή ανά εργαζόμενο).
Δεν επιτρέπεται η προσθήκη νέων εργασιών κατά τη διάρκεια μιας επανάληψης.	Επιτρέπεται η προσθήκη νέων εργασιών.
Οι εργασίες που περιλαμβάνονται στο Sprint Backlog θα εκτελεστούν από μία συγκεκριμένη ομάδα.	Ένας πίνακας Kanban μπορεί να τροφοδοτεί με εργασία πολλές ομάδες ή άτομα.
Προϋποθέτει τρεις ρόλους (Product Owner/Scrum Master/ομάδα ανάπτυξης)	Δεν υπάρχουν συγκεκριμένοι ρόλοι.
Ένας πίνακας Scrum αρχικοποιείται σε κάθε επανάληψη.	Ένας πίνακας Kanban είναι μόνιμος καθ' όλη τη διάρκεια της ανάπτυξης.
Απαιτεί την ιεράρχηση των απαιτήσεων του προϊόντος (product backlog).	Η ιεράρχηση είναι προαιρετική.

Πίνακας 5.8: *Διαφορές της μεθόδου Scrum και της μεθόδου Kanban*

Στις επόμενες παραγράφους θα παρουσιάσουμε αναλυτικά τις βασικές πρακτικές της μεθόδου Kanban.

5.7.3 Η οπτικοποίηση της εργασίας

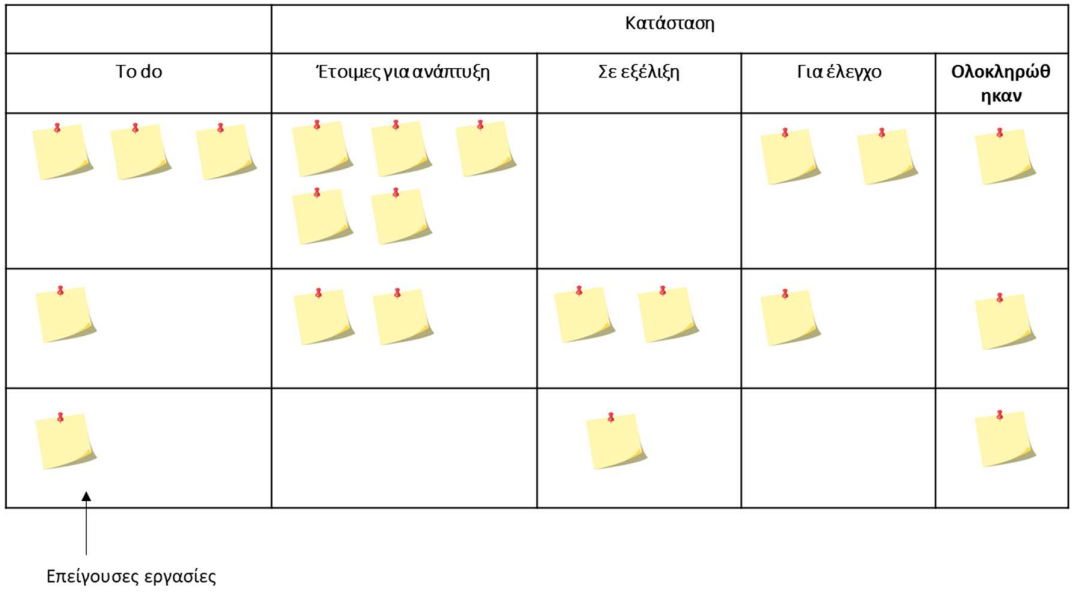
Μια κεντρική ιδέα της μεθόδου Kanban, και ίσως και η πιο χαρακτηριστική της μεθόδου είναι η οπτικοποίηση της εργασίας. Η οπτικοποίηση της εργασίας γίνεται με τη χρήση των πινάκων Kanban, οι οποίοι περιέχουν κάρτες Kanban.

Ο πίνακας Kanban

Η οπτικοποίηση στη μέθοδο Kanban, επιτυγχάνεται με τη χρήση του πίνακα Kanban. Ένας πίνακας Kanban είναι ένας πίνακας, όπου κάθε στήλη αντιστοιχεί σε ένα βήμα της ροής εργασίας. Αντίστοιχα, κάθε κάρτα Kanban αντιπροσωπεύει ένα αντικείμενο εργασίας (work item).

Ο πίνακας Kanban είναι ένα δυνατό εργαλείο επικοινωνίας (information radiator), αφού επιτυγχάνει την εύκολη, γρήγορη και άμεση επικοινωνία. Ένας φυσικός πίνακας Kanban έχει μεγάλες διαστάσεις, είναι τοποθετημένος σε ένα κεντρικό σημείο, περιέχει γραφήματα που δείχνουν την πρόοδο του έργου. Υπάρχουν πολλές εφαρμογές που υλοποιούν ψηφιακούς πίνακες Kanban. Όμως η φυσική μορφή του πίνακα έχει σημαντικά πλεονεκτήματα, αφού ο πίνακας αποτελεί ένα σημείο συνάντησης και διαλόγου της ομάδας. Επιπλέον, οι ψηφιακές εφαρμογές αν και έχουν πολλά πλεονεκτήματα, τείνουν να καθοδηγούν την ομάδα, αυτοματοποιώντας διεργασίες, ενώ ταυτόχρονα θέτουν περιορισμούς για τον τρόπο ανάπτυξης του λογισμικού.

Όπως αναφέραμε οι στήλες του πίνακα Kanban αντιστοιχούν με τη ροή εργασίας που έχει υιοθετήσει η επιχείρηση. Επομένως είναι σημαντικό να είναι ξεκάθαρη η ροή της εργασίας που χρησιμοποιείται. Συνήθως η πρώτη στήλη αφορά πάντα τις εργασίες προς εκτέλεση (to do). Στη συνέχεια, οι επόμενες στήλες αναφέρονται σε επόμενα βήματα της ροής εργασίας και περιλαμβάνουν τις εργασίες που είναι έτοιμες προς ανάπτυξη, αυτές που είναι σε εξέλιξη, αυτές που είναι προς έλεγχο. Θα μπορούσε να πει κάποιος ότι οι εργασίες προς ανάπτυξη με τις εργασίες προς εξέλιξη είναι κοινές. Όμως η ύπαρξη δυο διαφορετικών στηλών διαφοροποιεί την ουρά αναμονής και την ουρά της πραγματικής εργασίας. Η μορφή του πίνακα είναι παρόμοια με αυτή που έχει παρουσιαστεί στην Εικόνα 4.18 αλλά παρουσιάζεται για λόγους πληρότητας και στην Εικόνα 5.13.



Εικόνα 5.13: *Ο πίνακας Kanban*

Ο πίνακας Kanban παρουσιάζει την εξέλιξη και τη ροή της εργασίας με εποπτικό τρόπο καθώς οι κάρτες μετακινούνται από τη μια στήλη στην επόμενη. Το γεγονός ότι στη στήλη «Σε εξέλιξη» υπάρχουν τρεις κάρτες/εργασίες, σημαίνει ότι η ομάδα έργου έχει δυναμικότητα τριών εργασιών. Συνεπώς, κάθε κάρτα Kanban αντιπροσωπεύει ένα μόνο κομμάτι της εργασίας και ο αριθμός των διαθέσιμων καρτών σε ένα φυσικό σύστημα Kanban δείχνει τη δυναμικότητα ενός συστήματος. Αντίστοιχα σε ψηφιακά συστήματα διαχείρισης Kanban θέτουμε όρια στις εργασίες σε εξέλιξη (WIP limits), θέμα που θα συζητηθεί σε αμέσως επόμενη παράγραφο.

Μια νέα εργασία μπορεί να αρχίσει μόνο όταν υπάρχει κάποια διαθέσιμη κάρτα, και μια κάρτα είναι διαθέσιμη μόνο όταν η προηγούμενη εργασία έχει ολοκληρωθεί, πράγμα που σημαίνει ότι οι νέες εργασίες θα πρέπει να περιμένουν στην ουρά μέχρι οι υπάρχουσες εργασίες να έχουν ολοκληρωθεί.

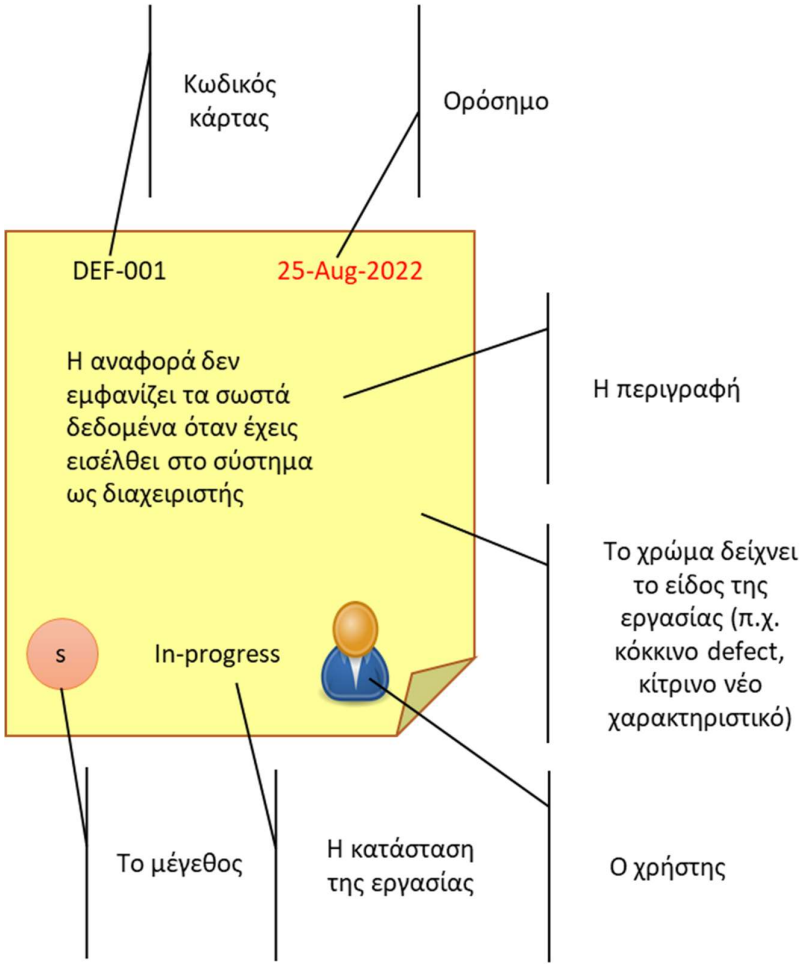
Η κάρτα Kanban

Μια κάρτα Kanban είναι η οπτική αναπαράσταση ενός αντικειμένου εργασίας (work item). Είναι ένα βασικό στοιχείο της μεθόδου Kanban και αντιπροσωπεύει την εργασία που έχει ζητηθεί ή βρίσκεται ήδη σε εξέλιξη (Work In Progress – WIP). Μια κάρτα Kanban περιέχει πληροφορίες σχετικά με την εργασία, την κατάσταση στην οποία βρίσκεται, την περίληψη της εργασίας, τον υπεύθυνο για την εργασία, την προθεσμία υλοποίησης, κ.λπ.

Οι κάρτες Kanban μπορεί να είναι είτε φυσικές είτε ψηφιακές, με χρήση των κατάλληλων εργαλείων. Προφανώς η χρήση των ψηφιακών συστημάτων έχει κερδίσει σημαντικό έδαφος τα τελευταία χρόνια. Σε κάθε περίπτωση υπάρχει πλειάδα προτύπων που χρησιμοποιούνται από διάφορες επιχειρήσεις και οργανισμούς.

Στην απλούστερη μορφή της μια κάρτα Kanban περιέχει για κάθε εργασία (βλέπε Εικόνα 5.14):

- Περιγραφή
- Μέγεθος εργασίας
- Μέλος της ομάδας που εργάζεται ή είναι υπεύθυνο για την εργασία
- Ορόσημο ολοκλήρωσης
- Αν έχει επείγοντα χαρακτήρα
- Κ.λπ.



Εικόνα 5.14: *Η κάρτα Kanban*

5.7.4 Η εργασία σε εξέλιξη

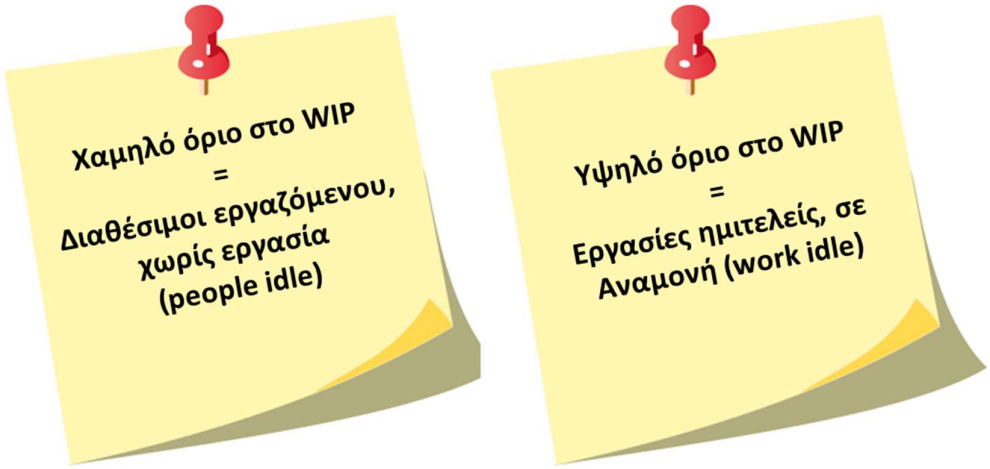
Η εργασία σε εξέλιξη (WIP) είναι με απλά λόγια όλη εργασία που εκτελείται μια δεδομένη στιγμή. Περιλαμβάνονται εργασίες που γίνονται ενεργά μια δεδομένη στιγμή, αλλά και εργασίες που πρέπει να ελεγχθούν, εργασίες που αναμένουν στα εισερχόμενα του ηλεκτρονικού ταχυδρομείου που δεν έχουν ξεκινήσει ακόμα. Γενικότερα, είναι όλες οι ημιτελείς εργασίες που πρέπει να γίνουν ώστε να μπορέσουμε να προσφέρουμε αξία στον τελικό πελάτη.

Επομένως το WIP μπορεί να έχει πολλές δυνατές μορφές. Στην ανάπτυξη λογισμικού το WIP μπορεί να είναι:

- Ιστορίες χρηστών που δεν έχουν αναπτυχθεί ακόμη.
- Μη ελεγμένος κώδικας.
- Κώδικας που δεν έχει ενσωματωθεί στο παραγωγικό σύστημα.
- Κ.λπ.

Ο περιορισμός του WIP είναι μια από τις βασικές αρχές της μεθόδου Kanban. Αυτό δεν σημαίνει ότι θα εργαζόμαστε λιγότερο ή ότι θα γίνει λιγότερη εργασία, αλλά ότι πρέπει να είμαστε συγκεντρωμένοι σε ένα αντικείμενο εργασίας και να κάνουμε λιγότερη δουλειά ταυτόχρονα. Ο περιορισμός του WIP, για κάθε εργαζόμενο έχει αποδειχθεί ότι βοηθά στην ολοκλήρωση περισσοτέρων εργασιών για μια εύλογη χρονική διάρκεια (Hammarberg & Sunden, 2014).

Επομένως είναι ιδιαίτερα σημαντικό να περιορίσουμε το WIP ή ακόμη καλύτερα να βρούμε το σωστό σημείο ισορροπίας. Η ισορροπία προφανώς εξαρτάται από τη σπουδαιότητα του έργου, την επείγουσα φύση του, το μέγεθος της ομάδας, τη διαθεσιμότητα των μελών της ομάδας, κ.λπ. Γενικά, θα πρέπει να στοχεύουμε στο να έχουμε ρεαλιστικά κατά προτίμηση χαμηλά όρια για το WIP, και ότι είναι προτιμότερο να έχουμε διαθέσιμους εργαζόμενους, χωρίς εργασία, παρά πολλές εργασίες σε εξέλιξη. Προφανώς υπάρχει σχέση μεταξύ των χαμηλών ορίων και τις διαθεσιμότητας των εργαζομένων.



Εικόνα 5.15: *Τα όρια του WIP*

Ο περιορισμός του WIP μπορεί να γίνει με εφαρμογή διάφορων τεχνικών ή κανόνων. Για παράδειγμα, ένας απλός κανόνας για τον περιορισμό των εργασιών ενός εργαζομένου είναι να ξεκινάει μια νέα εργασία μόνο αν έχει ολοκληρωθεί η προηγούμενη.

Εναλλακτικά μπορούμε να ορίσουμε όρια για το WIP είτε συνολικά για την ομάδα είτε για κάθε βήμα της ροής εργασίας (για κάθε στήλη του πίνακα Kanban). Εάν οι περιορισμοί στο WIP είναι σε επίπεδο ομάδας, τότε προάγεται η συνεργασία και η συνυπευθυνότητα των μελών της ομάδας έργου, ενώ όταν εστιαζόμαστε στη ροή εργασίας μπορούμε να βρούμε τα προβληματικά σημεία ή τα σημεία που προκαλούν καθυστερήσεις.

5.7.5 Διαχείριση της ροής εργασιών

Μια ομάδα Kanban παράγει προϊόντα/λύσεις/λογισμικό για την επιχείρηση με μια σταθερή ροή. Στο πλαίσιο της διαχείρισης ενός έργου, η ροή ορίζεται ως ο τρόπος με τον οποίο ένα αντικείμενο, μια πληροφορία ή ένα κομμάτι εργασίας κινείται μέσα σε μια επιχειρηματική διεργασία (business process).

Η διαχείριση της ροής της εργασίας είναι μία από τις βασικές πρακτικές του Kanban που έχει ως στόχο την παρακολούθηση και μέτρηση του τρόπου ροής της εργασίας, αλλά και διατήρηση της απρόσκοπτης κίνησης της εργασίας. Υπάρχουν πολλοί τρόποι για να βελτιώσουμε τη ροή εργασίας μέσα σε ένα σύστημα:

- Μείωση των εργασιών σε εξέλιξη (WIP) που παρουσιάσαμε στην προηγούμενη παράγραφο.
- Μείωση του χρόνου αναμονής των εργασιών, με προετοιμασία τμημάτων εργασιών και με πρόσληψη προσωπικού, όπου είναι αναγκαίο.
- Με εξάλειψη των εμποδίων όπου αυτά εμφανίζονται. Συνήθως τα εμπόδια προέρχονται από εξωγενείς παράγοντες εκτός του ελέγχου μας.

5.7.6 Οι πολιτικές της διαδικασίας πρέπει να είναι ρητές

Μια ομάδα Kanban καθορίζει τις πολιτικές που αφορούν τις εργασίες σε εξέλιξη, την ιεράρχηση (αναπλήρωση της ουράς εργασίας), τον ρυθμό παράδοσης, τον χρόνο παράδοσης για τις κατηγορίες των

υπηρεσιών, και πώς τα ζητήματα θα πρέπει να αντιμετωπιστούν / κλιμακωθούν. Οι πολιτικές αυτές πρέπει να είναι ρητές και να αντικατοπτρίζουν την κατάσταση που αντιμετωπίζει η ομάδα. Επίσης πρέπει να είναι γνωστές τόσο στην ομάδα όσο και στους εμπλεκόμενους φορείς, και αρκετά εύπλαστες, έτσι ώστε να μπορούν να εξελιχθούν με την πάροδο του χρόνου.

5.8 Η ακεραιότητα του συστήματος

Σύμφωνα με τις αρχές της λιτής ανάπτυξης συστημάτων, ένα πρόβλημα μπορεί να εντοπιστεί είτε μετά την εμφάνισή του με τη μορφή ελαττώματος (defect), είτε μπορεί να εξαλειφθεί πριν το σύστημα μπει σε λειτουργία, ακυρώνοντας εκ των προτέρων τις αιτίες που οδηγούν σε αυτό πρόβλημα.

Στα τέλη της δεκαετίας του 1980, ο Clark και Fujimoto (1991). ξεκίνησαν να μελετούν πώς ορισμένες επιχειρήσεις αναπτύσσουν με συνέπεια ανώτερα ποιοτικώς προϊόντα σε σχέση με άλλες και διαπίστωσε ότι η βασική διαφορά ήταν η ακεραιότητα του προϊόντος (product integrity). Διαπίστωσε ότι η ακεραιότητα του προϊόντος έχει δύο διαστάσεις: την εξωτερική και την εσωτερική ακεραιότητα, τις οποίες και ονόμασε αντιληπτή ακεραιότητα (perceived integrity) και εννοιολογική ακεραιότητα (conceptual integrity). Η αντιληπτή ακεραιότητα σημαίνει ότι το σύνολο του προϊόντος επιτυγχάνει μια ισορροπία λειτουργίας, χρηστικότητας, αξιοπιστίας και οικονομίας που ικανοποιεί τους πελάτες. Αντίστοιχα, η εννοιολογική ακεραιότητα σημαίνει ότι οι κεντρικές έννοιες/συστατικά του συστήματος συνεργάζονται ως ένα ομαλό και συνεκτικό σύνολο. Όταν τα συστατικά (components) του συστήματος ταιριάζουν και λειτουργούν καλά μεταξύ τους, η αρχιτεκτονική του συστήματος επιτυγχάνει μια αποτελεσματική ισορροπία μεταξύ της ευελιξίας (flexibility), της δυνατότητας συντήρησης (maintainability), της αποτελεσματικότητας (efficiency) και της απόκρισης (responsiveness). Για παράδειγμα, όταν ένας ιστότοπος μιας αεροπορικής εταιρείας έχει δύο διαφορετικά συστήματα κρατήσεων, αυτό είναι μια σαφής ένδειξη ότι υπάρχουν δύο σαφώς διαφορετικές αρχιτεκτονικές που χρησιμοποιούνται για την πραγματοποίηση μιας κράτησης. Επίσης, η εννοιολογική ακεραιότητα είναι προϋπόθεση για την αντιληπτή ακεραιότητα. Συνεπώς, όταν ένα σύστημα δεν έχει ένα συνεπές σύνολο ιδεών σχεδίασης, αυτό έχει αντίκτυπο στη χρηστικότητα, κ.λπ. (Poppendieck & Poppendieck, 2003)

5.9 Η βελτιστοποίηση του συστήματος συνολικά

Ένα σύστημα αποτελείται από αλληλοεξαρτώμενα και αλληλοεπιδρώντα μέρη που ενώνονται για έναν σκοπό. Ένα σύστημα δεν είναι απλώς το άθροισμα των μερών του - είναι το προϊόν των αλληλεπιδράσεών τους. Τα καλύτερα συστατικά δεν δημιουργούν απαραίτητα και ένα καλύτερο σύστημα, αφού η ικανότητα ενός συστήματος να επιτύχει τον σκοπό του εξαρτάται και από το πόσο καλά συνεργάζονται τα συστατικά μεταξύ τους. Η κατανόηση των στοιχείων της συστημικής σκέψης βοηθά τους ηγέτες και τις ομάδες να αναγνωρίσουν το «γιατί» και το «τι» των πράξεών τους, καθώς και τον αντίκτυπο που έχουν αυτές στους γύρω τους. Αυτό οδηγεί σε μια πιο ευέλικτη επιχείρηση που μπορεί να προσαρμοστεί καλύτερα στον διαρκώς μεταβαλλόμενο επιχειρηματικό χώρο.

Για να κατανοήσουμε καλύτερα την παραπάνω έννοια θα πρέπει να κάνουμε τις εξής παρατηρήσεις:

1) Η ανάπτυξη λογισμικού αποτελεί βασική συνιστώσα της ανάπτυξη ψηφιακών συστημάτων
2) Η επιχείρηση για την οποία αναπτύσσεται το ψηφιακό σύστημα είναι και αυτή ένα σύστημα
3) Το ψηφιακό σύστημα έχει ως στόχο τη βελτιστοποίηση της ροής αξίας.

Τα μέλη της ομάδας πρέπει να κατανοούν ξεκάθαρα τα όρια του συστήματος, τι είναι και πώς αλληλοεπιδρά με το περιβάλλον και τα συστήματα γύρω από αυτό. Η βελτιστοποίηση ενός συστατικού δεν βελτιστοποιεί το σύστημα. Τα συστατικά μπορούν να γίνουν εγωιστικά και να δεσμεύσουν τους πόρους - υπολογιστική ισχύ, μνήμη, ηλεκτρική ενέργεια, οτιδήποτε άλλο- που χρειάζονται άλλα συστατικά. Συνεπώς η βέλτιστη λειτουργία ενός συστήματος, προϋποθέτει την καλύτερη κατανόηση της συμπεριφοράς σε υψηλότερο επίπεδο π.χ. αυτό της επιχειρησιακής αρχιτεκτονικής (https://www.scaledagileframework.com/apply-systems-thinking/).

Ο στοχευμένος σχεδιασμός (intentional design) είναι θεμελιώδης για τη συστημική σκέψη, αφού η αξία ενός συστήματος περνά μέσα από τις διασυνδέσεις του[7]. Αυτές οι διεπαφές και οι εξαρτήσεις που δημιουργούνται είναι κρίσιμες για την παροχή της τελικής αξίας προς τον πελάτη και συνεπώς είναι ζωτικής σημασίας. Επιπλέον, ένα σύστημα δεν μπορεί να εξελιχθεί ταχύτερα από το πιο αργό σημείο ολοκλήρωσής του. Όταν επομένως έχουμε αλλαγές, όσο πιο γρήγορα ενσωματώνονται τόσο πιο γρήγορα μπορούμε να αξιολογήσουμε το πλήρες σύστημα.

Η τοπική βελτιστοποίηση, δηλαδή η βελτιστοποίηση μιας λειτουργίας ή ενός υποσυστήματος, είναι επίσης ένα σοβαρό ζήτημα στην ανάπτυξη λογισμικού και αποτελεί συχνά μια αυτοεκπληρούμενη προφητεία. Στο βιβλίο τους, η Mary και ο Tom Poppendieck περιγράφουν δύο φαύλους κύκλους στους οποίους συχνά πέφτουν οι ομάδες που εφαρμόζουν τη λιτή προσέγγιση.

Το πρώτο πρόβλημα είναι η απελευθέρωση (release) μη επαρκώς ελεγμένου κώδικα για χάρη της ταχύτητας. Όταν η ομάδα ανάπτυξης πιέζεται να παραδώσει το λογισμικό, χωρίς να έχει ελεγχθεί επαρκώς, τότε η ομάδα ανάπτυξης απελευθερώνει κώδικα που δεν πληροί τις απαιτήσεις ποιότητας. Αυτό έχει ως συνέπεια την αύξηση της συνολικής πολυπλοκότητας του συστήματος (baseline), καθώς και την αύξηση των σφαλμάτων που υπάρχουν στο σύστημα και δεν έχουν ακόμη ανακαλυφθεί μέσω του ελέγχου. Προφανώς περισσότερα σφάλματα σημαίνει, περισσότερη εργασία για να διορθωθούν, γεγονός που ασκεί ακόμη μεγαλύτερη πίεση στην ομάδα ανάπτυξης για γρήγορη ανάπτυξη.

Το δεύτερο πρόβλημα σχετίζεται με τους ελέγχους του λογισμικού (testing). Όταν υπάρχει φόρτος εργασίας ο χρόνος που μεσολαβεί μεταξύ της ανάπτυξης του λογισμικού και του ελέγχου είναι μεγάλος, δημιουργείται δηλαδή ένας μεγάλος χρόνος κύκλου (cycle time). Κατά τη διάρκεια του χρόνου αυτού οι προγραμματιστές συνεχίζουν να γράφουν κώδικα που μπορεί να βασίζεται στον μη ελεγμένο – πιθανά ελλαττωματικό κώδικα, με αποτέλεσμα να δημιουργούνται περισσότερα σφάλματα και συνεπώς να απαιτούνται περισσότερες δοκιμές.

Το αντίδοτο στην τοπική βελτιστοποίηση είναι η βελτιστοποίηση του συνόλου του συστήματος. Η συνολική βελτιστοποίηση ή η εστίαση στη συνολική εικόνα αποτελεί μια βασική αρχή της λιτής ανάπτυξης συστημάτων που στοχεύει στην εξάλειψη αυτών των φαύλων κύκλων λειτουργώντας με καλύτερη κατανόηση της δυναμικότητας της ομάδας και του αντίκτυπου αυτού του είδους των προβλημάτων στη ροή εργασίας.

Υπάρχει μια δεύτερη πτυχή στη συστημική σκέψη: η επιχείρηση που αναπτύσσει το σύστημα, η διοίκηση και οι διεργασίες της επιχείρησης, οι εργαζόμενοι αποτελούν επίσης ένα σύστημα, το οποίο θα πρέπει να διαχειριστούμε. Αν παραλείψουμε αυτή την πτυχή τα συστατικά της επιχείρησης για την οποία κατασκευάζουμε το λογισμικό, θα βελτιστοποιηθούν τοπικά και θα γίνουν εγωιστικά, περιορίζοντας την ταχύτητα και την ποιότητα της παράδοσης αξίας. Αυτό οδηγεί σε ένα αριθμό παρατηρήσεων που σχετίζονται με τη συστημική σκέψη:

- Η κατασκευή πολύπλοκων συστημάτων είναι μια κοινωνική προσπάθεια. Ως εκ τούτου, οι ηγέτες πρέπει να δημιουργούν ένα περιβάλλον όπου οι άνθρωποι συνεργάζονται άκοπα, με σκοπό το καλύτερο δυνατό αποτέλεσμα.
- Οι προμηθευτές και οι πελάτες αποτελούν αναπόσπαστο κομμάτι της ροής αξίας. Συνεπώς, πρέπει να αντιμετωπίζονται ως συνεργάτες, αναπτύσσοντας σχέσεις εμπιστοσύνης.
- Η βελτιστοποίηση ενός συστατικού δεν βελτιστοποιεί το σύστημα και δεν βελτιώνει κατ' ανάγκη τη ροή αξίας.
- Όπως συμβαίνει και με τα φυσικά συστήματα, η αξία του συστήματος περνά μέσω των διεπαφών του. Η επιτάχυνση της ροής αξίας επιτυγχάνεται με την εξάλειψη των σιλό και τη δημιουργία διαλειτουργικών συστημάτων (π.χ. Agile Release Trains - ARTs, Solution Trains).

[7] Ο στοχευμένος σχεδιασμός είναι μια διαδικασία που διερευνά μια ποικιλία αλληλεπιδράσεων και σημείων επαφής με τους χρήστες.

Τέλος μια τρίτη πτυχή είναι ότι κάθε επιχείρηση υλοποιεί μια ροή αξίας και επιδιώκει να βελτιστοποιήσει συνολικά τη ροή αξίας, όχι μόνο κάποιες μεμονωμένες λειτουργίες. Αν η επιχείρηση προσπαθήσει να βελτιστοποιήσει ένα μικρότερο τμήμα της αλυσίδας, τότε θα υπάρξει πρόβλημα ή τουλάχιστον δεν θα υπάρξει η αναμενόμενη βελτίωση. Δεν θα πρέπει να ξεχνάμε ότι για να δημιουργήσουμε αξία για τον πελάτη, πρέπει να αντιληφθούμε την αξία του προϊόντος/υπηρεσίας που παρέχουμε, υπό την οπτική του πελάτη. Αυτή ακριβώς είναι η φιλοσοφία της λιτής διοίκησης όσον αφορά στην εξυπηρέτηση του πελάτη, και έχει να κάνει με την αλυσίδα αξίας συνολικά, όπως την αντιλαμβάνεται ο πελάτης/καταναλωτής. Η αλυσίδα αυτή δε σταματάει τη στιγμή της πώλησης, αλλά επεκτείνεται και στη χρήση του λογισμικού/υπηρεσίας. Δηλαδή ξεκινά από τη στιγμή που ο πελάτης εκδηλώνει επιθυμία επίλυσης ενός προβλήματος, έως την επίλυση αυτού του προβλήματος, με τη μέγιστη ταχύτητα και χωρίς προβλήματα για εκείνον.

Επισημαίνεται ότι όταν υπάρχουν πολλά τμήματα της επιχείρησης που εμπλέκονται στη ροή αξίας τότε είναι αναμενόμενες καθυστερήσεις που οφείλονται σε μετακύλιση της ευθύνης από το ένα τμήμα στο άλλο (hand-off), και επιπλέον δεν υπάρχει ένας υπεύθυνος που να ασχολείται με την πραγματική ανάγκη του πελάτη. Μια λύση σε αυτό το πρόβλημα είναι η οργάνωση μιας διεπιστημονικής ομάδας ανάπτυξης λογισμικού που να είναι ικανή να αναπτύξει το λογισμικό από την αρχή μέχρι το τέλος, χωρίς αναφορά σε άλλες ομάδες.

Για να γίνει κατανοητή αυτή η λογική, έστω ένας πελάτης ο οποίος αντιμετωπίζει πρόβλημα με την εκτύπωση μια αναφοράς. Ο τεχνικός του εκτυπωτή που έχει αναλάβει την επίλυση του προβλήματος μετά από έλεγχο, δηλώνει ότι ο εκτυπωτής είναι συνδεδεμένος και λειτουργικός. Η πρότασή του είναι ότι το πρόβλημα του πελάτη θα πρέπει να επιλυθεί από το τμήμα υποστήριξης του λογισμικού. Η ορθή προσέγγιση θα ήταν, η άμεση συνεργασία μεταξύ των δύο τμημάτων, χωρίς την παρέμβαση του πελάτη/χρήστη (τουλάχιστον έως την εύρεση της λύσης και τον έλεγχό της).

Η συστημική σκέψη απαιτεί επίσης μια νέα προσέγγιση στη διοίκηση, μια προοπτική όπου τα διευθυντικά στελέχη επιλύουν προβλήματα, η οπτική τους γωνία είναι μακρόπνοη, εξαλείφουν προληπτικά τα εμπόδια και υλοποιούν τις αλλαγές που είναι απαραίτητες για τη βελτίωση των συστημάτων και της απόδοσης.

Βιβλιογραφία/Αναφορές

Anderson, D. J. (2003). *Agile management for software engineering: Applying the theory of constraints for business results*. Prentice Hall Professional.

Arnold, J. J., & Yüce, Ö. (2013). Black Swan Farming Using Cost of Delay: Discover, Nurture and Speed Up Delivery of Value. In *2013 Agile Conference* (pp. 101-116). IEEE.

Ashmore, S., & Runyan, K. (2014). *Introduction to agile methods*. Addison-Wesley Professional.

Brechner, E. (2015). *Agile project management with Kanban*. Pearson Education.

Bhat, U. N. (2008). An introduction to queueing theory: modeling and analysis in applications (Vol. 36). Boston: Birkhäuser.

Clark, K. B., & Fujimoto, T. (1991). New product development performance. *Harvard Business School Press, Cambridge, MA*.

Goljan, J. (2021). Developing a Cost of Delay (CoD) Framework for the DoD & Analyzing the Current State of Air Force Agile Cost Estimation.

Hammarberg, M., & Sunden, J. (2014). *Kanban in action*. Manning Publications Co.

Janes, A., & Succi, G. (2014). Lean software development in action. In *Lean software development in action*. Springer, Berlin, Heidelberg.

Kanabanize.com (n.d.) *Cost of Delay: the Economic Impact of a Delay in Project Delivery*. Ανακτήθηκε 10η Αυγούστου, 2022 από https://kanbanize.com/lean-management/value-waste/cost-of-delay

Krafcik, J. F. (1988). Triumph of the lean production system. *Sloan Management Review, 30*(1), pp. 41-51.

Martin, R. C. (2003). *Agile software development: principles, patterns, and practices*. Prentice Hall PTR.

Ohno, T., & Bodek, N. (2019). Toyota production system: beyond large-scale production. Productivity press.

Oosterwal, D. P. (2010). The lean machine: how Harley-Davidson drove top-line growth and profitability with revolutionary lean product development. Amacom.

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: an agile toolkit*. Addison-Wesley.

Poppendieck, M., & Poppendieck, T. (2007). *Implementing lean software development: from concept to cash*. Pearson Education.

Putnik, G.D. and Putnik, Z. (2012), "Lean vs agile in the context of complexity management in organizations", *The Learning Organization*, Vol. 19 No. 3, pp. 248-266. https://doi.org/10.1108/09696471211220046

Rehkopf, M. (n.d.). Kanban vs. scrum: which agile are you?. Ανακτήθηκε 10η Αυγούστου 2022 από https://www.atlassian.com/agile/kanban/kanban-vs-scrum

Rother, M., & Shook, J. (2003). Learning to see: value stream mapping to add value and eliminate muda. Lean enterprise institute.

SAFE (2021). Principle #2 – Apply systems thinking. Ανακτήθηκε 16η Αυγούστου, 2022 από https://www.scaledagileframework.com/apply-systems-thinking/.

Thimbleby, H. (1988). Delaying commitment (programming strategy). *IEEE Software*, *5*(3), 78-86.

Thomas, D., & Hunt, A. (2019). *The Pragmatic Programmer: your journey to mastery*. Addison-Wesley Professional.

Ward, A. C., & Sobek II, D. K. (2014). *Lean product and process development*. Lean Enterprise Institute.

Womack, J. P., & Jones, D. T. (2003). Lean thinking: Banish waste and create wealth in your corporation. London: Free Press.

Womack, J. P., Jones, D. T., & Roos, D. (1990). The machine that change the world. *Rawson Associates, NY*.

Αδαμίδης, Ε. (2016). Λιτή παραγωγή και JIT. Περιλαμβάνεται στο Αδαμίδης, Ε. 2016. *Σχεδιασμός και Διοίκηση Βιομηχανικών Μονάδων*. Kallipos, Open Academic Editions. chapter 8. http://hdl.handle.net/11419/6280

Γερογιάννης, Β., Κακαρόντζας, Γ., Καμέας, Α., Σταμέλος, Γ., & Φιτσιλής, Π. (2006). Αντικειμενοστρεφής Ανάπτυξη Λογισμικού με τη UML. Αθήνα: Κλειδάριθμος.

Γερογιάννης, Β. (2013). *Η μέθοδος Kanban*. Print SMEs Project. http://sprint.teilar.gr

Καραγιάννη, Ε. (2020). Εφαρμογή της Lean Φιλοσοφίας σε εταιρικό Τμήμα Ανάπτυξης Λογισμικού, Athens MBA.

Μέρμιγκα, Ε. (2020). Lean Management: η περίπτωση του αιματολογικού εργαστηρίου του νοσοκομείου Παπαγεωργίου. Πανεπιστήμιο Μακεδονίας.

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Τι επιτυγχάνουμε με τη λιτή διοίκηση;

Απάντηση/Λύση

Η εφαρμογή της λιτής διοίκησης επιτυγχάνει τα ακόλουθα:

- Μειώνουμε τον χρόνο εκτέλεσης των διεργασιών
- Βελτιώνουμε τον χρόνο παράδοσης των προϊόντων ή των υπηρεσιών
- Μειώνουμε ή εξαλείφουμε την πιθανότητα δημιουργίας ελαττωμάτων
- Μειώνουμε τα επίπεδα των αποθεμάτων απογραφής και
- Βελτιστοποιούμε τη χρήση των πόρων.

Κριτήριο αξιολόγησης 2

Να περιγράψετε τη γενική φιλοσοφία/λογική της λιτής παραγωγής. Να τη συγκρίνετε με αυτήν που διέπει τα συμβατικά συστήματα παραγωγής.

Απάντηση/Λύση

Οι διαφορές μεταξύ παραδοσιακής παραγωγής και της λιτής παραγωγής είναι οι ακόλουθες:

Παραδοσιακή παραγωγή: Η παραγωγή που βασίζεται σε πρόβλεψη πωλήσεων (Push).

Λιτή παραγωγή: Η παραγωγή καθοδηγείται από τη ζήτηση των πελατών. Τα προϊόντα παράγονται μόνο όταν γίνεται παραγγελία.

Παραδοσιακή παραγωγή: Τα προβλήματα αντιμετωπίζονται ως ακριβώς αυτό, προβλήματα.

Λιτή παραγωγή: Τα προβλήματα αντιμετωπίζονται ως ευκαιρίες για βελτίωση συχνά μέσω της ανάλυσης της βασικής αιτίας.

Παραδοσιακή παραγωγή: Οι εργασίες σε εξέλιξη (WIP) θεωρούνται ως κανονικό μέρος της παραγωγής.

Λιτή παραγωγή: Η ύπαρξη WIP είναι ένα σημάδι ότι κάποια διεργασία πρέπει να βελτιωθεί και θεωρείται ως σπατάλη που πρέπει να μειωθεί ή να εξαλειφθεί (το ίδιο ισχύει και για το απόθεμα).

Παραδοσιακή παραγωγή: Βελτίωση του συστήματος (αγνοώντας όλα τα είδη της σπατάλης στις διεργασίες).

Λιτή παραγωγή: Βελτίωση του συστήματος μέσω α) της εξάλειψης της σπατάλης και β) της βελτίωσης των τρεχουσών διαδικασιών.

Παραδοσιακή παραγωγή: Η διοίκηση είναι ο κύριος μοχλός της αλλαγής.

Λιτή παραγωγή: Όλοι οι εργαζόμενοι είναι υπεύθυνοι, είναι εκπαιδευμένοι στις αρχές της λιτής διοίκησης και ενθαρρύνονται να αναζητούν τρόπους βελτίωσης των διεργασιών.

Παραδοσιακή παραγωγή: Εάν μια διεργασία λειτουργεί κανονικά δεν ασχολούμαστε μαζί της.

Λιτή παραγωγή: Αναζητάμε πάντα τρόπους βελτίωσης των διεργασιών.

Παραδοσιακή παραγωγή: Η εργασία είναι τυποποιημένη εργασία και ο κάθε εργαζόμενος είναι εξειδικευμένος.

Λιτή παραγωγή: Όλοι εκτελούν την ίδια εργασία με τον ίδιο ακριβώς τρόπο μέχρι να ανακαλυφθεί ένας νέος καλύτερος τρόπος.

Παραδοσιακή παραγωγή: Επικεντρώνεται στην εκπαίδευση και βασίζεται στους ανθρώπους για να μην κάνουν λάθη.

Λιτή παραγωγή: Επικεντρώνεται στη δημιουργία διεργασιών που δεν επιτρέπουν να γίνουν σφάλματα (ένας εργαζόμενος δεν μπορεί να κάνει λάθος ή είναι δύσκολο να το κάνει).

Παραδοσιακή παραγωγή: Βασίζεται στη συστημική σκέψη (βλέπει τον οργανισμό στο σύνολό του), και συχνά αγνοεί ή δεν μπορεί να δει τις ευκαιρίες για βελτίωση.

Λιτή παραγωγή: Θεωρεί τον οργανισμό ως μια σειρά αλληλένδετων διεργασιών που μπορούν και πρέπει να βελτιωθούν.

Κριτήριο αξιολόγησης 3

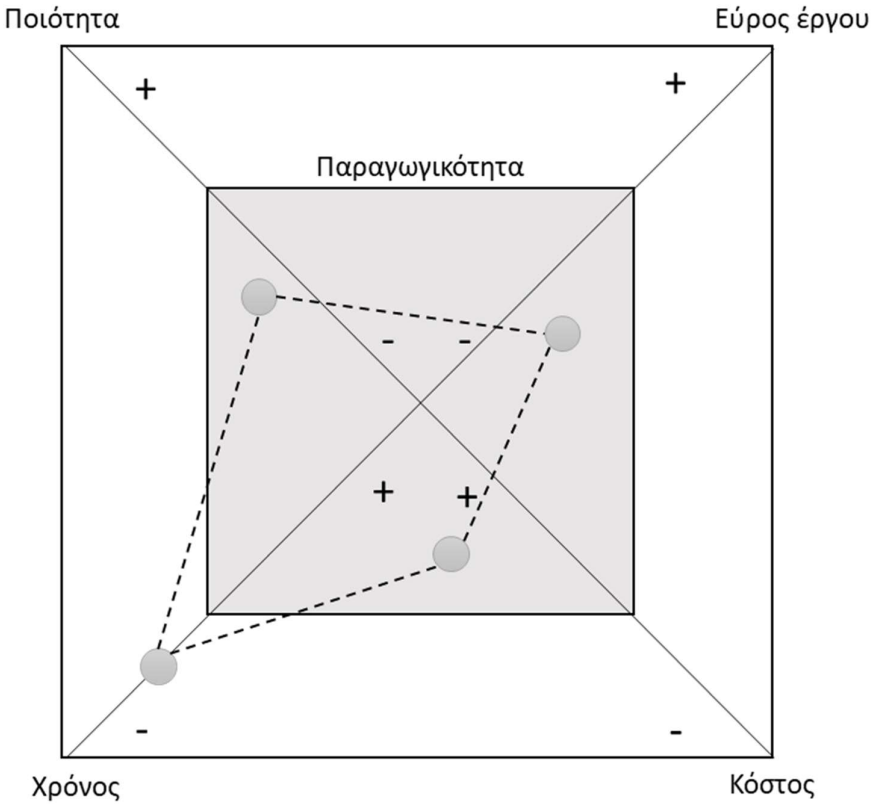
Ο Harry Sneed παρουσίασε ένα μοντέλο που ονομάζεται «τετράγωνο του διαβόλου» (Janes & Succi, 2014). για να εκφράσει πόσο δύσκολο είναι να παρέχεις τη σωστή ποιότητα και τον σωστό αριθμό χαρακτηριστικών, την κατάλληλη χρονική στιγμή (ορόσημο) και σύμφωνα με τον προϋπολογισμό.

Το τετράγωνο του διαβόλου περιγράφει ότι δεν είναι δυνατό (ή είναι τουλάχιστον δύσκολο) να επιλέξουμε αυθαίρετα την ποιότητα, το εύρος, τον χρόνο ανάπτυξης και το κόστος ανάπτυξης ενός έργου λογισμικού, καθώς η παραγωγικότητα (που αντιπροσωπεύεται από την γκρίζα επιφάνεια) παραμένει αμετάβλητη βραχυπρόθεσμα.

Έστω ότι ένας πελάτης θέλει ένα έργο να ολοκληρωθεί νωρίτερα από το προγραμματισμένο. Είναι σαν να θέλει να σύρει μια γωνία του παραλληλογράμμου παραγωγικότητας στη μία πλευρά —την πλευρά του «χρόνου»— προς το μείον. Το τετράγωνο του διαβόλου μας λέει ότι η επιφάνεια της γκρίζας περιοχής πρέπει να παραμείνει ίδια. Επομένως, μια ή περισσότερες από τις άλλες κορυφές πρέπει να κινηθούν προς το κέντρο. Συνεπώς, :

- **είτε η ποιότητα μειώνεται,**
- **είτε το εύρος μειώνεται (λιγότερα χαρακτηριστικά) ή**
- **το κόστος αυξάνεται.**

Ποιος θα πρέπει να είναι ο στόχος της λιτής φιλοσοφίας σε αυτό το πλαίσιο; Ποια είναι τα αναμενόμενα αποτελέσματα στην παραγωγικότητα όταν μια εταιρεία εισάγει τη λιτή φιλοσοφία;



Απάντηση/Λύση

Η λιτή φιλοσοφία προσανατολίζεται στην αύξηση της παραγωγικότητας. Επομένως το ζητούμενο είναι να αυξήσουμε την επιφάνεια του τετραγώνου της παραγωγικότητας. Μπορούμε να αυξήσουμε την παραγωγικότητά μας με:

- Την καλύτερη οργάνωση της εργασίας μας με τέτοιο τρόπο ώστε να μειώσουμε τη σπατάλη,
- Τη βελτίωση των γνώσεών μας σχετικά με τη διαδικασία ανάπτυξης και
- Την αύξηση της αξίας που παρέχουμε στον πελάτη.

Αυτό σημαίνει ότι οι πόροι που έχουμε θα παραμένουν οι ίδιοι. Θα πρέπει όμως να χρησιμοποιούνται με πιο αποτελεσματικό και αποδοτικό τρόπο.

Κριτήριο αξιολόγησης 4

Ποιες θεωρούνται ημιτελείς εργασίες στο λογισμικό (Partially Done Work);

Απάντηση/Λύση

Ημιτελείς εργασίες στην ανάπτυξη λογισμικού θεωρούνται οι ακόλουθες:

- Προδιαγραφές/Απαιτήσεις που μένουν στο στάδιο της καταγραφής και δεν προχωράνε στο επόμενο βήμα της ανάπτυξης. Όσο πιο πολύ αργεί η ανάπτυξη του λογισμικού, τόσο πιο πιθανό είναι ότι οι προδιαγραφές αυτές θα απαρχαιωθούν ή θα πρέπει να αλλάξουν αφού δεν θα συμβαδίζουν με τις ανάγκες των χρηστών.

- Μη ελεγμένος κώδικας. Ο κώδικας πρέπει να ελέγχεται άμεσα από τους προγραμματιστές και να μην παραμένει η εκκρεμότητα του ελέγχου.
- Μη τεκμηριωμένος κώδικας. Η τεκμηρίωση του κώδικα θα πρέπει να γίνεται κατά τη διάρκεια της συγγραφής του κώδικα. Επίσης ο ίδιος ο κώδικας θα πρέπει να είναι έτσι δομημένος και με σύνταξη τέτοια, ώστε να είναι εύκολη η «ανάγνωσή» του.
- Λογισμικό που δεν μεταφέρεται στην παραγωγή. Το ζητούμενο είναι το ολοκληρωμένο λογισμικό να μπει σε παραγωγική λειτουργία όσο πιο σύντομα γίνεται. Πολλές φορές αυτό είναι δύσκολο, διότι το νέο software μπορεί να προκαλέσει αναστάτωση στη δουλειά των χρηστών, χωρίς αυτό να σημαίνει προβλήματα της εφαρμογής. Ενδεχομένως να πρέπει οι χρήστες να «βγουν» εκτός συστήματος, να σταματήσουν οι εργασίες τους για κάποιο διάστημα έως ότου περαστούν οι αλλαγές κλπ. Παρόλα αυτά η προσπάθεια όλων θα πρέπει να είναι η μεταφορά στην παραγωγή όσο πιο σύντομα γίνεται μετά από την οριστικοποίηση του ελέγχου από τους χρήστες.

Κριτήριο αξιολόγησης 5

Σκεφτείτε παραδείγματα από τη ζωή σας ή την εργασία σας όταν έχετε χρησιμοποιήσει επιλογές για να καθυστερήσετε τη λήψη αποφάσεων. Για παράδειγμα, έχετε πληρώσει ποτέ επιπλέον προκαταβολή για να κλειδώσετε ένα χαμηλό επιτόκιο καθώς διαπραγματευόσασταν ένα δάνειο; Πόσο αποτελεσματική ήταν για εσάς αυτή η προσέγγιση; Συμπληρώστε τον παρακάτω πίνακα:

Παράδειγμα όπου κρατάμε τις επιλογές μας ανοικτές	**Πολύ ευνοϊκό αποτέλεσμα**	**Το αποτέλεσμα ήταν ουδέτερο**	**Το αποτέλεσμα δεν ήταν ευνοϊκό**
Διαπραγμάτευση δανείου	**X**		
Παράδειγμα 1			
Παράδειγμα 2			
Παράδειγμα 3			

Απάντηση/Λύση

Παράδειγμα όπου κρατάμε τις επιλογές μας ανοικτές	**Πολύ ευνοϊκό αποτέλεσμα**	**Το αποτέλεσμα ήταν ουδέτερο**	**Το αποτέλεσμα δεν ήταν ευνοϊκό**
Διαπραγμάτευση δανείου	**X**		
Δημιουργία sprint backlog	**X**		
Κλείσιμο αεροπορικού εισιτηρίου			**X**
Καθυστερημένη απόφαση αγοράς εξοπλισμού	**X**		

Η δημιουργία του sprint backlog είναι ένα καλό παράδειγμα καθυστερημένης λήψης απόφασης, αφού η απόφαση για το ποιες ιστορίες χρήστη θα συμπεριληφθούν στο sprint λαμβάνεται την τελευταία στιγμή.

Στο κλείσιμο ενός αεροπορικού εισιτηρίου θα πρέπει να εξετάσουμε τις συνθήκες καθώς το αποτέλεσμα μπορεί να είναι ευνοϊκό σε περίπτωση που η ανάγκη μετακίνησης εκλείψει ή αρνητικό σε περίπτωση που η καθυστερημένη κράτηση αυξήσει το κόστος μετακίνησης.

Συνήθως η καθυστερημένη απόφαση στην αγορά εξοπλισμού είναι θετική, αφού ένα νέο μοντέλο μπορεί να βγει στην αγορά, η να υπάρξει πτώση τιμής.

Κριτήριο αξιολόγησης 6

Ποια είναι τα πλεονεκτήματα λήψης αποφάσεων την τελευταία στιγμή (Last Responsible Moment);

Απάντηση/Λύση

Η ιδέα πίσω από το Last Responsible Moment είναι απλή. Αντί να κάνουμε όλα όσα χρειάζεται να γίνουν ταυτόχρονα ή προκαταβολικά, αναπτύσσουμε τον κώδικα στο χρονικό σημείο που χρειάζεται να γίνει, είτε στο σημείο στο οποίο θα ήταν ανεύθυνο να μην γίνει.

Υπάρχουν τρία σημαντικά οφέλη από τη στρατηγική αυτή:

- Το πρώτο είναι ότι ξοδεύουμε λιγότερο χρόνο υλοποιώντας τις απαιτήσεις, διότι αντί να ξοδεύουμε χρόνο για να τελειοποιήσουμε ένα χαρακτηριστικό και να καλύψουμε κάθε δυνατή περίπτωση, χρησιμοποιούμε τον ίδιο χρόνο για να κάνουμε περισσότερα πράγματα. Στη συνέχεια, μπορούμε να τελειοποιήσουμε μια απαίτηση, όταν αυτή γίνει σημαντική.
- Το δεύτερο είναι ότι πολλές φορές αυτά που στην αρχή φαίνονται χρήσιμα και απαραίτητα τελικά δεν είναι και τόσο σημαντικά και η Last Responsible Moment δεν έρχεται ποτέ. Όλοι οι προγραμματιστές στην καριέρα τους έχουν υλοποιήσει λειτουργίες που δεν χρησιμοποιήθηκαν ποτέ. Με τη λογική να αποφασίσουμε και να υλοποιούμε την Last Responsible Moment μειώνεται η πιθανότητα να φτιάξουμε κάτι άχρηστο, επειδή ο χρόνος μεταξύ ανάπτυξης του λογισμικού και της αποδέσμευσης αυτού είναι πολύ λιγότερος.
- Το τρίτο πλεονέκτημα είναι ότι επειδή η πληροφορική είναι ένας ταχέως εξελισσόμενος κλάδος, είναι πιθανό ότι, αν καθυστερήσουμε να κάνουμε κάτι, μπορεί να προκύψει καλύτερος τρόπος υλοποίησης.

Κριτήριο αξιολόγησης 7

Ο Γιάννης έχει ένα μικρό κατάστημα. Θέλει να γνωρίζει τον μέσο αριθμό πελατών που βρίσκονται σε ουρά στο κατάστημά του, για να αποφασίσει αν χρειάζεται να προσθέσει περισσότερο χώρο για να μπορεί να εξυπηρετεί περισσότερους πελάτες. Επί του παρόντος, η περιοχή αναμονής του δεν μπορεί να φιλοξενήσει περισσότερα από οκτώ άτομα.

Απάντηση/Λύση

Ο Γιάννης μέτρησε ότι κάθε ώρα φτάνουν στο κατάστημά του 20 πελάτες κατά μέσο όρο. Προσδιόρισε επίσης ότι, κατά μέσο όρο, ένας πελάτης μένει στο κατάστημά του περίπου 6 λεπτά (ή 0,1 ώρες). Με βάση αυτά τα δεδομένα, ο Γιάννης, εφαρμόζοντας τον νόμο του Little, μπορεί να βρει τον μέσο αριθμό πελατών που βρίσκονται σε αναμονή στο κατάστημά του.

L = 20 x 0,1 = 2 πελάτες

Ο νόμος του Little δείχνει ότι, κατά μέσο όρο, υπάρχουν μόνο δύο πελάτες που περιμένουν στο κατάστημα. Επομένως, δεν χρειάζεται να δημιουργήσει περισσότερο χώρο στο κατάστημα για να εξυπηρετήσει καλύτερα τους πελάτες του με δεδομένο ότι στην παρούσα φάση υπάρχει χώρος για την αναμονή οκτώ πελατών.

Κριτήριο αξιολόγησης 8

Ποιο είναι το κόστος της καθυστέρησης για το παράδειγμα με τα τρία έργα της παραγράφου 5.6.2 όταν γίνει χρήση του CD3.

Απάντηση/Λύση

Εάν δώσουμε προτεραιότητα με βάση την τιμή CD3, το CoD για τις πρώτες τέσσερις εβδομάδες είναι 120.000 €, για τις επόμενες οκτώ θα είναι 600.000 € και για τις τελευταίες δύο θα είναι 70.000 €. Το συνολικό κόστος καθυστέρησης θα είναι 790.000 €.

	Εβδομάδες	Εβδομαδιαίο κέρδος	Κόστος καθυστέρησης (CoD)
Project 2	4	30.000	4x30.000=**120.000**
Project 3	8	50.000	(8+4)x50.000=**600.000**
Project 1	2	5.000	(2+8+4)x5.000=**70.000**
		Σύνολο	**790.000**

Κριτήριο αξιολόγησης 9

Ποιο είναι το κόστος της καθυστέρησης για το παράδειγμα με τα τρία έργα της παραγράφου 5.6.2 όταν τα έργα εκτελεστούν παράλληλα

Απάντηση/Λύση

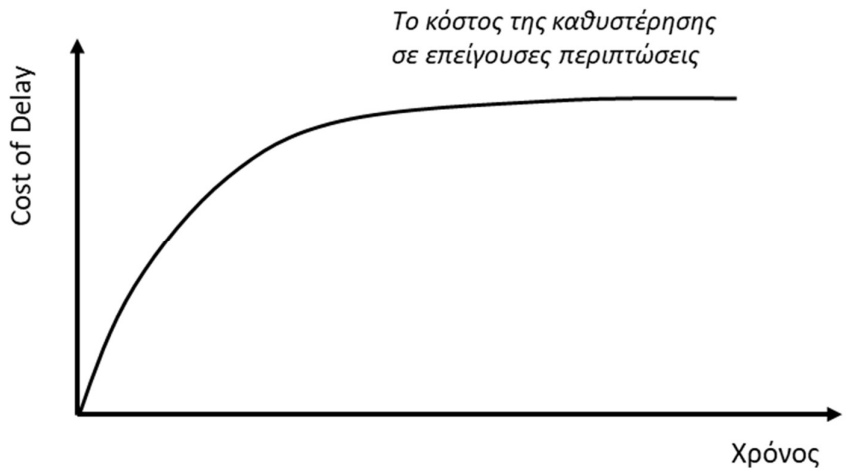
Εάν αποφασίσουμε να υλοποιήσουμε όλα τα έργα ταυτόχρονα, τότε θα ολοκληρωθούν και ταυτόχρονα την $14^{η}$ εβδομάδα. Δηλαδή θα χρειαστεί να περάσουν 14 εβδομάδες, πριν η επιχείρηση μπορέσει να αποκτήσει οποιαδήποτε επιχειρηματική αξία. Επομένως, το κόστος καθυστέρησης θα είναι 14 εβδομάδες πολλαπλασιασμένο με το άθροισμα του εβδομαδιαίου κέρδους και των 3 έργων που είναι 85.000€. Αυτό σημαίνει ότι 85.000x14=1.190.000€. Προφανώς αυτή είναι και η χειρότερη δυνατή επιλογή.

Κριτήριο αξιολόγησης 10

Δώστε ένα παράδειγμα όπου το κόστος καθυστέρησης υπαγορεύεται από τον επείγοντα χαρακτήρα του έργου.

Απάντηση/Λύση

Το κόστος καθυστέρησης που υπαγορεύεται από τον επείγοντα χαρακτήρα ενός έργου αυξάνεται γρήγορα σε πολύ σύντομο χρονικό διάστημα και στη συνέχεια παραμένει σχετικά σταθερό (βλέπε Εικόνα). Ένα καλό παράδειγμα τέτοιου τύπου CoD είναι όταν γνωρίζουμε ότι ο ανταγωνιστής αναπτύσσει ένα πρωτοποριακό χαρακτηριστικό/προϊόν που θα επηρεάσει δραστικά τη θέση του στην αγορά. Στην περίπτωση αυτή θα πρέπει επειγόντως να αναπτύξουμε αντίστοιχα ή και καλύτερα χαρακτηριστικά στο σχετικό προϊόν.



Κριτήριο αξιολόγησης 11

Έστω ο παρακάτω Kanban πίνακας (Ashmore & Runyan, 2014):

	Κατάσταση			
Το do	**Έτοιμες για ανάπτυξη**	**Σε εξέλιξη**	**Για έλεγχο**	**Ολοκληρώθηκαν**
Task M Task N Task O Task P Task Q Task R	Task H Task I Task J Task K Task L	Task D Task E Task F Task G	Task A Task B Task C	

Επίσης έχουν οριστεί τα παρακάτω WIP limits

Το do: Χωρίς όριο

Έτοιμες για ανάπτυξη: WIP limit = 5

Σε εξέλιξη: WIP limit = 4

Για έλεγχο: WIP limit = 3

Ολοκληρώθηκαν: Χωρίς όριο.

Αναφέρετε μερικά από τα προβλήματα που μπορούν να δημιουργηθούν με βάση τον παραπάνω πίνακα και τα δοθέντα όρια.

Απάντηση/Λύση

Έχοντας αυτά τα συγκεκριμένα WIP όρια, μπορούμε να εντοπίσουμε πού βρίσκονται τα προβληματικά σημεία. Στον παραπάνω πίνακα όλες οι εργασίες βρίσκονται στο ανώτερο όριο. Δηλαδή έχουμε έτοιμες για ανάπτυξη πέντε εργασίες, σε εξέλιξη τέσσερις εργασίες και για έλεγχο τρεις εργασίες.

Εάν ο προγραμματιστής ολοκληρώσει την εργασία D, δεν μπορεί να τη μεταφέρει στην στήλη για έλεγχο επειδή η ουρά του είναι ήδη γεμάτη. Πρέπει να περιμένει έως ότου ολοκληρωθεί μια από τις εργασίες A, B ή C, που είναι στη στήλη του ελέγχου, και να μεταφερθεί στη στήλη με τις ολοκληρωμένες εργασίες, προτού η εργασία D μπορεί να μπει στην ουρά/στήλη του ελέγχου. Αν γινόταν η μετακίνηση της εργασίας D στη στήλη ελέγχου θα παραβιαζόταν το όριο του ελέγχου (WIP= 3).

Επομένως κακή επιλογή WIP limits μπορεί να προκαλέσει σοβαρά προβλήματα και καθυστερήσεις.

Κριτήριο αξιολόγησης 12

Εξηγήστε τη διαφορά μεταξύ αντιληπτής (perceived integrity) και εννοιολογικής ακεραιότητας (conceptual integrity) και δώστε παραδείγματα.

Απάντηση/Λύση

Σύμφωνα με τον Kim Clark η αντιληπτή ακεραιότητα (perceived integrity) αναφέρεται στην ισορροπία που επιτυγχάνει ένα προϊόν μεταξύ της ομαλής λειτουργίας, της χρηστικότητας, της αξιοπιστίας και της οικονομίας που οδηγεί στην ικανοποίηση του πελάτη.

Η εννοιολογική ακεραιότητα είναι το γεγονός ότι οι κεντρικές έννοιες/συστατικά του συστήματος συνεργάζονται με ομαλό και συνεκτικό τρόπο.

Για παράδειγμα, ο μηχανισμός αναζήτησης της Google. Η μηχανή αναζήτησης της Google είναι γρήγορη, ο τρόπος εμφάνισης των αποτελεσμάτων είναι απλός, υπάρχει δυνατότητα μετάφρασης, δεν χρειάζεται κάποιος να γράψει τα πάντα σωστά, αφού η Google εντοπίζει τυπογραφικά λάθη και προτείνει ευγενικά μια διόρθωση, κ.λπ. Χιλιάδες χρήστες νιώθουν ικανοποιημένοι με τη μηχανή αναζήτησης της Google και συνεπώς η εφαρμογή αυτή έχει μεγάλο βαθμό αντιληπτής ακεραιότητας.

Αντίστοιχα, ένα σύστημα με υψηλή εννοιολογική ακεραιότητα, μπορεί να είναι ένα σύστημα ανανέωσης τηλεφωνικού χρόνου για κινητό τηλέφωνο, που επιτρέπει κάποιος χρήστης να πληρώσει με πολλούς διαφορετικούς τρόπους, με απρόσκοπτο τρόπο. Είτε επιθυμεί να πληρώσει με κάρτα Mastecard, είτε με VISA, είτε με την υπηρεσία paypal τα βήματα είναι παρόμοια.

Η προσέγγιση DevOps

An organized system of machines,
to which motion is communicated by a transmitting mechanism
from a central automation,
is the most developed form of production by machinery

Karl Marx

Σύνοψη
Η ανάγκη για συνεχή ροή λογισμικού έχει κάνει την προσέγγιση DevOps μια από τις πιο δημοφιλείς μεθόδους. Το έκτο κεφάλαιο παρουσιάζει την προσέγγιση DevOps, τις βασικές έννοιες, τον κύκλο ζωής και τις πρακτικές. Επίσης δίνεται έμφαση στην κουλτούρα DevOps και πως η προσέγγιση αυτή εντάσσεται στις σύγχρονες επιχειρήσεις. Στη συνέχεια παρουσιάζονται με πρακτικό τρόπο το πώς τα εργαλεία μπορούν να βοηθήσουν στην αυτοματοποίηση της προσέγγισης DevOps.

6 Η προσέγγιση DevOps

Θα αναρωτηθεί κανείς ποια είναι η σχέση ενός αρχαίου ρωμαϊκού υδραγωγείου με την προσέγγιση DevOps. Οι αρχαίες πόλεις χτίζονταν σε μέρη όπου υπήρχε άφθονο νερό, όπως και η Ρώμη. Το νερό το προμήθευε ο ποταμός Τίβερης καθώς και διάφορες πηγές και πηγάδια της περιοχής. Από τον τέταρτο αιώνα π.Χ., όμως, η Ρώμη άρχισε να επεκτείνεται με ταχύτατους ρυθμούς, γεγονός που αύξησε κατά πολύ και τις ανάγκες για νερό. Κατά τον 3ο αιώνα μ.Χ. η Ρώμη είχε 11 υδραγωγεία, πολλά από τα οποία αποτελούν θαύματα της μηχανικής αλλά και έργα μεγάλης κλίμακας. Για παράδειγμα, το Κλαυδιανό Υδραγωγείο, που διασώζεται μερικώς, είχε μήκος 69 χιλιόμετρα, από τα οποία τα 10 χιλιόμετρα αποτελούνταν από αψίδες, οι οποίες σε αρκετές περιπτώσεις έφταναν στα 27 μέτρα ύψος! Τα υδραγωγεία ήταν λοιπόν αυτά που επέτρεψαν την ανάπτυξη της πόλης και βελτίωσαν σημαντικά την υγεία των κατοίκων καθώς το νερό ήταν καθαρό (https://en.wikipedia.org/wiki/Roman_aqueduct). Οι Ρωμαίοι λοιπόν ήταν αυτοί που πέτυχαν τη **συνεχή παροχή ύδατος σε μεγάλη κλίμακα**.

Εικόνα 6.1: *Το ρωμαϊκό υδραγωγείο Pont Du Gard, Nimes Πηγή: https://commons.wikimedia.org*

Αντίστοιχα, στα τέλη του 19ου και στις αρχές του 20ου αιώνα αναπτύχθηκαν τα δίκτυα παραγωγής και μεταφοράς ηλεκτρικής ενέργειας τα οποία επέτρεπαν τη **συνεχή παροχή ηλεκτρικού ρεύματος** σε ολόκληρες πόλεις (https://en.wikipedia.org/wiki/History_of_electric_power_transmission).

Σήμερα, στις αρχές του 21ου αιώνα, οι έννοιες της συνεχούς ανάπτυξης λογισμικού αλλά και της συνεχούς παράδοσης αυτού κερδίζουν συνεχώς έδαφος. Η συνεχής παράδοση λογισμικού (Continuous Delivery - CD) είναι μια σχετικά νέα προσέγγιση ανάπτυξης λογισμικού και οι επιχειρήσεις που την έχουν υιοθετήσει αποκομίζουν σημαντικά οφέλη. Αντίστοιχα, οι χρήστες λογισμικού μετατρέπονται όλο και περισσότερο σε καταναλωτές λογισμικού, αφού το λογισμικό είναι διαθέσιμο αυτόματα, απαιτεί στη χρήση του ελάχιστες τεχνικές γνώσεις επιτρέποντάς τους να εστιαστούν αποκλειστικά στη χρήση του. Μπορούμε λοιπόν με σχετική βεβαιότητα να πούμε ότι ένα από τα βασικά χαρακτηριστικά του 21ου αιώνα θα είναι η **συνεχής παράδοση του λογισμικού.**

Παρ' όλα αυτά, οι μηχανισμοί και οι διαδικασίες που επιτρέπουν τη συνεχή παράδοση λογισμικού με χαμηλό ρίσκο δεν έχουν γίνει μέρος της καθημερινής πρακτικής στις περισσότερες επιχειρήσεις και αντίστοιχα στα έργα ανάπτυξης λογισμικού.

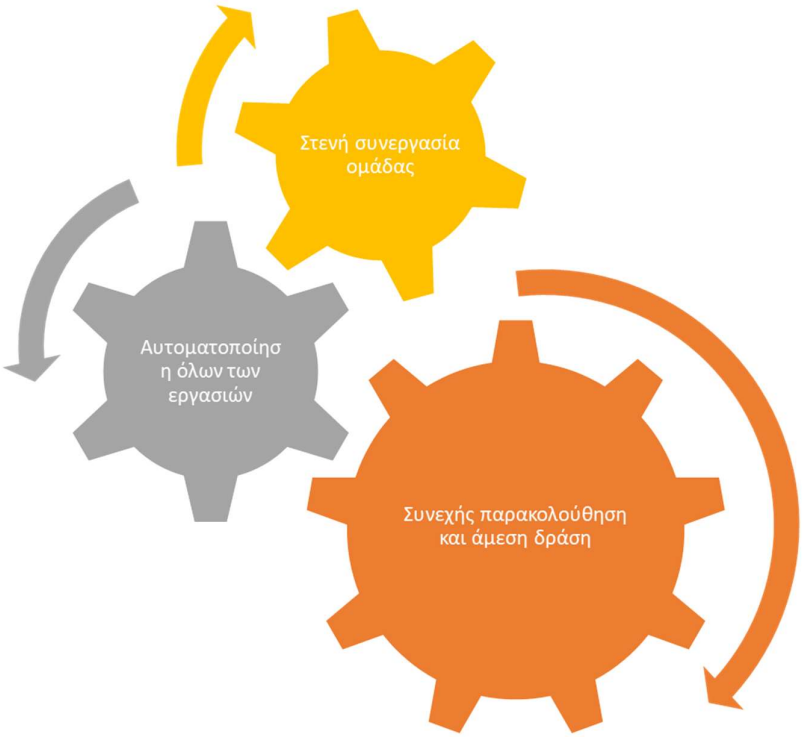
Στόχος μας λοιπόν θα πρέπει να είναι να κάνουμε τη συνεχή παράδοση λογισμικού μια αξιόπιστη, προβλέψιμη, διαφανή και σε μεγάλο βαθμό αυτοματοποιημένη διαδικασία με καλά κατανοητούς, και μετρήσιμους κινδύνους. Ταυτόχρονα, για το μέσο έργο λογισμικού ο χρόνος κύκλου θα πρέπει να μειωθεί σε ώρες ή ακόμη και σε λεπτά.

Είναι γνωστό ότι το μεγαλύτερο μέρος του κόστους της ανάπτυξης λογισμικού λαμβάνει χώρα μετά την παράδοση και την πρώτη αποδέσμευση αυτού. Το κόστος μετά την πρώτη αποδέσμευση αφορά το κόστος υποστήριξης, συντήρησης, προσθήκης νέων λειτουργιών και επιδιόρθωσης των σφαλμάτων. Αυτό ισχύει ιδιαίτερα όταν το λογισμικό έχει αναπτυχθεί με ευέλικτες επαναληπτικές μεθόδους, όπου η πρώτη έκδοση περιέχει μικρό μέρος της λειτουργικότητας. Άλλωστε αυτό αναφέρει και η πρώτη αρχή του Agile Manifesto: «Βασική προτεραιότητα αποτελεί η ικανοποίηση του πελάτη με τη συνεχή παράδοση σημαντικού τμήματος του λογισμικού από τα πρώτα κιόλας στάδια της παραγωγής (επικέντρωση στον πελάτη)». Αυτό αντανακλά την πραγματικότητα, η οποία είναι ότι η πρώτη έκδοση είναι μόνο η αρχή της συνεχούς διαδικασίας παράδοσης και του τελικού προϊόντος (Humble & Farley, 2010).

6.1 Μια σύντομη εισαγωγή στην προσέγγιση DevOps

Το DevOps είναι το ακρωνύμιο που προκύπτει από το συνδυασμό της ανάπτυξης λογισμικού (Development – Dev) και της λειτουργίας της ψηφιακής υπηρεσίας (Operations – Ops). Αναφέρεται σε μια συλλογική προσέγγιση σύμφωνα με την οποία η ομάδα ανάπτυξης λογισμικού και η ομάδα λειτουργίας της ψηφιακής υπηρεσίας μιας επιχείρησης, εργάζονται μαζί για την επίτευξη του τελικού αποτελέσματος και της μέγιστης επιχειρηματικής αξίας. Αν κανείς ήθελε να περιγράψει τις τρεις βασικές συνιστώσες της προσέγγισης DevOps αυτές θα ήταν (βλέπε Εικόνα 6.2):

- Στενή συνεργασία ομάδας.
- Αυτοματοποίηση όλων των εργασιών ανάπτυξης και λειτουργίας της ψηφιακής υπηρεσίας.
- Συνεχής παρακολούθηση και άμεση δράση.

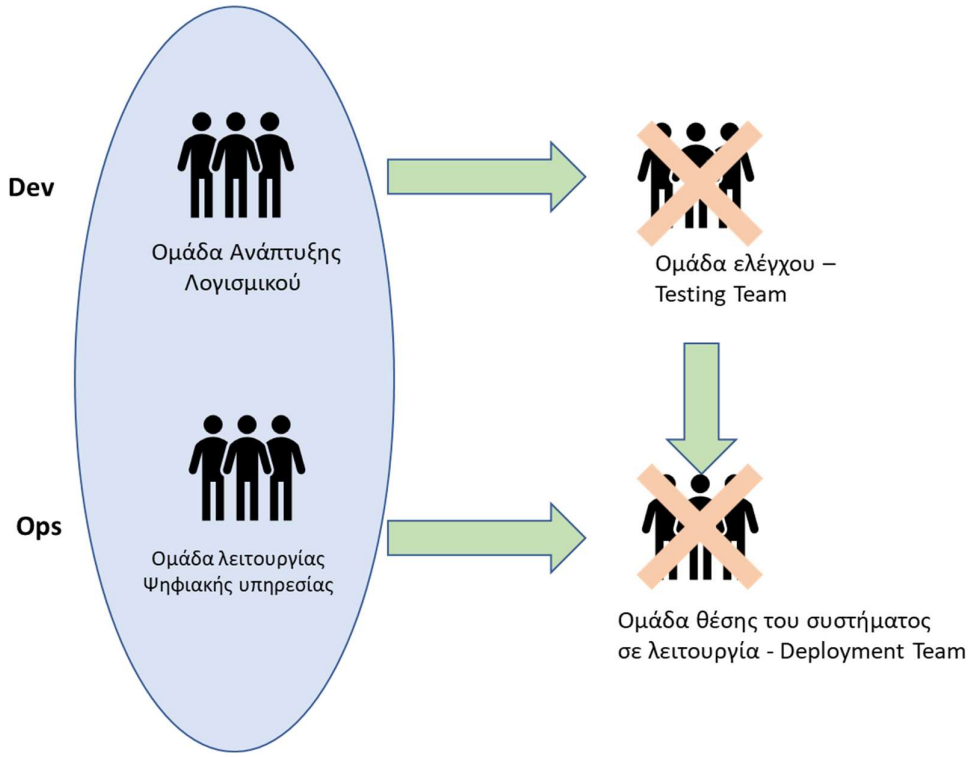


Εικόνα 6.2: *Οι βασικές συνιστώσες της προσέγγισης DevOps*

Το DevOps ορίζεται ως "ένα σύνολο πρακτικών που αποσκοπούν στη μείωση του χρόνου μεταξύ της πρότασης για μια αλλαγή σε ένα πληροφοριακό σύστημα και της υλοποίησης αυτής μέσω της τοποθέτησης της νέας έκδοσης του συστήματος σε κανονική λειτουργία, ενώ ταυτόχρονα εξασφαλίζουμε υψηλή ποιότητα της ψηφιακής υπηρεσίας" https://en.wikipedia.org/wiki/DevOps. Το DevOps είναι ένα σύνολο βέλτιστων πρακτικών που συνδυάζει τις δύο κύριες φάσεις του κύκλου ζωής των συστημάτων λογισμικού: την ανάπτυξη λογισμικού (Dev) και τις λειτουργίες συστήματος λογισμικού (Ops). Πρόκειται για μια σύγχρονη προσέγγιση που στοχεύει στη συντόμευση του κύκλου ζωής ανάπτυξης συστημάτων και στη συνεχή παροχή υψηλής ποιότητας λογισμικού (Humble & Kim, 2018).

Ειδικά σήμερα, η προσέγγιση DevOps είναι απολύτως απαραίτητη, δεδομένου ότι τα συστήματα λογισμικού είναι ζωτικής σημασίας για τη λειτουργία όλων των οργανισμών και ιδιαίτερα των πολύπλοκων συστημάτων, πρέπει να λειτουργούν με 24 ώρες το 24ωρο, 7 ημέρες την εβδομάδα, με πιεστικές και μεταβαλλόμενες ανάγκες. Η προσέγγιση DevOps εξαλείφει τα κενά μεταξύ της ομάδας ανάπτυξης και της ομάδας λειτουργίας που παρέχει την ψηφιακή υπηρεσία, δημιουργώντας μια ενιαία ομάδα.

Το DevOps δίνει έμφαση στην οικοδόμηση εμπιστοσύνης και στην καλύτερη σχέση μεταξύ των προγραμματιστών που αναπτύσσουν το λογισμικό και των διαχειριστών του συστήματος. Αυτό βοηθά σημαντικά την επιχείρηση να ευθυγραμμίσει τις τεχνικές με τις επιχειρηματικές απαιτήσεις (βλέπε Εικόνα 6.3) (Freeman, 2019).



Εικόνα 6.3: *Η ενοποίηση της ομάδας ανάπτυξης με τη ομάδα παροχής της ψηφιακής υπηρεσίας*

Οι δύο αυτές ομάδες έχουν διαφορετική στρατηγικά προσέγγιση αφού η ομάδα ανάπτυξης λογισμικού είναι συνήθως μια καινοτόμα ομάδα που εστιάζεται στην αλλαγή και στη δημιουργία κάτι καινούργιου, σε αντίθεση με την ομάδα λειτουργίας της ψηφιακής υπηρεσίας, η οποία είναι προσανατολισμένη στη σταθερότητα, την αδιάλειπτη λειτουργία, την επαναληψιμότητα, η οποία απαιτείται για τη λειτουργία μιας υπηρεσίας. Πιο συγκεκριμένα σε μια ομάδα ανάπτυξης, η ομάδα έχει ως βασικά καθήκοντα (Fitsilis, 2021):

- να εργάζεται με πελάτες που ζητούν αλλαγές,
- να υλοποιεί τις προτεινόμενες αλλαγές,
- να αναπτύσσει νέες δυνατότητες και λειτουργίες,
- να μπορεί να ενσωματώσει την ανατροφοδότηση από χρήστες σε μελλοντικές παραδόσεις.

Αντίθετα, μια ομάδα λειτουργίας/παροχής μιας ψηφιακής υπηρεσίας έχει ως βασικές εργασίες:

- την εγκατάσταση νέου λογισμικού,
- τη συνεργασία με πελάτες που ζητούν ένα περιβάλλον ομαλής λειτουργίας,
- την απρόσκοπτη εξυπηρέτηση πελατών,
- την παρακολούθηση των προβλημάτων, τις αποτυχίες και τις διακοπές λειτουργίας του συστήματος, και
- την παροχή μια σταθερής και καλά λειτουργούσας υπηρεσίας.

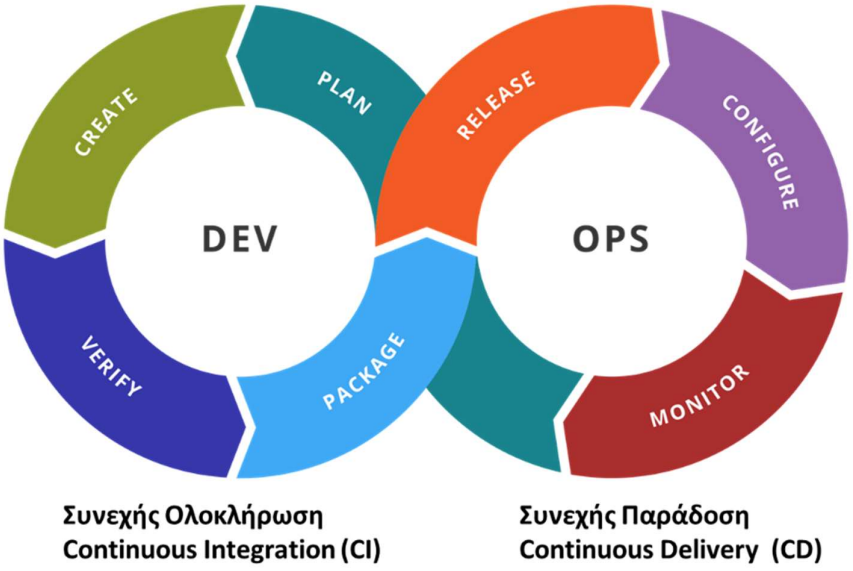
Συνεπώς, στην προσέγγιση DevOps πρέπει να επιτύχουμε μια νέα ισορροπία, η οποία προϋποθέτει αλλαγές τόσο στην ομάδα ανάπτυξης όσο και στην ομάδα λειτουργίας. Η ομάδα ανάπτυξης θα πρέπει να κατανοήσει τη φύση των συστημάτων παραγωγής, στα οποία θα εκτελούνται οι εφαρμογές τους. Ποια είναι τα πρότυπα για τα συστήματα παραγωγής και πώς πρέπει να λειτουργούν οι εφαρμογές τους σε αυτά; Μέσα σε ποιους περιορισμούς λειτουργούν οι εφαρμογές; Επίσης, η ομάδα ανάπτυξης πρέπει να εμπλακεί περισσότερο στους ελέγχους και να είναι σίγουροι ότι ο κώδικάς είναι όχι μόνο χωρίς σφάλματα, αλλά και

ποια είναι η απόδοσή του στο παραγωγικό σύστημα. Τέλος θα πρέπει να αποκρυπτογραφήσουν πώς αλληλοεπιδρούν τα υποσυστήματα μεταξύ τους και πώς ένα υποσύστημα μπορεί να προκαλέσει επιβράδυνση ή ακόμα και κατάρρευση ενός άλλου.

Αντίστοιχα, η ομάδα λειτουργίας πρέπει να γνωρίζει ποιος κώδικας έρχεται προς εγκατάσταση και πώς μπορεί να επηρεάσει το πληροφοριακό σύστημα. Αυτό απαιτεί εμπλοκή και συνεργασία από νωρίς με την ομάδα ανάπτυξης από τη φάση της καταγραφής των απαιτήσεων. Επίσης, θα πρέπει να αυτοματοποιεί τον τρόπο διαχείρισης των συστημάτων τους, διότι μόνο μέσω της αυτοματοποίησης μπορεί να επιτευχθούν γρήγορες αλλαγές με σταθερότητα. Ο αυτοματισμός επιτρέπει όχι μόνο γρήγορες αλλαγές αλλά και γρήγορες επαναλήψεις, στην περίπτωση που δεν πάει καλά. Στην ιδανική περίπτωση, η ομάδα λειτουργίας είναι αυτή που απελευθερώνει τις νέες εκδόσεις του λογισμικού. Αυτό μπορεί να γίνει μόνο όταν η πληροφοριακή υποδομή είναι υπό διαχείριση (managed services) και σαφώς καταγεγραμμένη. Τέλος, η ομάδα λειτουργίας, σε συνεργασία με την ομάδα ανάπτυξης, θα πρέπει να είναι σε θέση να εντοπίζει άμεσα τις δυσλειτουργίες, ώστε να αναγνωρίζονται τα πιθανά προβλήματα (Sharma, 2017).

Η προσέγγιση DevOps απεικονίζεται συνήθως ως ένας ατέρμονος βρόγχος που περιλαμβάνει τα παρακάτω βήματα:

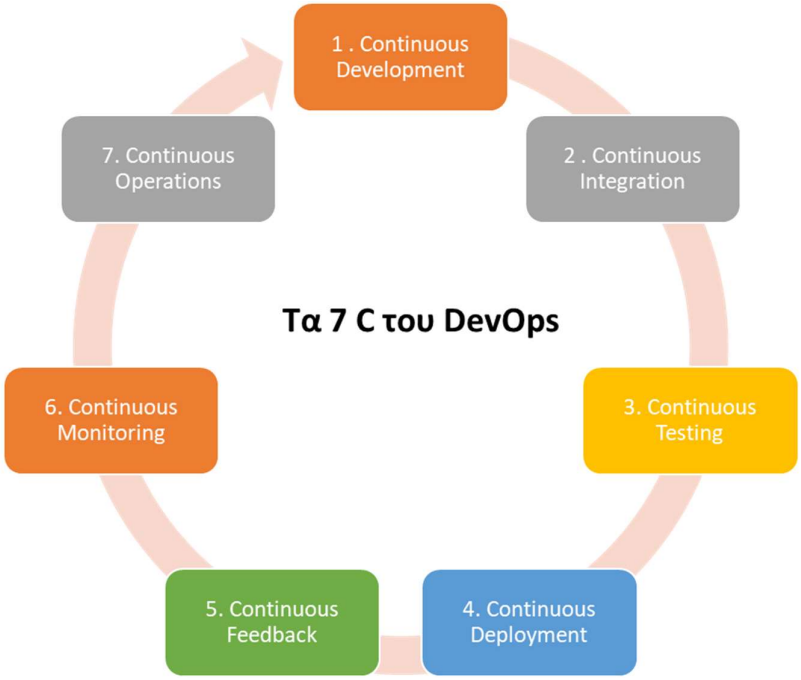
- Σχεδιασμός (Plan): Σε αυτό το στάδιο, οι ομάδες προσδιορίζουν τις επιχειρηματικές απαιτήσεις και συλλέγουν παρατηρήσεις από τους τελικούς χρήστες. Δημιουργούν έναν οδικό χάρτη του έργου με σκοπό να μεγιστοποιήσουν την επιχειρηματική αξία και να παραδώσουν το επιθυμητό προϊόν.
- Ανάπτυξη (Create): Σε αυτή τη φάση πραγματοποιείται η ανάπτυξη του κώδικα. Στο τέλος της ανάπτυξης ο κώδικας του συστήματος αποθηκεύεται στον κοινόχρηστο αποθετήριο κώδικα.
- Επαλήθευση (Verify): Μόλις το σύστημα λογισμικού είναι έτοιμο κι έχει ολοκληρωθεί σε ένα ενιαίο σύνολο, ελέγχεται πρώτα στο περιβάλλον δοκιμής (test environment) κάνοντας διάφορους τύπους δοκιμών, όπως δοκιμές αποδοχής χρήστη (User Acceptance Test -UAT), δοκιμές ασφαλείας (security test), δοκιμές ενοποίησης (integration testing), δοκιμές απόδοσης (performance testing) κ.λπ., χρησιμοποιώντας αυτοματοποιημένα εργαλεία ώστε να εξασφαλιστεί η ποιότητα του λογισμικού.
- Συσκευασία (Packaging): Η συσκευασία του κώδικα είναι ένα βασικό βήμα στον κύκλο ζωής ανάπτυξης λογισμικού DevOps. Είναι η διαδικασία λήψης πηγαίου κώδικα, εύρεσης τυχόν εξαρτήσεων που απαιτούνται και μετατροπής του σε κώδικα που μπορεί να εκτελεστεί στο παραγωγικό σύστημα.
- Αποδέσμευση (Release): Μόλις το λογισμικό έχει περάσει όλες τις δοκιμές, η ομάδα λειτουργίας προγραμματίζει την εγκατάσταση της έκδοσης ή των εκδόσεων στο παραγωγικό σύστημα με βάση την επείγουσα φύση της αποδέσμευσης ή γενικότερα ανάλογα με τις ανάγκες του οργανισμού.
- Διαχείριση διαμόρφωσης (Configure): Σε αυτό το στάδιο το λογισμικό είναι υπό τον έλεγχο του συστήματος διαμόρφωσης, ενώ έχουν εντοπιστεί τα στοιχεία διαμόρφωσης λογισμικού (Configurable Item - CI), οι διαμορφώσεις είναι αποθηκευμένες στο σχετικό εργαλείο, ο κώδικας είναι αποθηκευμένος στο αποθετήριο κώδικα καθώς και τα άλλα τεχνουργήματα (artifacts), κ.λπ.. Τεχνολογίες περιγραφής της πληροφοριακής υποδομής με κώδικα όπως Infrastructure-as-Code (IaC) βοηθούν σημαντικά στην αυτοματοποιημένη δημιουργία του περιβάλλοντος παραγωγής. Στη συνέχεια απελευθερώνεται το λογισμικό με τη βοήθεια σχετικών εργαλείων.
- Παρακολούθηση (Monitoring): Σε αυτό το στάδιο παρακολουθούμε τη λειτουργία του συστήματος συλλέγοντας δεδομένα, είτε σχετικά με τη συμπεριφορά των πελατών, είτε με την απόδοση της εφαρμογής κ.λπ. Η παρακολούθηση συνολικά του περιβάλλοντος λειτουργίας βοηθά τις ομάδες DevOps να βρουν τα σημεία που επηρεάζουν τόσο τη λειτουργία του συστήματος αλλά και αυτά που επηρεάζουν την παραγωγικότητα της ομάδας DevOps.



Εικόνα 6.4: *Ο κύκλος ζωής του DevOps*

6.2 Οι πρακτικές του DevOps

Όπως έχουμε ήδη αναφέρει, η προσέγγιση DevOps είναι ένας ατέρμονος βρόγχος, από τον σχεδιασμό έως την παρακολούθηση του λειτουργούντος συστήματος. Επομένως, εφόσον στόχος είναι η συνεχής αποδέσμευση νέων εκδόσεων σε όσο δυνατό μικρότερο χρόνο, το DevOps είναι μια συνεχής διεργασία. Ας αναλύσουμε λοιπόν ολόκληρο τον κύκλο ζωής σε επτά φάσεις ή όπως συνηθίζεται να λέγεται «τα 7C του DevOps" (βλέπε Εικόνα 6.5).



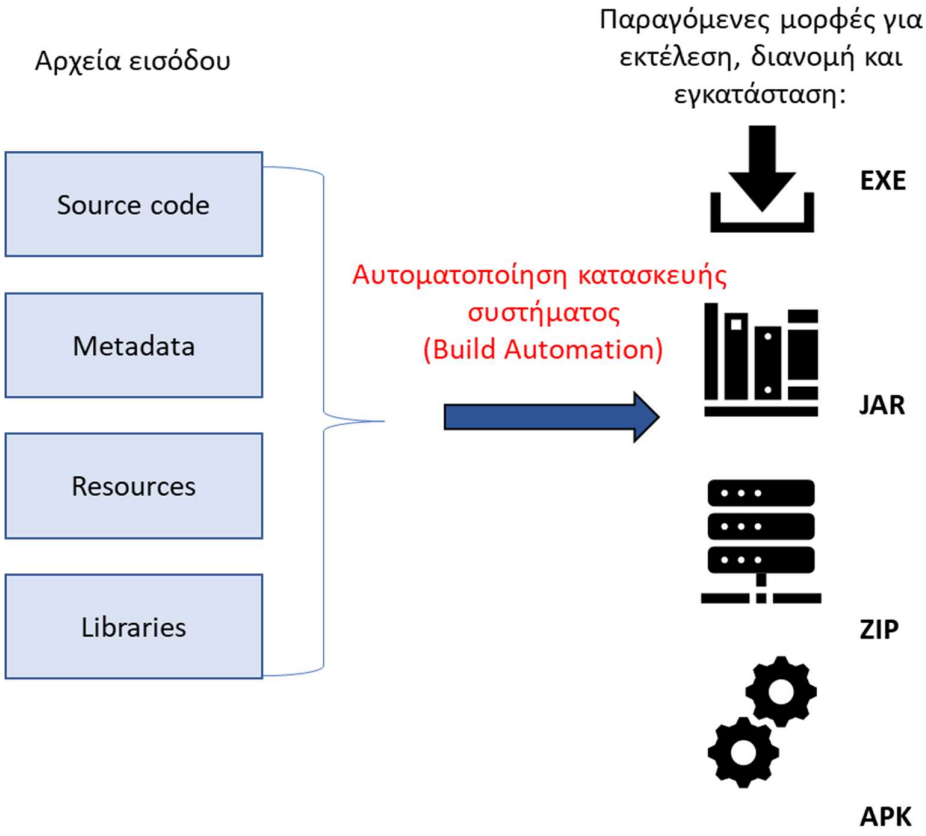
Εικόνα 6.5: *Τα 7C του DevOps*

6.2.1 Συνεχής Ανάπτυξη (Continuous Development)

Αυτή η φάση παίζει καθοριστικό ρόλο στην οριοθέτηση του οράματος για ολόκληρο τον κύκλο ανάπτυξης του λογισμικού του έργου. Επικεντρώνεται κυρίως στον προγραμματισμό (planning) και στην υλοποίηση του έργου. Κατά τη διάρκεια αυτής της φάσης, συγκεντρώνονται οι απαιτήσεις του έργου και συζητούνται με τους ενδιαφερόμενους/πελάτες. Επιπλέον, ενημερώνεται ο κατάλογος με τις ανεκτέλεστες απαιτήσεις του προϊόντος (product backlog), το οποίο εκλεπτύνεται όπου είναι απαραίτητο.

Μόλις η ομάδα ανάπτυξης συμφωνήσει στις επιχειρηματικές ανάγκες με τον ιδιοκτήτη προϊόντος ή γενικότερα με τους πελάτες, αρχίζει να κωδικοποιεί τις επιθυμητές απαιτήσεις. Είναι μια συνεχής διαδικασία όπου απαιτείται από τους προγραμματιστές να κωδικοποιούν κάθε φορά που συμβαίνουν αλλαγές στην απαιτήσεις του έργου ή για την επίλυση προβλημάτων γενικότερα.

Η συνεχής ανάπτυξη, εκτός από την ανάπτυξη κώδικα, περιλαμβάνει και την κατασκευή (build) του συστήματος. Στόχος είναι πάντα ο αυτοματισμός της κατασκευής με τη μικρότερη δυνατή παρέμβαση από τους προγραμματιστές (build automation). Η κατασκευή του συστήματος λαμβάνει ως είσοδο όλα τα αναγκαία αρχεία που συνήθως προέρχονται από το περιβάλλον ανάπτυξης (πηγαίος κώδικας, βιβλιοθήκες λογισμικού, κ.λπ.) και παράγει το αντίστοιχο εκτελέσιμο στην μορφή που απαιτείται (π.χ. exe, jar, zip) (βλέπε Εικόνα 6.6).



Εικόνα 6.6: *Αυτοματοποίηση κατασκευής συστήματος (build automation)*

Η κατασκευή του συστήματος απαιτεί την εύρεση όλων των συσχετίσεων του κώδικα, των εξαρτήσεων αυτού, αλλά και όλες τις αναγκαίες πληροφορίες που σχετίζονται με το έργο. Συνήθως, οι πληροφορίες περιλαμβάνουν, τα αρχεία διαμόρφωσης, τους εμπλεκόμενους προγραμματιστές καθώς και τους ρόλους τους, το χρησιμοποιούμενο σύστημα διαχείρισης σφαλμάτων, τις άδειες λογισμικού, τη διεύθυνση URL της τοποθεσίας του έργου, τις εξαρτήσεις του έργου και όλα τα άλλα μικρά κομμάτια (π.χ. γραφικά, βιβλιοθήκες, plugins) που είναι αναγκαία για να λειτουργήσει το σύστημα. Όλες αυτές οι πληροφορίες αποθηκεύονται στο σύστημα ανάπτυξης σε αρχεία ειδικού τύπου. Στο δημοφιλές σύστημα ανοικτού κώδικα Maven (https://maven.apache.org/), τα αρχεία αυτά είναι αρχεία XML (eXtended Markup Language) και ονομάζονται POM (Project Object Model) αρχεία (Varanasi & Belida, 2014; Siriwardena, 2015).

Τέλος, συνηθισμένες εργασίες που περιλαμβάνει η κατασκευή του συστήματος είναι:

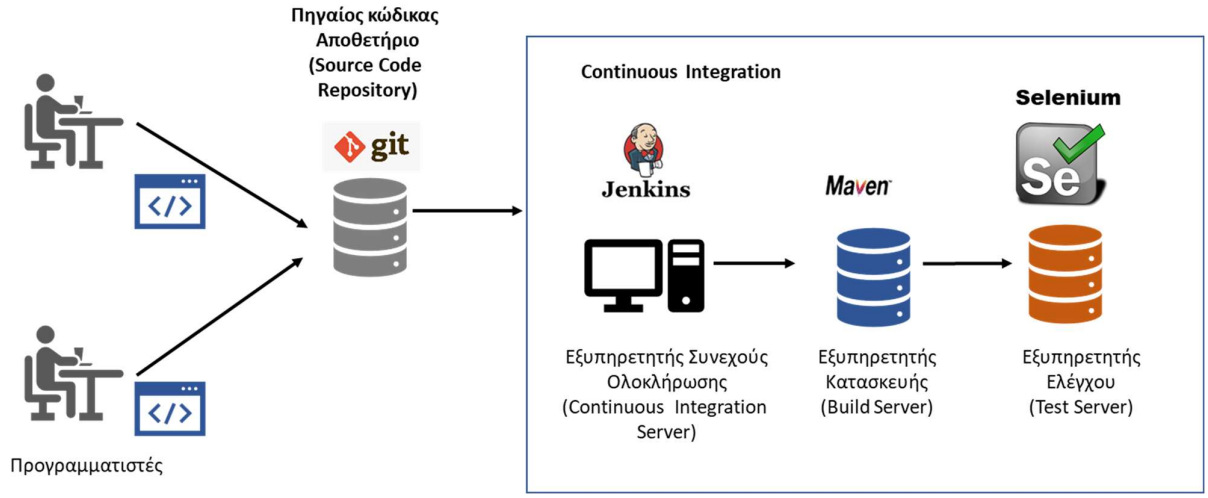
- Κύριες εργασίες
- Προεπεξεργασία – Preprocessing
- Μεταγλώττιση – Compilation
- Μεταεπεξεργασία – Postprocessing
- Συσκευασία - Packaging
- Συμπληρωματικές εργασίες
- Εκτέλεση δοκιμών
- Εγκατάσταση

6.2.2 Συνεχής Ενσωμάτωση (Continuous Integration)

Πολλές φορές, η ανάπτυξη μιας εφαρμογής λογισμικού και ιδιαίτερα ενός πληροφοριακού συστήματος γίνεται από πολλές ομάδες προγραμματιστών που εργάζονται σε ξεχωριστά συστατικά (components) του συστήματος. Επιπλέον, τις περισσότερες φορές, υπάρχει ανάγκη για αλληλεπίδραση με άλλα εξωτερικά συστήματα ή υπηρεσίες. Ορισμένες από αυτές τις εξωτερικές εφαρμογές ή υπηρεσίες μπορεί να είναι παλαιού τύπου εφαρμογές που υπάρχουν στην επιχείρηση από πολλά χρόνια (legacy applications) ή μπορεί να είναι εξωτερικές υπηρεσίες τρίτων. Υπάρχει επομένως, μια εγγενής ανάγκη για τους προγραμματιστές να ενσωματώσουν στο υπό ανάπτυξη σύστημα συστατικά που έχουν δημιουργηθεί από άλλες ομάδες ανάπτυξης ή να ολοκληρώνουν το σύστημα με άλλες εφαρμογές και υπηρεσίες. Αυτή η ανάγκη καθιστά τη διεργασία της ενσωμάτωσης μια ουσιαστική και πολύπλοκη εργασία στην ανάπτυξη λογισμικού.

Στις παραδοσιακές διαδικασίες ανάπτυξης λογισμικού και στο παρελθόν, η ολοκλήρωση ήταν ένα δευτερεύον σύνολο εργασιών που πραγματοποιείται μετά την κατασκευή των συστατικών ή του λογισμικού, ή ακόμη μερικές φορές στο τέλος της ανάπτυξης του πληροφοριακού συστήματος. Σύμφωνα με τον Fowler (Fowler, 2006), η συνεχής ολοκλήρωση είναι μια πρακτική ανάπτυξης λογισμικού όπου τα μέλη μιας ομάδας ενσωματώνουν την εργασία τους συχνά και, συνήθως κάθε μέλος της ομάδας ενσωματώνει τουλάχιστον καθημερινά την εργασία του, οδηγώντας σε πολλαπλές ενσωματώσεις ανά ημέρα. Κάθε ενσωμάτωση επαληθεύεται από ένα αυτοματοποιημένο build του συστήματος, το οποίο και ελέγχεται για τον εντοπισμό σφαλμάτων ενσωμάτωσης, το συντομότερο δυνατό.

Στην Εικόνα 6.7 παρουσιάζεται με γραφικό τρόπο η ροή εργασίας σε ένα οικοσύστημα συνεχούς ενσωμάτωσης με τη χρήση των αντίστοιχων εργαλείων.



Εικόνα 6.7: *Οικοσύστημα εργαλείων συνεχούς ενοποίησης*

Ο Fowler και ο Foemmel (2006) έχουν περιγράψει τη συνεχή ενσωμάτωση με δέκα καλές πρακτικές, οι οποίες είναι:

1. **Χρήση ενός αποθετηρίου (singe source repository).** Ο Fowler τονίζει ότι είναι σημαντικό να χρησιμοποιούμε εργαλεία διαχείρισης εκδόσεων (configuration management tools)[8], είτε έχουμε αποκλειστικά κώδικα είτε και άλλα τεχνουργήματα. Το αποθετήριο θα πρέπει να επιτρέπει την πρόσβαση πολλών χρηστών, τη διαχείριση εκδόσεων, τη διακλάδωση και τη συγχώνευση εκδόσεων, τη δημιουργία baseline κώδικα, κ.λπ.. Θα πρέπει με λίγα λόγια να δίνει πλήρη έλεγχο των αρχείων και να επιτρέπει σε πολλούς προγραμματιστές, από διαφορετικές τοποθεσίες, να εργάζονται στο ίδιο σύνολο αρχείων.
2. **Να αυτοματοποιεί την κατασκευή του συστήματος (build).** Η αυτοματοποίηση της κατασκευής είναι η λειτουργία που μας επιτρέπει να κάνουμε τη συνεχή ενσωμάτωση. Επιπλέον, θα πρέπει να είναι δυνατή, όταν απαιτείται, η κατασκευή του συστήματος για πολλές πλατφόρμες (π.χ. Windows, iOS, Android).
3. **Η δυνατότητα αυτοελέγχου του build**. Ακριβώς όπως το build πρέπει να είναι αυτοματοποιημένο, έτσι πρέπει να είναι αυτοματοποιημένες και οι δοκιμές, δηλαδή να υπάρχει δυνατότητα αυτοελέγχου. Στόχος της συνεχούς ολοκλήρωσης δεν είναι μόνο η ενσωμάτωση της εργασίας της ομάδας, αλλά και η διαπίστωση εάν η εφαρμογή ή το πληροφοριακό σύστημα που κατασκευάζεται λειτουργεί και αποδίδει όπως αναμένεται. Αυτό απαιτεί τη δημιουργία ενός συνόλου αυτοματοποιημένων σεναρίων δοκιμής (test scripts), για τους μοναδιαίους ελέγχους (unit tests), καθώς και για τον έλεγχο των συστατικών αλλά και της εφαρμογής. Αυτή η διαδικασία αυτοελέγχου απαιτεί ότι τα script δημιουργίας της νέας έκδοσης, περιλαμβάνουν τη δυνατότητα build του λογισμικού, εάν είναι αναγκαίο, την παροχή (provisioning) του διακομιστή ελέγχου (test server), την παροχή περιβάλλοντος ελέγχου, την εγκατάσταση του λογισμικού στον διακομιστή ελέγχου, τη ρύθμιση των δεδομένων δοκιμής και την εκτέλεση των κατάλληλων σεναρίων ελέγχου. Η απαίτηση για ύπαρξη περιβάλλοντος

[8] Η διαχείριση σχηματισμών είναι η διαδικασία καταγραφής και τεκμηρίωσης των αλλαγών του συστήματος με τη χρήση γραμμών αναφοράς (baselines), η οποία: α) παρακολουθεί και καταγράφει τα λειτουργικά και φυσικά χαρακτηριστικά ενός αντικειμένου και ενός συστήματος, καθώς αυτό εξελίσσεται μέσα στον χρόνο, β) ελέγχει και καταγράφει τις αλλαγές αυτών των χαρακτηριστικών μέσα στον χρόνο, γ) μπορεί να παρουσιάσει ανά πάσα στιγμή και με λεπτομέρειες την ιστορία εξέλιξης του συστήματος, καθώς και την παρούσα κατάσταση και δ) ελέγχει ότι όλα τα επιμέρους στοιχεία ενός συστήματος ακολουθούν τους κανόνες (Φιτσιλής, 2018).

ελέγχου συχνά είναι εφικτή μέσω χρήσης τεχνικών εικονοποίησης (testing virtualization). Επίσης η αυτοματοποίηση της δημιουργίας περιβάλλοντος ελέγχου είναι εφικτή λόγω της ύπαρξης δυνατοτήτων να περιγράψουμε την αναγκαία υποδομή με κώδικα/λογισμικό (Infrustructure as Code - IaC)[9].

4. **Όλα τα μέλη της ομάδας θα πρέπει να κάνουν commit**[10] στην κύρια γραμμή καθημερινά. Ο στόχος είναι ότι ο κάθε προγραμματιστής πρέπει καθημερινά να κάνει commit για όλα τα αρχεία και συστατικά που δουλεύει στην κύρια γραμμή ανάπτυξης του έργου ώστε να διασφαλιστεί ότι οι ενσωματώσεις παραμένουν όσο το δυνατόν απλούστερες. Πολλές φορές οι προγραμματιστές εργάζονται ανεξάρτητα στις αλλαγές του κώδικά τους σε τοπικές εκδόσεις αρχείων και κάνουν commit μόνο στην τελική έκδοση, οπότε συνειδητοποιούν ότι η εργασία τους επηρεάζεται από την εργασία άλλων προγραμματιστών. Αυτό μπορεί να οδηγήσει σε καθυστερήσεις στην απελευθέρωση του λογισμικού ή σε αλλαγές της τελευταίας στιγμής που δεν έχουν ελεγχθεί σωστά. Η τακτική ενσωμάτωση του κώδικα μπορεί να διασφαλίσει ότι αυτές οι εξαρτήσεις εντοπίζονται νωρίτερα, ώστε η ομάδα ανάπτυξης να μπορεί να τις αντιμετωπίσει έγκαιρα και χωρίς χρονικούς περιορισμούς.

5. Θα πρέπει να διασφαλίσουμε ότι **κάθε commit συμβάλλει σε μια βασική γραμμή**. Η διασφάλιση αυτή θα επιτρέψει την εκτέλεση αυτοματοποιημένων ελέγχων παλινδρόμησης (regression tests) και βοηθά στο να εντοπίζουμε και να επιλύουμε τα προβλήματα εγκαίρως. Ο έλεγχος παλινδρόμησης ορίζεται ως ένας τύπος δοκιμής λογισμικού με σκοπό την επιβεβαίωση ότι μια πρόσφατη αλλαγή στον κώδικα δεν επηρεάζει αρνητικά τις υπάρχουσες λειτουργίες.

6. Η κατασκευή του συστήματος (build) **πρέπει να γίνεται γρήγορα.** Ένα γρήγορο build επιτρέπει την άμεση και συνεχή ενσωμάτωση και αποτελεί προϋπόθεση ώστε το build να γίνεται συνεχώς.

7. **Δυνατότητα δοκιμής του λογισμικού σε κλώνο του περιβάλλοντος παραγωγής.** Η δοκιμή σε ένα περιβάλλον που δεν αντιπροσωπεύει με ακρίβεια το σύστημα παραγωγής δημιουργεί σημαντικό κίνδυνο αστοχίας. Στόχος αυτής της πρακτικής είναι η δοκιμή σε έναν κλώνο του περιβάλλοντος παραγωγής. Δεν είναι ωστόσο πάντα δυνατό να δημιουργηθεί ένας κλώνος ενός ολόκληρου περιβάλλοντος πολλαπλών διακομιστών μόνο για δοκιμή, είτε λόγω μεγέθους, είτε λόγω ύπαρξης άλλων λειτουργιών (λογισμικού) που τρέχουν παράλληλα στο σύστημα παραγωγής.

8. Εκτός και αν συντρέχουν άλλοι λόγοι, κάθε μέλος της ομάδας DevOps θα πρέπει να έχει **πρόσβαση στην τελευταία έκδοση του εκτελέσιμου λογισμικού**. Συνεπώς θα πρέπει να διευκολύνουμε οποιονδήποτε από την ομάδα να αποκτήσει το πιο πρόσφατο εκτελέσιμο αρχείο. Οποιοσδήποτε σχετίζεται με το έργο θα πρέπει να έχει πρόσβαση σε αυτό που έχει κατασκευαστεί και θα πρέπει να έχει έναν τρόπο αλληλεπίδρασης με αυτό. Αυτό επιτρέπει την επαλήθευση του αποτελέσματος σε σχέση με τις αρχικές προδιαγραφές από όλα τα μέλη της ομάδας DevOps.

9. Μεταξύ των μελών της ομάδας DevOps, θα πρέπει να υπάρχει **πλήρης διαφάνεια**. Θα πρέπει να εξασφαλίσουμε ότι όλα τα μέλη της ομάδας ενημερώνονται κατάλληλα και γνωρίζουν την

[9] Το Infrastructure as Code (IaC) (Rahman et al., 2019) είναι μια σύγχρονη προσέγγιση για την αυτοματοποιημένη διαχείριση της υποδομής διακομιστή δεδομένων, αποθήκευσης και δικτύωσης. Το IaC ως τεχνική απλοποιεί σημαντικά τη διαμόρφωση και τη διαχείριση πληροφοριακής υποδομής μεγάλης κλίμακας. Με τις παραδοσιακές τεχνικές διαχείρισης πληροφοριακής υποδομής, κάθε αλλαγή διαμόρφωσης απαιτεί την παρέμβαση του διαχειριστή του συστήματος. Με την τεχνική IaC, οι πληροφορίες διαμόρφωσης πληροφοριακής υποδομής περιγράφονται σε τυποποιημένα αρχεία, τα οποία μπορούν να εκτελεστούν από το λογισμικό που διαχειρίζεται την πληροφοριακή υποδομή. Το IaC μπορεί να βελτιώσει την παραγωγικότητα και την αξιοπιστία επειδή εξαλείφει τα βήματα μη αυτόματης διαμόρφωσης.

[10] Η εντολή – ενέργεια του "commit" χρησιμοποιείται για την αποθήκευση των αλλαγών στο αποθετήριο.

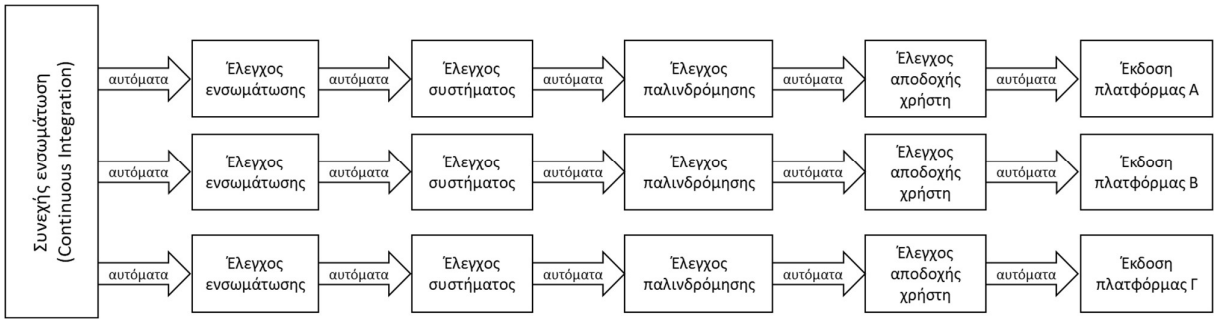
τρέχουσα κατάσταση. Αυτό είναι μια καλή πρακτική που σχετίζεται μάλλον με την επικοινωνία και τη συνεργασία, παρά με τη συνεχή ενσωμάτωση. Ωστόσο, η σημασία της για τις ομάδες που ασκούν συνεχή ενσωμάτωση είναι ιδιαίτερα σημαντική.

10. **Αυτοματοποίηση της διάταξης (deployment) του συστήματος.** Η συνεχής ολοκλήρωση οδηγεί στη συνεχή παράδοση του λογισμικού και τη γενικότερη αυτοματοποίηση της παραγωγής του λογισμικού.

6.2.3 Συνεχής Έλεγχος (Continuous Testing)

Ο συνεχής έλεγχος περιλαμβάνει τον ορισμό του περιβάλλοντος ελέγχου, και τις διαδικασίες εφαρμογής για κάθε στάδιο/βήμα της παράδοσης του λογισμικού. Τα αντικείμενα που ελέγχονται καθώς και τα είδη των ελέγχων που διεξάγονται μπορούν να αλλάξουν ανάλογα με το στάδιο της υλοποίησης. Στόχος μας είναι να αυτοματοποιήσουμε πλήρως τη διαδικασία ελέγχου ώστε να απαιτεί την ελάχιστη δυνατή ανθρώπινη παρέμβαση. Ο συνεχής έλεγχος είναι πραγματικά συνυφασμένος με τις διαδικασίες συνεχούς ενοποίησης και συνεχούς παράδοσης (Verona et al. 2016). Ο συνεχής έλεγχος επιτυγχάνεται δοκιμάζοντας όλες τις πτυχές της εφαρμογής και του περιβάλλοντος, συμπεριλαμβανομένων, μέσω πολλών και διαφορετικών ειδών ελέγχου, όπως ενδεικτικά παρουσιάζεται παρακάτω (Βλέπε Εικόνα 6.8):

- Μοναδιαίος έλεγχος (Unit testing)
- Λειτουργικός έλεγχος (Functional testing)
- Έλεγχος απόδοσης (Performance testing)
- Έλεγχος ενσωμάτωσης(Integration testing)
- Έλεγχος ενοποίησης συστήματος (System integration testing)
- Έλεγχος ασφαλείας (Security testing)
- Έλεγχος αποδοχής χρηστών (User acceptance testing)

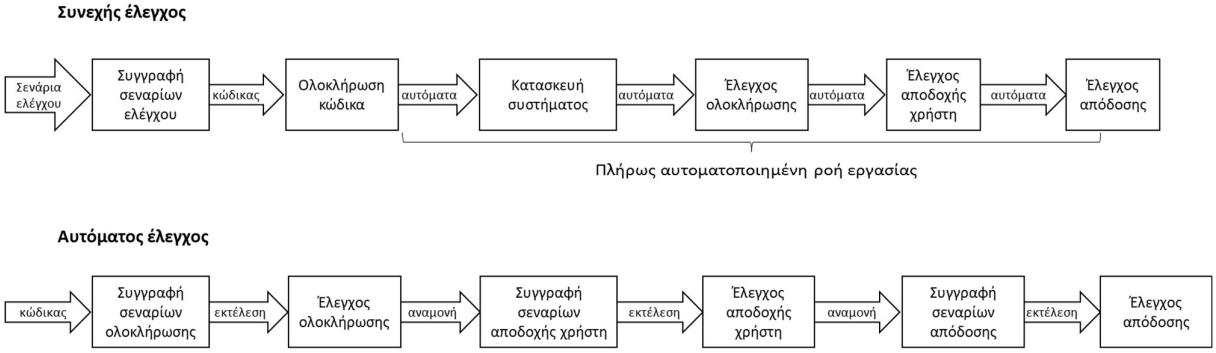


Εικόνα 6.8: *Η λογική του συνεχούς ελέγχου*

Στον συνεχή έλεγχο, η μεγαλύτερη πρόκληση είναι ότι ορισμένες από τις εφαρμογές, τις υπηρεσίες και τις πηγές δεδομένων που απαιτούνται για την εκτέλεση ορισμένων ελέγχων, ενδέχεται να μην είναι διαθέσιμες. Εναλλακτικά, ακόμα κι αν είναι διαθέσιμα, το κόστος που σχετίζεται με τη χρήση τους μπορεί να είναι απαγορευτικό για τη συνεχή εκτέλεση δοκιμών. Η λύση στο πρόβλημα αυτό είναι η πρακτική που είναι γνωστή ως εικονικοποίηση ελέγχου (testing virtualization). Σε αυτή τη στρατηγική ελέγχου, αντικαθιστούμε τις πραγματικές εφαρμογές, υπηρεσίες και πηγές δεδομένων με τις οποίες η εφαρμογή πρέπει να επικοινωνεί και να αλληλοεπιδρά κατά τη διάρκεια του ελέγχου, με εικονικές εφαρμογές. Αυτά τα εικονικά στιγμιότυπα καθιστούν δυνατό τον έλεγχο της λειτουργικότητας των εφαρμογών, της ενοποίησης και της απόδοσης αυτής χωρίς να είναι διαθέσιμο ολόκληρο το πληροφοριακό οικοσύστημα.

Πολλές φορές χρησιμοποιείται ο όρος αυτοματοποιημένος έλεγχος (automated testing), όμως οι έννοιες δεν είναι ταυτόσημες. Ο στόχος του συνεχούς ελέγχου είναι να αυτοματοποιήσει τον έλεγχο σε σταθερά περιβάλλοντα ελέγχου που μοιάζουν με αυτό της παραγωγής, ενώ του αυτοματοποιημένου ελέγχου

να ελέγξει ένα σύνολο ιστοριών χρηστών ή απαιτήσεων. Στην Εικόνα 6.9. παρουσιάζονται αναλυτικά οι δύο προσεγγίσεις.



Εικόνα 6.9: *Η διαφορά μεταξύ του συνεχούς ελέγχου και του αυτοματοποιημένου ελέγχου*

Οι κύριες διαφορές μεταξύ των συνεχούς ελέγχου και του αυτοματοποιημένου ελέγχου μπορούν να ομαδοποιηθούν σε τρεις μεγάλες κατηγορίες: αυτές που αφορούν τους κινδύνους, στο εύρος του ελέγχου και στο χρονοδιάγραμμα.

- Οι επιχειρήσεις σήμερα όχι μόνο διαθέτουν πολλές από τις εσωτερικές τους εφαρμογές στον τελικό χρήστη μέσω διαδικτύου, αλλά έχουν επίσης αναπτύξει μεγάλες ποσότητες πρόσθετου λογισμικού που επεκτείνει και συμπληρώνει αυτές τις εφαρμογές. Για παράδειγμα, οι αεροπορικές εταιρείες δίνουν τη δυνατότητα εκτός της αγοράς αεροπορικού εισιτηρίου, τη δυνατότητα ενοικίασης αυτοκινήτου, κ.λπ. Η έκθεση ολοένα και περισσότερων καινοτόμων λειτουργιών στον τελικό χρήστη είναι αυτή που προσφέρει ανταγωνιστική διαφοροποίηση επίσης και επιπλέον αυξάνει τον αριθμό, την ποικιλία και την πολυπλοκότητα των πιθανών σημείων αστοχίας. Συνεπώς, μεγάλης κλίμακας «αστοχίες λογισμικού» έχουν πολύ σοβαρές επιχειρηματικές επιπτώσεις και οι κίνδυνοι που σχετίζονται με τη λειτουργία του λογισμικού επηρεάζουν άμεσα την οικονομική κατάσταση της επιχείρησης. Στο πλαίσιο αυτό, ο συνεχής έλεγχος εστιάζεται στους επιχειρηματικούς κινδύνους ενώ ο αυτοματοποιημένος έλεγχος, σε αναλυτικούς, χαμηλού επιπέδου, ελέγχους.
- Ο συνεχής έλεγχος θα πρέπει να καλύπτει πλήρως μια επιχειρηματική διεργασία, ώστε να εξασφαλίζει ότι ικανοποιούνται οι προσδοκίες του χρήστη, ενώ ο αυτοματοποιημένος έλεγχος δεν εστιάζεται συνολικά στην επιχειρηματική διεργασία. Δηλαδή, αν και είναι σημαντικό να γνωρίζουμε ότι ένας μοναδιαίος έλεγχος απέτυχε ή ότι ένας έλεγχος διεπαφής χρήστη ολοκληρώθηκε σωστά, εξίσου σημαντικό είναι να γνωρίζουμε εάν και πώς η συνολική εμπειρία χρήστη επηρεάζεται από τις πρόσφατες αλλαγές της εφαρμογής.
- Η προσέγγιση DevOps επιτρέπει τη συνεχή παράδοση και εγκατάσταση του λογισμικού ακόμη και κάθε λεπτό της ώρας. Επομένως, ο συνεχής έλεγχος είναι αυτός που μας εξασφαλίζει ότι το σύστημα είναι ακέραιο και χωρίς προβλήματα. Ο αυτοματοποιημένος έλεγχος μπορεί μόνο να εξασφαλίσει ότι τα επιμέρους συστήματα λειτουργούν κανονικά. Αυτή η ταχύτητα συνεχούς παράδοσης, η οποία βασίζεται στο συνεχή έλεγχο, πολλές φορές αποτελεί για τις επιχειρήσεις ανταγωνιστικό πλεονέκτημα ή τους δίνει τη δυνατότητα για ανταγωνιστική διαφοροποίηση. Αυτός είναι και ο βασικός λόγος που η μεγάλη πλειοψηφία των οργανισμών στρέφεται στις ευέλικτες μεθόδους και στο DevOps για να επιταχύνουν τις διαδικασίες παράδοσης.

6.2.4 Η Συνεχής Διάταξη του Συστήματος (Continuous Deployment)

Η λέξη deploy προέρχεται από τη γαλλική λέξη deployer, που χρησιμοποιείται στη στρατιωτική ορολογία και σημαίνει την στρατηγική διάταξη – παράταξη ενός ή περισσότερων αντικειμένων. Στο πλαίσιο της πληροφορικής, η διάταξη ενός πληροφοριακού συστήματος περιλαμβάνει όλες τις διαδικασίες που

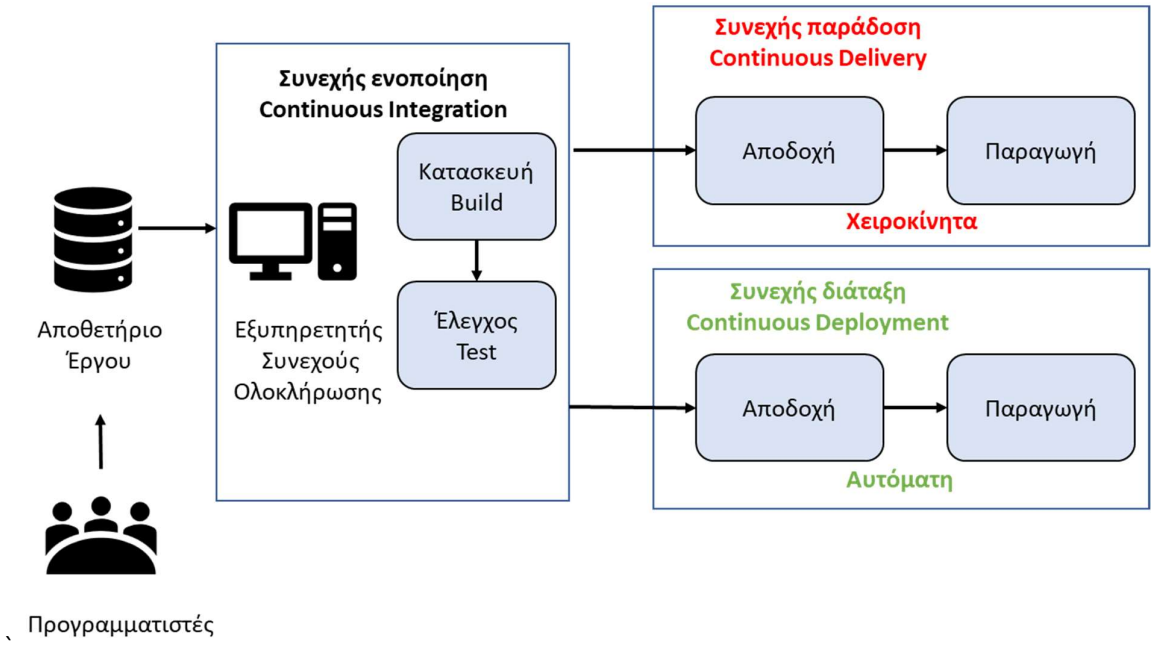
εμπλέκονται στη σωστή λειτουργία νέου λογισμικού ή υλικού στο περιβάλλον του, συμπεριλαμβανομένης της εγκατάστασης, της διαμόρφωσης, της εκτέλεσης, του ελέγχου και της πραγματοποίησης των απαραίτητων αλλαγών.

Η συνεχής διάταξη του συστήματος είναι λοιπόν η αυτόματη τοποθέτηση του πληροφοριακού συστήματος στο παραγωγικό σύστημα, με την προϋπόθεση ότι έχει περάσει επιτυχώς όλους τους αναγκαίους ελέγχους. Είναι μια πλήρως αυτοματοποιημένη διαδικασία που ξεκινά με το commit του κώδικα στο αποθετήριο και ολοκληρώνεται με την παραγωγική λειτουργία του συστήματος. Το εντυπωσιακό είναι ότι ιδανικά δεν θα πρέπει να υπάρχει καμία ανθρώπινη παρέμβαση.

Η αυτοματοποίηση διάταξης ενός συστήματος (deployment automation) αναφέρεται στα εργαλεία που επιτρέπουν την εγκατάσταση του λογισμικού στο περιβάλλον παραγωγής με αυτοματοποιημένο τρόπο. Η αυτοματοποίηση παράταξης απαιτεί τα παρακάτω δεδομένα εισόδου:

- Τα πακέτα λογισμικού που δημιουργούνται από τη διεργασία συνεχούς ενοποίησης (CI)
- Δέσμες ενεργειών για τη ρύθμιση παραμέτρων του περιβάλλοντος προορισμού (configuration scripts), την αποδέσμευση των πακέτων λογισμικού και την εκτέλεση των ελέγχων αποδέσμευσης (*smoke test*).
- Πληροφορίες ρύθμισης παραμέτρων για το συγκεκριμένο περιβάλλον.

Η συνεχής διάταξη ως έννοια, διαφέρει από αυτή της συνεχούς παράδοσης διότι η συνεχής διάταξη στη γενική περίπτωση απαιτεί πλήρη αυτοματοποίηση, ενώ η συνεχής παράδοση μπορεί να εμπεριέχει και βήματα που εκτελούνται χειροκίνητα από την ομάδα DevOps. Η διαφορά των δύο εννοιών παρουσιάζεται στην Εικόνα 6.10.



Εικόνα 6.10: *Η διαφορά μεταξύ του εννοιών συνεχούς παράδοσης και διάταξης*

6.2.5 Συνεχής ανατροφοδότηση (Continuous Feedback)

Αυτή η λογική της συνεχούς ανατροφοδότησης χρησιμοποιείται στρατηγικά για να διασφαλίσει ότι το υπό ανάπτυξη προϊόν βρίσκεται σε αρμονία με τις απαιτήσεις της αγοράς. Για το σκοπό αυτό η ομάδα DevOps μελετά το υπάρχον προϊόν για να βελτιώσει τη λειτουργία του, ώστε να είναι το καλύτερο δυνατό και να μεγιστοποιεί την αξία για τον πελάτη. Για το λόγο αυτό λαμβάνονται υπόψη τα σχόλια των πελατών σχετικά με το προϊόν και κάθε παρατήρηση/σχόλιο/λάθος αναλύεται και υλοποιείται το συντομότερο δυνατόν.

6.2.6 Συνεχής Παρακολούθηση (Continuous Monitoring)

Κατά την παραγωγή, η ομάδα DevOps διασφαλίζει ότι μια εφαρμογή λειτουργεί όπως είναι επιθυμητό και ότι το παραγωγικό περιβάλλον είναι σταθερό μέσω συνεχούς παρακολούθησης. Συνήθως, οι ομάδες DevOps έχουν τα δικά τους εργαλεία για να παρακολουθούν το παραγωγικό περιβάλλον είτε αφορά το πληροφοριακό σύστημα είτε την πληροφοριακή υποδομή. Τελικά, η ομάδα DevOps χρειάζεται να βεβαιωθεί ότι οι εφαρμογές αποδίδουν σε όλα τα επίπεδα.

Καθώς η τεχνολογία σε αυτόν τον χώρο έχει αναπτυχθεί, υπάρχουν επίσης εργαλεία και υπηρεσίες που παρακολουθούν τη συμπεριφορά των εφαρμογών σε συνδυασμό με το πώς αισθάνονται οι χρήστες για το σύστημα (sentiment analysis), παρέχοντας ακόμη πιο λεπτομερή ανατροφοδότηση. Με λίγα λόγια, η συνεχής παρακολούθηση απαιτεί τη συλλογή και την ανάλυση δεδομένων σε τέσσερις κατηγορίες δεδομένων:

- Απόδοση εφαρμογής
- Απόδοση συστήματος
- Συμπεριφορά χρηστών εφαρμογής
- Συναίσθημα χρηστών

Ωστόσο, είναι σημαντικό οι ομάδες DevOps να μην συλλέγουν απλώς αυτά τα δεδομένα αλλά και να αναλύουν ώστε να επιτύχουν το ζητούμενο, ήτοι τη συνεχή βελτίωση. Δεν θα πρέπει να ξεχνούμε ότι αυτός ακριβώς είναι ο βασικός σκοπός του DevOps, η επίτευξη συνεχούς βελτίωσης. Η υιοθέτηση του DevOps δεν γίνεται στο πλαίσιο ενός έργου αλλά είναι μια συνεχής προσπάθεια. Στόχος είναι, όπως οραματίστηκε ο Senge τη δεκαετία του '90 (Garvin et al., 2008), η επιχείρηση να γίνει ένας οργανισμός μάθησης. Στην προσέγγιση DevOps, ένας οργανισμός πρέπει να μαθαίνει συνεχώς από το λογισμικό που μόλις παρέδωσε και να βελτιώνεται συνεχώς. Γενικότερα, μπορούμε να εστιαστούμε σε τρεις τομείς βελτίωσης:

- Την εφαρμογή. Οι αλλαγές της εφαρμογής που μόλις παραδόθηκε λειτουργούν και εκτελούνται με τον επιθυμητό τρόπο; Τι μπορείτε να μάθετε από τα σχόλια των χρηστών για να βελτιώσετε την εφαρμογή στην επόμενη επανάληψη;
- Το περιβάλλον στο οποίο εκτελείται η εφαρμογή, λειτουργεί όπως επιθυμείτε; Τηρούνται οι συμφωνίες επιπέδου υπηρεσιών (Service level Agreements - SLA); Τι μπορείτε να μάθετε από τη συνεχή ανατροφοδότηση για να βελτιώσετε το παραγωγικό περιβάλλον στην επόμενη επανάληψη;
- Τη διεργασία. Τι μπορείτε να μάθετε ώστε να βελτιώσετε τις ίδιες τις διεργασίες παράδοσης στην επόμενη επανάληψη;

Συνήθως, οι περισσότερες επιχειρήσεις καταβάλλουν σημαντικές προσπάθειες για τη συνεχή βελτίωση του λογισμικού που παρέχεται, πολύ λίγες όμως επιχειρήσεις κάνουν το ίδιο για τη συνεχή βελτίωση του παραγωγικού περιβάλλοντος, με βάση πραγματικές μετρήσεις.

6.2.7 Συνεχής Λειτουργία (Continuous Operations)

Η συνεχής λειτουργία έχει ως στόχο να διασφαλίσει ότι όλες οι ψηφιακές υπηρεσίες είναι διαθέσιμες και λειτουργούν σωστά, όταν οι πελάτες και οι χρήστες θέλουν να κάνουν χρήση τους, στο πλαίσιο πάντα της συμφωνίας SLAs που είναι σε ισχύ (Φιτσιλής, 2019).

Οι βασικοί στόχοι μιας ομάδας DevOps που στοχεύει στη συνεχή λειτουργία του συστήματος θα πρέπει να είναι:

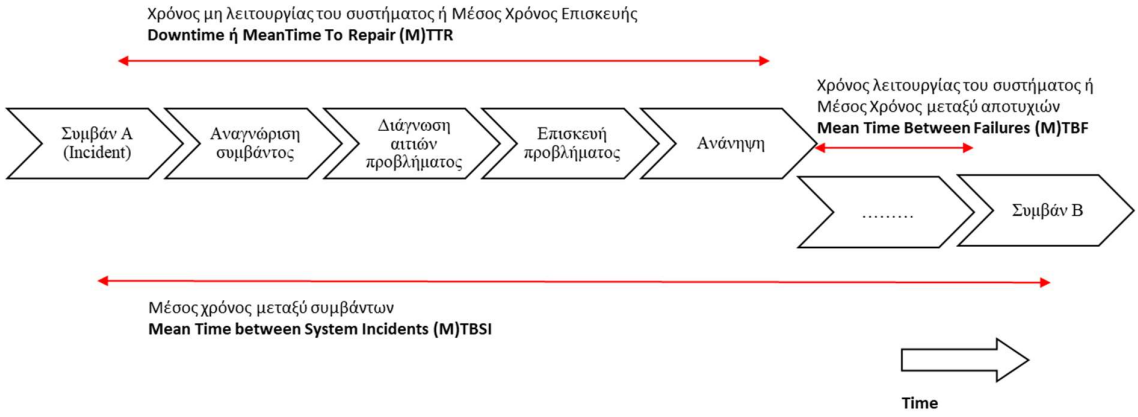
- Ο προσδιορισμός των απαιτήσεων διαθεσιμότητας της ψηφιακής υπηρεσίας, σε στενή συνεργασία με τους πελάτες,
- Η διασφάλιση του επιπέδου της διαθεσιμότητας των ψηφιακών υπηρεσιών,

- Η παρακολούθηση της διαθεσιμότητας των ψηφιακών υπηρεσιών και ο εντοπισμός των προβλημάτων, όταν αυτά συμβαίνουν στο μικρότερο δυνατό χρονικό διάστημα,
- Να προτείνει βελτιώσεις στις υποδομές πληροφορικής και των υπηρεσιών με στόχο την αύξηση των επιπέδων της διαθεσιμότητας.

Οι βασικοί δείκτες, τους οποίους θα πρέπει να μετρούμε και να βελτιστοποιούμε έχοντας ως στόχο την υψηλή διαθεσιμότητα των παρεχομένων υπηρεσιών είναι (Murthy & Kobbacy, 2008):

- Διαθεσιμότητα: Είναι ο χρόνος λειτουργίας της υπηρεσίας ως ποσοστό του συνολικού χρόνου που έχει συμφωνηθεί ότι οι υπηρεσίες πρέπει να είναι διαθέσιμες στους χρήστες.
- Αξιοπιστία: Είναι ο χρόνος της ορθής λειτουργίας των υπηρεσιών.
- Συντηρησιμότητα: Η ικανότητα να είναι η υπηρεσία λειτουργική ή αποκαταστάσιμη σε περίπτωση διακοπής.
- Δυναμικότητα: Καθορίζει τη διαθεσιμότητα και καταλληλόλητα των συστατικών της υπηρεσίας.

Στην Εικόνα 6.11 παρουσιάζεται διαδοχικά η αλληλουχία συμβάντων και το πώς τα διαχειριζόμαστε. Έστω ότι ένα γεγονός που προκαλεί προβληματική λειτουργία συμβαίνει. Η καταγραφή του γεγονότος είναι το πρώτο βήμα στη διαχείρισή του. Αν και αυτό ακούγεται τετριμμένο, στα πληροφοριακά συστήματα δεν είναι μια απλή ενέργεια, αφού πολλά γεγονότα δυσλειτουργίας μπορεί να περάσουν απαρατήρητα αν δεν έχουμε δημιουργήσει κατάλληλους μηχανισμούς καταγραφής, όπως ημερολόγια καταγραφής γεγονότων (event logs). Επιπλέον, σε μεγάλα πληροφοριακά συστήματα δημιουργούνται και καταγράφονται καθημερινά χιλιάδες γεγονότα και επομένως θα πρέπει να υπάρχουν μηχανισμοί φιλτραρίσματος και ειδοποίησης των χειριστών των συστημάτων με αυτοματοποιημένο τρόπο. Στη συνέχεια θα πρέπει να γίνει η διάγνωση των αιτιών του προβλήματος, ο προγραμματισμός της επισκευής και η ανάνηψη του συστήματος (Marcus, 2003).

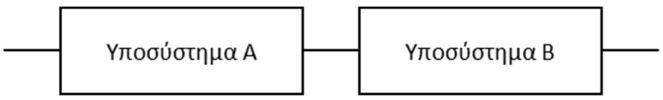
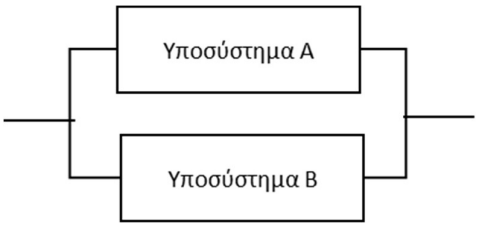
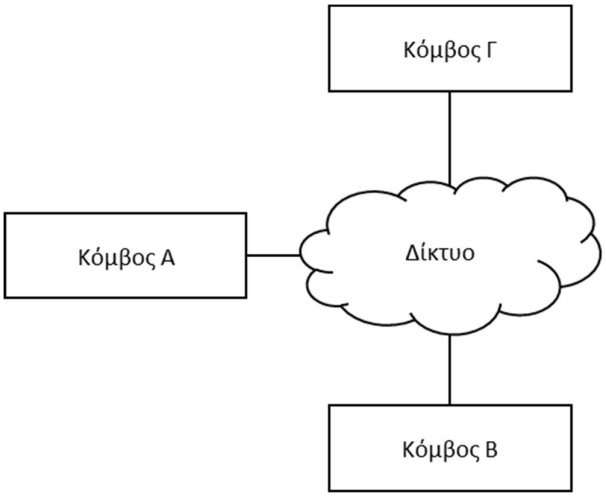


Εικόνα 6.11: *Η διαθεσιμότητα ενός πληροφοριακού συστήματος*

Ο υπολογισμός της διαθεσιμότητας είναι απλός και δίνεται από τον τύπο

$$Availability = \frac{MTBF}{MTBF + MTTR}$$

Στην πράξη όμως, επειδή ένα πληροφοριακό σύστημα αποτελείται από πολλά επιμέρους συστήματα, η συνολική διαθεσιμότητα είναι η συνισταμένη της διαθεσιμότητας των επιμέρους συστημάτων. Στην περίπτωση αυτή θα πρέπει να εξετάσουμε την τοπολογία του πληροφοριακού συστήματος και να δούμε αν τα υποσυστήματα είναι τοποθετημένα στη σειρά ή παράλληλα. Ο Πίνακας 6.1 παρουσιάζει τον τύπο υπολογισμού της διαθεσιμότητας ανάλογα με τη συνδεσμολογία (Φιτσιλής, 2019).

Διάταξη	Τύπος Υπολογισμού
	$Availability = Availability_A * Availability_B$
	$Availability = 1 - (1 - Availability_A)^2$
	Εάν η διαθεσιμότητα του κάθε κόμβου είναι a (όλοι οι κόμβοι είναι ίδιοι), τότε η πιθανότητα ότι ο κόμβος είναι εκτός λειτουργίας είναι (1-a). Ο συνολικός αριθμός των τρόπων που το σύστημα μπορεί να βγει εκτός λειτουργίας είναι ο συνδυασμός των διαφορετικών περιπτώσεων και δίνεται από τον τύπο: $F = \frac{n * (n-1)}{2}$ Όπου n είναι ο αριθμός των διαθέσιμων κόμβων. Η διαθεσιμότητα του συστήματος δίνεται από τον τύπο: $\text{Availability} = 1 - \frac{n * (n-1)}{2} * (1-a)^2$ Για παράδειγμα εάν ένα σύστημα αποτελείται από τρεις κόμβους, όπου κάθε κόμβος έχει διαθεσιμότητα 0,99, τότε η συνολική διαθεσιμότητα του συστήματος είναι: $\text{Availability} = 1 - \frac{3 * (3-1)}{2} * (1-0{,}99)^2$ $= 1 - 3 * (0{,}01)^2 = 1 - 0{,}0003 = 0{,}9997$

Πίνακας 6.1: *Διαφορετικές περιπτώσεις υπολογισμού διαθεσιμότητας.*

Η διαθεσιμότητα συνήθως υπολογίζεται με τον αριθμό των 9 που υπάρχουν στο ποσοστό της διαθεσιμότητας. Ο Πίνακας 6.2 παρουσιάζει τη διαθεσιμότητα καθώς και τον χρόνο που ένα σύστημα δεν θα είναι διαθέσιμο για το αντίστοιχο ποσοστό. Τα ποσοστά διαθεσιμότητας καθορίζονται από την κρισιμότητα του συστήματος για την επιχείρηση και αντίστοιχα καθορίζουν την αρχιτεκτονική του πληροφοριακού συστήματος αλλά και τις διαθέσιμες εφεδρείες που υπάρχουν. Για παράδειγμα, η επίτευξη διαθεσιμότητας 99,9999% είναι ένας εξαιρετικά φιλόδοξος αλλά και πολύ δύσκολος στόχος.

Διαθεσιμότητα	Αριθμός 9	Χρόνος μη διαθεσιμότητας
90%	1	36,5 ημέρες/έτος

Διαθεσιμότητα	Αριθμός 9	Χρόνος μη διαθεσιμότητας
99%	2	3,65 ημέρες/έτος
99,9%	3	8,76 ώρες/έτος
99,99%	4	52 λεπτά/έτος
99,999%	5	5 λεπτά/έτος
99,9999%	6	31 δευτερόλεπτα/έτος

Πίνακας 6.2: *Κατηγορίες διαθεσιμότητας συστημάτων.*

Αντίστοιχα, ο μέσος χρόνος επισκευής (MTTR) είναι ιδιαίτερα σημαντικός αφού καθορίζει την πολιτική που θα ακολουθήσουμε τόσο για το διαθέσιμο προσωπικό τεχνικής υποστήριξης όσο και για την αποθήκη ανταλλακτικών. Ο Πίνακας 6.3 παρουσιάζει ενδεικτικές περιπτώσεις διαθεσιμότητας προσωπικού και ανταλλακτικών και την επίπτωση που έχουν στον μέσο χρόνο επισκευής.

Περίπτωση	Ανταλλακτικά	Προσωπικό	Εκτιμώμενος μέσος χρόνος επισκευής (MTTR)
Αλλαγή υλικού (π.χ. σκληρού δίσκου)	Στην αποθήκη της επιχείρησης	Διαθέσιμο σε 24ωρη βάση	30 λεπτά
Αλλαγή υλικού (π.χ. σκληρού δίσκου)	Στην αποθήκη της επιχείρησης	Διαθέσιμο σε ώρες εργασίας από Δευτέρα ως Παρασκευή	Έως τρεις ημέρες εάν το πρόβλημα συμβεί την Παρασκευή το απόγευμα.
Αλλαγή υλικού (π.χ. σκληρού δίσκου)	Μη διαθέσιμο, πρέπει να γίνει αγορά και αποστολή. Η αποστολή απαιτεί δύο ημέρες	Διαθέσιμο σε ώρες εργασίας από Δευτέρα ως Παρασκευή	Έως τρεις ημέρες εάν το πρόβλημα συμβεί την Παρασκευή το απόγευμα. Το εξάρτημα θα παραγγελθεί την Δευτέρα, θα το παραλάβουμε και θα το εγκαταστήσουμε την Τετάρτη.
Πρόβλημα στο λογισμικό	Δεν απαιτούνται	Διαθέσιμο σε 24ωρη βάση	Άμεση εργασία πάνω στο πρόβλημα
Πρόβλημα στο λογισμικό	Δεν απαιτούνται	Διαθέσιμο σε ώρες εργασίας	Την ίδια ή επόμενη ημέρα.
Πρόβλημα στο λογισμικό	Δεν απαιτούνται	Η συντήρηση γίνεται από εξειδικευμένη εταιρεία	Ανάλογα με το συμβόλαιο συντήρησης και SLA. Συνήθως την επόμενη εργάσιμη ημέρα.

Πίνακας 6.3: *Περιπτώσεις προβλημάτων και ο μέσος χρόνος επισκευής.*

Μια σημαντική παράμετρος της συνεχούς λειτουργίας είναι η συνέχεια της λειτουργίας (service continuity) των ψηφιακών υπηρεσιών που στοχεύει στην εξασφάλιση της ικανότητας μιας επιχείρησης, σε περίπτωση καταστροφής, να επαναφέρει τις υπηρεσίες των πληροφοριακών συστημάτων σε ένα συγκεκριμένο επίπεδο λειτουργίας, το οποίο θα καλύπτει τις ελάχιστες απαραίτητες επιχειρησιακές λειτουργίες. Συνήθως, το επίπεδο λειτουργίας είναι υποδεέστερο αυτού της κανονικής λειτουργίας, αλλά σε κάθε περίπτωση εξασφαλίζει κατ' ελάχιστο τη λειτουργία της επιχείρησης.

Τα παραπάνω διαφοροποιούνται σημαντικά από τη λειτουργία και τη χρήση του Υπολογιστικού Νέφους (ΥΝ). Πρόκειται για μια τάση που είναι έντονη και διαφαίνεται να ισχυροποιείται καθημερινά καθώς όλο και περισσότερες επιχειρήσεις επιλέγουν τη λειτουργία των συστημάτων τους στο υπολογιστικό νέφος (computing cloud). Αντίστοιχα, οι ομάδες DevOps επιλέγουν όλο και περισσότερο για τη λειτουργία των συστημάτων το ΥΝ, διότι τα πλεονεκτήματα που προσφέρει είναι ιδιαίτερα σημαντικά. Σύμφωνα με τον ορισμό του NIST (National Institute of Standards and Technology) το ΥΝ έχει πέντε βασικά χαρακτηριστικά (Mell & Grance, 2011):

- **Είναι διαθέσιμο κατ' απαίτηση (on-demand)**, δηλαδή η επιχείρηση μπορεί να προμηθευτεί υπολογιστικούς πόρους όπως π.χ. χρόνο στον διακομιστή ή χώρο αποθήκευσης όποτε το χρειαστεί, αυτομάτως, χωρίς να απαιτείται η παρέμβαση από τον πάροχο της κάθε υπηρεσίας.

Επίσης, οι χρήστες του ΥΝ **αυτοεξυπηρετούνται (**self-service), γεγονός που σημαίνει ότι δεν απαιτείται στη γενική περίπτωση η παρέμβαση κάποιου χειριστή.

- Οι χρήστες του ΥΝ έχουν **ευρυζωνική σύνδεση** στο διαδίκτυο (broadband access). Όλες οι δυνατότητες είναι διαθέσιμες μέσω του διαδικτύου και η πρόσβαση γίνεται μέσω τυποποιημένων μηχανισμών που λειτουργούν σε ετερογενείς συσκευές (π.χ. κινητά τηλέφωνα, tablets, φορητοί υπολογιστές και σταθμοί εργασίας). Με απλά λόγια οι χρήστες, εφόσον διαθέτουν σύνδεση στο διαδίκτυο, έχουν πρόσβαση στα δεδομένα και στις εφαρμογές τους από οποιονδήποτε τόπο και από οποιαδήποτε συσκευή, είτε πρόκειται για προσωπικό υπολογιστή, ταμπλέτα ή έξυπνο τηλέφωνο.

- **Συγκέντρωση πόρων (*resource pooling)***. Οι υπολογιστικοί πόροι του παρόχου υπηρεσιών ΥΝ συγκεντρώνονται για να εξυπηρετούν πολλούς καταναλωτές μέσω πολλαπλών μισθώσεων. Στο μοντέλο αυτό διαφορετικοί φυσικοί και εικονικοί πόροι εκχωρούνται δυναμικά σε συνάρτηση με τη ζήτηση των καταναλωτών. Η παροχή των υπηρεσιών είναι ανεξάρτητη από την τοποθεσία, δεδομένου ότι ο πελάτης δεν έχει κανέναν έλεγχο ή γνώση σχετικά με την ακριβή τοποθεσία των παρεχόμενων πόρων, αλλά μπορεί να είναι σε θέση να προσδιορίσει την τοποθεσία σε υψηλότερο επίπεδο (π.χ. χώρα ή κέντρο δεδομένων). Παραδείγματα υπολογιστικών πόρων περιλαμβάνουν τους αποθηκευτικούς πόρους, τις μονάδες επεξεργασίας, τη διαθέσιμη μνήμη και το εύρος ζώνης δικτύου.

- **Ταχεία ελαστικότητα (rapid elasticity)**. Οι πόροι του ΥΝ παρέχονται ή αναιρούνται δυναμικά και σε ορισμένες περιπτώσεις αυτομάτως, ανάλογα με τη ζήτηση. Για τον καταναλωτή, οι διαθέσιμες υπολογιστικές δυνατότητες συχνά φαίνονται απεριόριστες και μπορούν να χρησιμοποιηθούν σε οποιαδήποτε ποσότητα ανά πάσα στιγμή. Αυτό σημαίνει ότι ο πάροχος υπηρεσιών του ΥΝ προσαρμόζει σε πραγματικό χρόνο την υπολογιστική ισχύ στις ανάγκες του εκάστοτε χρήστη. Από την πλευρά του χρήστη αυτό σημαίνει ότι ο χρήστης θα μπορεί να καλύπτει τις ανάγκες του, ακόμη και σε περιόδους αιχμής, χωρίς να πρέπει να επενδύσει σε εξοπλισμό πληροφορικής που θα χρησιμοποιείται ελάχιστα ανάμεσα σε δύο περιόδους αιχμής. Αυτή η ελαστικότητα είναι εφικτή επειδή ο πάροχος θέτει τους υπολογιστικούς πόρους στη διάθεση πολλών χρηστών ταυτόχρονα. Η πρακτική αυτή επιτρέπει τη μέγιστη και καλύτερη δυνατή αξιοποίηση τεράστιων πάρκων εξυπηρετητών με πολλές χιλιάδες ηλεκτρονικούς υπολογιστές.

  Οι ελαστικοί πόροι αποτελούν κρίσιμη παράμετρο για τη μείωση του κόστους (Total Cost of Ownership - TCO) και τη μείωση του χρόνου εισαγωγής στην αγορά (Time to Market - TTM), αφού η πλειονότητα των δαπανών που συνδέονται με την ανάπτυξη υπηρεσιών προέρχεται από την προμήθεια και εγκατάσταση πόρων, και, εκ τούτου, απλοποιώντας τη διαδικασία παροχής πόρων μπορεί να δημιουργηθεί σημαντική μείωση του κόστους και να επιτραπεί η ταχύτερη παραγωγή εσόδων.

  Η ελαστικότητα εξαρτάται από τη μορφή της ζήτησης για υπολογιστικές υπηρεσίες, η οποία μεταβάλλεται και δεν είναι σταθερή. Για παράδειγμα, κάποιες ώρες της ημέρας υπάρχει μικρή ζήτηση για υπολογιστικές υπηρεσίες (π.χ. βραδινές ώρες), ή κάποιες περιόδους του έτους ή κάποιες ημέρες της εβδομάδας υπάρχει μεγάλη ζήτηση (π.χ. κάθε Σάββατο βράδυ οι καταναλωτές βλέπουν ταινίες σε συνδρομητική ψηφιακή τηλεόραση), ή σε κάποιες συγκεκριμένες στιγμές υπάρχει υπερβολική ζήτηση (π.χ. κατά τη διάρκεια μιας συναυλίας ή κατά τη διενέργεια εκλογών). Η ελαστικότητα λύνει ακριβώς αυτό το πρόβλημα με τον πιο αποδοτικό τρόπο, δηλαδή να διαθέτει τους κατάλληλους υπολογιστικούς πόρους στον καταναλωτή, την κατάλληλη στιγμή.

- Το ΥΝ προσφέρει **μετρήσιμες υπηρεσίες (measured services)**. Ο όρος "μετρήσιμη υπηρεσία" συνεπάγεται ότι η χρήση των υπολογιστικών πόρων παρακολουθείται και παρουσιάζεται στον καταναλωτή, παρέχοντας ικανότητα ενημέρωσης για την ποσότητα κατανάλωσης και των συναφών εξόδων. Τα συστήματα ΥΝ υλοποιούν ακριβώς αυτή την απαίτηση, δηλαδή ελέγχουν αυτόματα και βελτιστοποιούν την χρήση των πόρων, μέσω της μέτρησης αυτών. Επομένως, σε ένα ΥΝ η χρήση των πόρων παρακολουθείται, ελέγχεται και αναφέρεται, παρέχοντας διαφάνεια

τόσο για τον πάροχο όσο και για τον καταναλωτή της υπηρεσίας. Επιπλέον, οι παρεχόμενες υπηρεσίες παρέχονται σε εγγυημένο επίπεδο (χρήση Service Level Agreement - SLA). Για παράδειγμα, όποτε διαπιστώνεται αυξημένη χρήση μιας υπηρεσίας του ΥΝ, η οποία μειώνει την απόδοση αυτής (performance) σε σχέση με την αναμενόμενη, είναι πολύ απλό να προστεθούν επιπλέον υπολογιστικοί πόροι ώστε η απόδοση της υπηρεσίας να μείνει σταθερή.

Ο συνδυασμός της προσέγγισης DevOps με το ΥΝ αποτελεί ένα επιτυχημένο δίδυμο που επιτρέπει το γρήγορο και αποτελεσματικό ψηφιακό μετασχηματισμό των επιχειρήσεων για πολλούς λόγους:

- Επιτρέπει την ταχύτερη προσφορά των προϊόντων στην αγορά μέσω ταχύτερης πρόσβασης σε περιβάλλοντα ανάπτυξης και των βελτιστοποιημένων διαδικασιών προγραμματισμού.
- Οι διαθέσιμοι αυτοματισμοί και η προσέγγιση «υποδομή ως Κώδικας» (IaC) μειώνουν την πολυπλοκότητα του ΥΝ καθώς και συντήρηση αυτού.
- Το ΥΝ παρέχει υψηλή ασφάλεια με αυτοματοποιημένες, επαναλαμβανόμενες διαδικασίες που χρησιμεύουν για την εξάλειψη των σφαλμάτων και την ανάπτυξη ελέγχων ασφαλείας από την αρχή.
- Το ΥΝ αυξάνει τη διαθεσιμότητα του πληροφοριακού συστήματος με χρήση διάφορων τεχνικών ή αρχιτεκτονικών (π.χ. restful applications).
- Το ΥΝ αυξάνει την ελαστικότητα και την επεκτασιμότητα των ψηφιακών υπηρεσιών της επιχείρησης.

6.3 Τα χαρακτηριστικά της αρχιτεκτονικής DevOps

Οι αρχές του καλού αρχιτεκτονικού σχεδιασμού είναι γνωστές ήδη από την αρχαιότητα και περιγράφονται από τον Ρωμαίο αρχιτέκτονα Βιτρούβιο στο έργο «De architecture». Στο έργο αυτό περιγράφονται οι έννοιες «firmitas, utilitas, venustas», δηλαδή ότι η αρχιτεκτονική πρέπει να είναι σταθερή, χρήσιμη και όμορφη (McEvwen, 2004). Αντίστοιχα, η αρχιτεκτονική της μεθόδου DevOps είναι βασισμένη σε έξι άξονες που αντιστοιχούν γενικότερα σε μη λειτουργικά χαρακτηριστικά του λογισμικού. Αυτοί είναι (βλέπε Εικόνα 6.12):

- Συντηρησιμότητα - Maintainability
- Δυνατότητα κλιμάκωσης - Scalability
- Ασφάλεια - Security
- Χρηστικότητα - Usability
- Αξιοπιστία - Reliability
- Ευελιξία – Flexibility

Εικόνα 6.12: *Οι αρχιτεκτονικοί άξονες της προσέγγισης DevOps*

Το Πρότυπο Γλωσσάρι της Ορολογίας Τεχνολογίας Λογισμικού IEEE (1983) ορίζει τη **συντηρησιμότητα** (maintenability) ως την «ευκολία με την οποία ένα σύστημα λογισμικού ή συστατικό αυτού μπορεί να τροποποιηθεί για να διορθωθούν σφάλματα, να βελτιώσει την απόδοση ή άλλα χαρακτηριστικά ή να προσαρμοστεί σε ένα διαφορετικό αλλαγμένο περιβάλλον».

Γενικότερα η έννοια της συντηρησιμότητας των πληροφορικών συστημάτων θα μπορούσε να παρουσιαστεί λέγοντας πως ένα σύστημα χαρακτηρίζεται συντηρήσιμο όταν αναγνωρίζονται και αντιμετωπίζονται σφάλματά του σε επιθυμητά χρονικά διαστήματα με σχετική ευκολία. Η έννοια της συντηρησιμότητας έχει τέσσερις διαφορετικές οπτικές γωνίες που είναι οι ακόλουθες:

- Ακρίβεια (Accuracy and Clarity) στην τεκμηρίωση
- Αρθρωτός σχεδιασμός (modular)
- Αναγνωσιμότητα (Readability) και
- Απλότητα (Simplicity)

Με απλά λόγια η δυνατότητα συντήρησης του λογισμικού εξαρτάται από μερικούς διαφορετικούς παράγοντες, όπως πρέπει να είναι εύκολο να κατανοήσετε το λογισμικό (πώς λειτουργεί, τι κάνει και γιατί το κάνει όπως το κάνει), να είναι εύκολο να βρείτε τι πρέπει να αλλάξετε, να είναι εύκολο να κάνετε αλλαγές και τέλος να είναι εύκολο να ελέγξετε ότι οι αλλαγές δεν έχουν εισαγάγει νέα σφάλματα. Η συντηρησιμότητα (maintenability) είναι απαραίτητη επειδή ο κώδικας είναι σύνθετος και πάντα αλλάζει. Επίσης είναι αναγκαία διότι ο κώδικας θα πρέπει να ελέγχεται αυτόματα και να τεκμηριώνεται επαρκώς.

Η **επεκτασιμότητα** ή δυνατότητα κλιμάκωσης είναι το μέτρο της ικανότητας ενός συστήματος να αυξάνει ή να μειώνει την απόδοση και το κόστος του ως απόκριση στις αλλαγές των απαιτήσεων του πελάτη. Για παράδειγμα, πόσο καλά αποδίδει ένα σύστημα όταν αυξάνεται ο αριθμός των χρηστών ή πόσο καλά μια βάση δεδομένων ανταποκρίνεται σε αυξανόμενους αριθμούς ερωτημάτων κ.λπ. Η ιδιότητα αυτή είναι σημαντική για επιχειρήσεις που αναπτύσσονται γρήγορα ή σε περιπτώσεις που υπάρχει περιοδική ή εποχική ζήτηση. Η επεκτασιμότητα υλοποιείται είτε με την προσθήκη περισσότερης υπολογιστικής ισχύος, με την κάλυψη περισσότερων γεωγραφικών περιοχών, είτε με τη χρήση του υπολογιστικού νέφους. Θα πρέπει να

σημειωθεί ότι οι έννοιες της ελαστικότητας και η έννοια της κλιμάκωσης δεν ταυτίζονται, αφού η πρώτη αφορά τη μεταβλητή χρήση πόρων ανάλογα με τη ζήτηση, ενώ η δεύτερη την επέκταση των διαθέσιμων πόρων.

Η **ευχρηστία ή χρηστικότητα** είναι το μέτρο του πόσο καλά ένας συγκεκριμένος χρήστης σε ένα συγκεκριμένο πλαίσιο μπορεί να χρησιμοποιήσει ένα προϊόν/σχέδιο για να επιτύχει έναν καθορισμένο στόχο αποτελεσματικά, αποδοτικά και ικανοποιητικά. Η χρηστικότητα ενός προϊόντος είναι ένα θέμα που απασχολεί την ομάδα DevOps σε όλη τη διαδικασία ανάπτυξης - από τις απαιτήσεις έως το τελικό λογισμικό. Ο σχεδιασμός ενός εύχρηστου συστήματος λογισμικού είναι ιδιαίτερα δύσκολη υπόθεση αφού απαιτεί καλή κατανόηση του πεδίου προβλήματος, των αναγκών των χρηστών αλλά και μιας σειράς παραμέτρων όπως ενδεικτικά παρουσιάζεται παρακάτω:

- Μίμηση του πραγματικού κόσμου όσον αφορά τις έννοιες, τα εικονίδια και τη γλώσσα.
- Παρουσίαση άμεσων κατανοητών μηνυμάτων και ενεργειών που μπορούν να κάνουν οι χρήστες.
- Περιορισμένες επιλογές για τους χρήστες σε κάθε βήμα, ώστε να επιτυγχάνεται η απλότητα.
- Διατήρηση του περιεχομένου με συνέπεια.
- Προσκόλληση στους καθιερωμένους κανόνες σχετικά με τη λειτουργία και τη διάταξη των οπτικών στοιχείων (π.χ. τοποθέτηση λογότυπου, κουμπιά ιδίου μεγέθους, κ.λπ.).
- Χρήση κατάλληλου μεγέθους γραμματοσειράς, χρώμα, αντίθεση, κενά κ.λπ..
- Παρουσίαση ενημερωτικών μηνυμάτων σχετικά με την κατάσταση του συστήματος.
- Κ.λπ.

Η **αξιοπιστία λογισμικού** (software reliability) είναι η πιθανότητα λειτουργίας ενός συστήματος λογισμικού χωρίς αστοχίες για μια καθορισμένη περίοδο, σε ένα καθορισμένο περιβάλλον λειτουργίας. Η αξιοπιστία αποτελεί μια πελατοκεντρική άποψη για την ποιότητα του λογισμικού και σχετίζεται περισσότερο με τη λειτουργία παρά με το σχεδιασμό του λογισμικού, και ως εκ τούτου είναι ένα δυναμικό χαρακτηριστικό, που αλλάζει με τον χρόνο, παρά στατικό. Επίσης, η αξιοπιστία ως έννοια συνδέεται στενά με την διαθεσιμότητα που εξετάσαμε σε προηγούμενη παράγραφο. Η μέτρηση και η πρόβλεψη της αξιοπιστίας λογισμικού είναι ζωτικής σημασίας για τους διαχειριστές λογισμικού, τους μηχανικούς λογισμικού, τους χρήστες των προϊόντων κ.λπ. Η μέτρηση της αξιοπιστίας περιλαμβάνει:

- Διαθεσιμότητα: Το % του χρόνου λειτουργίας και πρόσβασης ενός συστήματος από τους πελάτες του.
- Λανθάνων χρόνος: Ο χρόνος μεταξύ αιτήσεων χρήστη και αποκρίσεων συστήματος.
- Ταχύτητα εκτέλεσης συναλλαγών: Πόσες συναλλαγές μπορεί να εκτελέσει το σύστημα ανά δευτερόλεπτο.
- Πιστότητα: Το επίπεδο στο οποίο το σύστημα αντιπροσωπεύει την πραγματική κατάσταση ενός φυσικού αντικειμένου.
- Ανθεκτικότητα: Η ικανότητα των συστημάτων να εξυπηρετούν τους χρήστες με την πάροδο του χρόνου.
- Κ.λπ.

Αντίστοιχα, **ευελιξία** (flexibility) ορίζεται ως η ικανότητα ενός συστήματος να εξελίσσεται για την ικανοποίηση των απαιτήσεων των πελατών με την πάροδο του χρόνου. Η ικανότητα προσαρμογής του λογισμικού τόσο σε σταδιακές όσο και σε δραστικές αλλαγές των επιχειρηματικών απαιτήσεων ή διαδικασιών στον ελάχιστο δυνατό χρόνο, προσπάθεια, κόστος (Nurdiani et al., 2018).

Τέλος η ασφάλεια είναι ένας ιδιαίτερα σημαντικός παράγοντας. Το γεγονός αυτό έχει οδηγήσει στην προσαρμογή της προσέγγισης DevOps, ώστε να ενσωματώσει την ασφάλεια μέσα στο DevOps με τη

δημιουργία του DevSecOps (https://www.devsecops.org/). Συνεπώς, το DevSecOps είναι μια πρακτική ανάπτυξης που ενσωματώνει πρακτικές ασφάλειας σε κάθε στάδιο του κύκλου ζωής ανάπτυξης λογισμικού για την ανάπτυξη ασφαλών εφαρμογών.

Ιστορικά, τα ζητήματα ασφάλειας αντιμετωπίζονται αργά στον κύκλο ζωής λογισμικού ή αντιμετωπίζονται ως μη λειτουργικές απαιτήσεις ή αντιμετωπίζονται κατά περίπτωση. Ωστόσο, με δεδομένο την ύπαρξη εξελιγμένων επιθέσεων κυβερνοασφάλειας και τις ομάδες ανάπτυξης που στρέφονται σε πιο σύντομες, συχνότερες επαναλήψεις σε εφαρμογές, το DevSecOps γίνεται πλέον μια αναγκαία πρακτική που εντάσσει την ασφάλεια των εφαρμογών μέσα σε αυτό το σύγχρονο οικοσύστημα ανάπτυξης. Η βασική λογική είναι ότι οι απαιτήσεις ασφάλειας εξ ’αρχής θα πρέπει να είναι βασική προτεραιότητα της ομάδας DevOps (Rajapakse et al., 2022).

Η εισαγωγή αυτής της λογικής στην προσέγγιση DevOps προϋποθέτει ότι (Wilson, 2020):

- Η ομάδα DevOps έχει την αναγκαία τεχνογνωσία σε θέματα ασφαλείας. Το πλήθος των διαφορετικών τύπων κινδύνων και επιθέσεων, οι διαφορετικές περιπτώσεις, τα διαφορετικά συστήματα κάνουν αυτή την τεχνογνωσία πολύ δύσκολο να αποκτηθεί. Σε πολλές περιπτώσεις έχει χρησιμοποιηθεί ο ρόλος του μηχανικού ασφαλείας αλλά σε κάθε περίπτωση όλη η ομάδα DevOps πρέπει να έχει σχετικές γνώσεις.
- Ο σχεδιασμός ασφαλείας θα πρέπει να είναι αρχική προϋπόθεση, τόσο σε οργανωτικό αλλά και σε τεχνικό επίπεδο. Τα συστήματα πρέπει είναι secure by design. Για το σχεδιασμό της ασφάλειας υπάρχει μεγάλος αριθμός τεχνικών αλλά και πολιτικών που μπορούν να χρησιμοποιηθούν όπως: μοντελοποίηση απειλών, κατηγοριοποίηση απειλών, τεχνικές για τη βελτίωση την ποιότητας του κώδικα (π.χ. clean code), ισχυρή αυθεντικοποίηση, λογοδοσία, έλεγχος πρόσβασης, κρυπτογράφηση κ.λπ..
- Θα πρέπει να υπάρχει αυτοματοποίηση των διαδικασιών ασφάλειας όπου είναι εφικτό. Στο σημείο αυτό θα πρέπει να τονίσουμε:
    - Την εισαγωγή τη λογικής του ελέγχου με έμφαση στην ασφάλεια σε όλα τα επίπεδα ελέγχου, από το μοναδιαίο έλεγχο έως το στατικό (Static Application Security Testing – SAST) και δυναμικό έλεγχο της ασφάλειας του λογισμικού (Dynamic Application Security Testing – DAST).
    - Τον έλεγχο ασφαλείας της υποδομής, δικτύου, κ.λπ. Η χρήση ΥΝ κάνει την όλη διαδικασία ακόμη πιο σύνθετη.
    - Τη συνεχή παρακολούθηση των απειλών, την άμεση ειδοποίηση της ομάδας DevOps και την άμεση, αυτοματοποιημένη αν είναι δυνατόν, απόκριση.
    - Τη χρήση μεγάλου αριθμού διαφορετικών εργαλείων για τη διαχείριση της ασφάλειας.

Βέβαια σύμφωνα με τον Ribeiro (2021), το DevSecOps σχετίζεται με την αλλαγή κουλτούρας των επιχειρήσεων ICT (Information and Communication Technology) και στοχεύει στην ενσωμάτωση της ασφάλειας στη σύγχρονη ανάπτυξη λογισμικού με τη χρήση της προσέγγισης DevOps. Ο απώτερος στόχος του DevSecOps είναι η συνεργασία των ομάδων ανάπτυξης, ασφάλειας και λειτουργίας με σκοπό τη δημιουργία επιχειρηματικής αξίας μέσω της γρήγορης παράδοσης ασφαλούς λογισμικού χρησιμοποιώντας την έννοια της «συνεχούς ασφάλειας».

6.4 Βασικές τεχνολογίες, εργαλεία του Devops

Η βασική ιδέα του οικοσυστήματος DevOps είναι ότι σε κάθε βήμα το κατάλληλο εργαλείο πρέπει να βοηθά στον αυτοματισμό κάθε διεργασίας. Για την επίτευξη αυτού του στόχου υπάρχει πληθώρα εργαλείων τα οποία συνδυαζόμενα αυτοματοποιούν την κατασκευή και λειτουργία του λογισμικού.

Ο Πίνακας 6.4 παρουσιάζει μια συλλογή από δημοφιλή εργαλεία ανά φάση του DevOps. Οπωσδήποτε η επιλογή του κάθε εργαλείου είναι συνδυασμός πολλών παραγόντων, όπως της τεχνογνωσίας της ομάδας DevOps, του συνδυασμού των εργαλείων που έχουν επιλογή, την ολοκλήρωση που προσφέρουν, κ.λπ. (Hernantes et al., 2016).

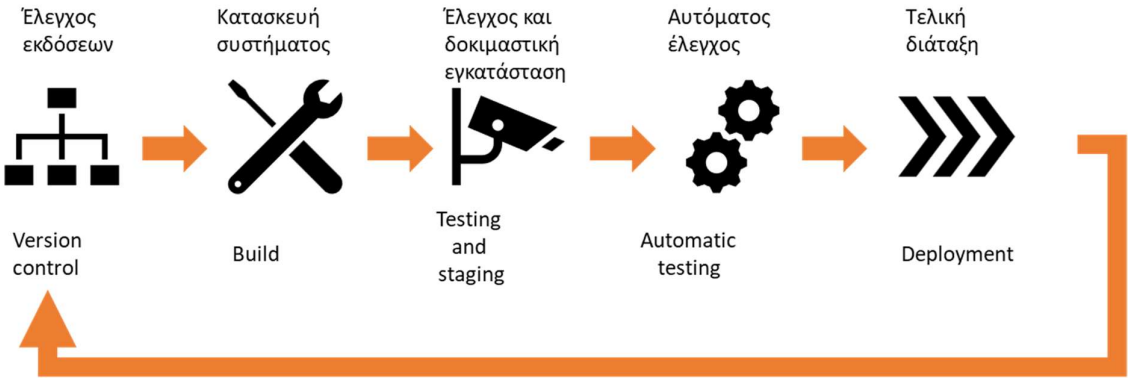
Φάση	**Εργαλεία**
Continuous Planning & Development	Git, SVN, Mercurial, Jira, Trello, Maven, Gradle, Subversion
Continuous Integration	Jenkin, Gitlab, TeamCity
Continuous Testing	Selenium, Junit, Jasmine, Cucumber
Continuous Deployment	Ansible, Chef, Docker, Kubernetes, Puppet
Continuous Monitoring	Splunk, Nagios, Kibana, Logstash,
Customer Feedback	Slack, ServiceNow, Flowdock
Continuous Operations	Chef, Kubernetes, Docker Swarm
Scripting	Perl, Python, Bash

Πίνακας 6.4: *Τα εργαλεία του οικοσυστήματος DevOps*

6.4.1 Οι αγωγοί (pipelines) DevOps

Μια βασική έννοια της προσέγγισης DevOps είναι η έννοια του αγωγού (pipeline). Ένας αγωγός DevOps είναι ένα σύνολο αυτοματοποιημένων διαδικασιών και εργαλείων που επιτρέπει στην ομάδα DevOps να εργαστεί συνεκτικά για την ανάπτυξη κώδικα σε ένα περιβάλλον παραγωγής. Συνήθως, ενώ ο σκοπός ενός αγωγού DevOps είναι κοινός, η επιλογή των εργαλείων και οι αυτοματισμοί που παρέχουν διαφέρουν ανά επιχείρηση. Διαφέρουν ως προς τα εργαλεία που επιλέγονται, στους αυτοματισμούς που παρέχει αλλά και στους τρόπους και τα σημεία ελέγχου που απαιτούν ανθρώπινη παρέμβαση προτού επιτραπεί να προχωρήσει ο κώδικας/λογισμικό σε επόμενη φάση (βλέπε Εικόνα 6.13).

Δεδομένου ότι δεν υπάρχει ένας τυπικός αγωγός DevOps, ο σχεδιασμός και η υλοποίηση ενός αγωγού DevOps από έναν οργανισμό εξαρτάται από τη στοίβα τεχνολογίας, το επίπεδο εμπειρίας, τον διαθέσιμο προϋπολογισμό και πολλά άλλα.



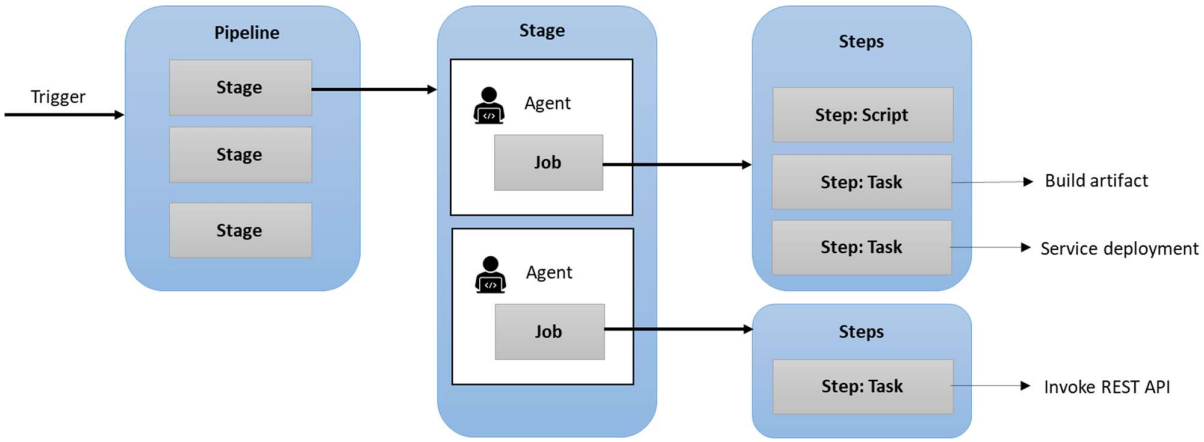
Εικόνα 6.13: *DevOps CI/CD pipeline*

Σε τεχνικό επίπεδο ένας αγωγός επιτρέπει την υλοποίηση ενός CI ή CD αγωγού σε οποιαδήποτε γλώσσα προγραμματισμού, ή για οποιαδήποτε πλατφόρμα στόχος (Azure Pipelines, 2022*)*.

Ένας αγωγός έχει τα παρακάτω τεχνικά χαρακτηριστικά:

- Ο αγωγός ξεκινά να εκτελείται μετά από ένα γεγονός (trigger). Ένα γεγονός μπορεί να συνδέεται με την αποθήκευση ενός αρχείου στο αποθετήριο ή να είναι ο προγραμματισμένος χρόνος για να ξεκινήσει η εκτέλεση του αγωγού.
- Ένας αγωγός αποτελείται από ένα ή περισσότερα στάδια (stages). Κάθε στάδιο είναι ένα λογικό όριο του αγωγού (π.χ. build, test)
- Ένας αγωγός μπορεί να αναπτυχθεί σε ένα ή περισσότερα περιβάλλοντα ανάπτυξης.
- Ένα στάδιο είναι ένας τρόπος οργάνωσης εργασιών και κάθε στάδιο μπορεί να έχει μία ή περισσότερες εργασίες (jobs).
- Κάθε εργασία εκτελείται σε έναν πράκτορα (agent). Μια δουλειά μπορεί επίσης να είναι χωρίς πράκτορα.
- Κάθε πράκτορας εκτελεί μια εργασία που περιέχει ένα ή περισσότερα βήματα.
- Ένα βήμα μπορεί να είναι μια εργασία (task) ή ένα σενάριο (script) και είναι το μικρότερο δομικό στοιχείο ενός αγωγού.
- Μια εργασία είναι ένα προσυσκευασμένο σενάριο που εκτελεί μια ενέργεια, όπως η κλήση ενός REST API (Representational State Transfer API) ή η δημοσίευση ενός τεχνουργήματος κατασκευής (build artifact).
- Ένα τεχνούργημα (artifact) είναι μια συλλογή αρχείων ή πακέτων που δημοσιεύονται σε μια εκτέλεση.

Οι βασικές έννοιες ενός αγωγού (pipeline) παρουσιάζονται στην Εικόνα 6.14.



Εικόνα 6.14: *Βασικές έννοιες ενός pipeline*

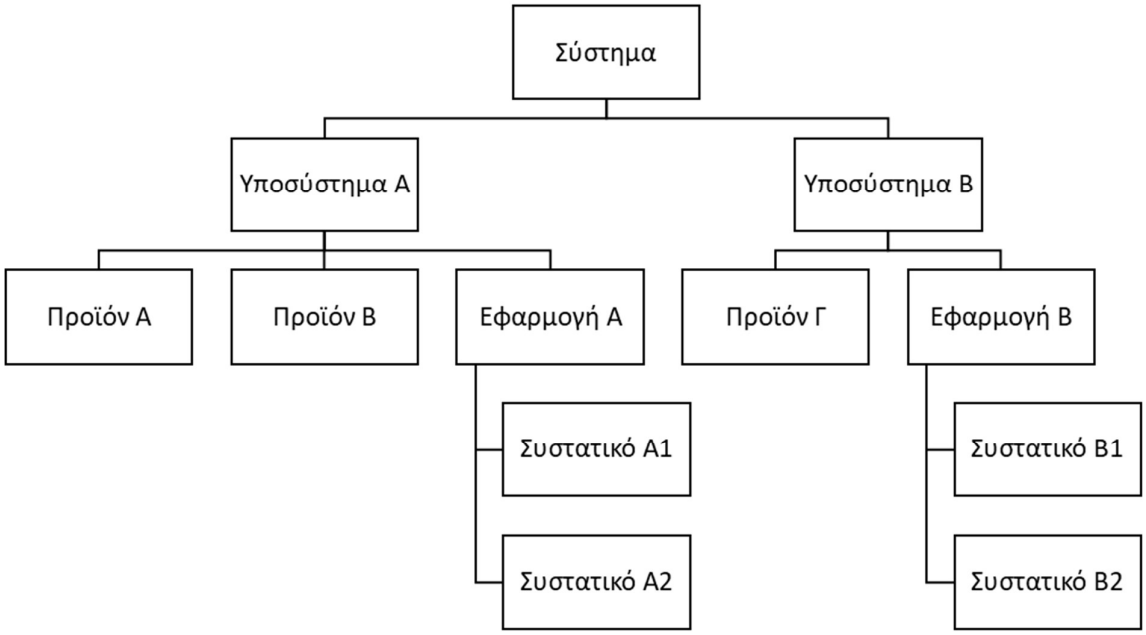
6.4.2 Συστήματα ελέγχου εκδόσεων και αποθετήρια κώδικα

Μια βασική διαδικασία για την ολοκληρωμένη διαχείριση του λογισμικού είναι η διαχείριση σχηματισμών (configuration management). Καλούμε σχηματισμό λογισμικού (software configuration) το σύνολο όλων των συστατικών στοιχείων λογισμικού που κατασκευάζονται στις διάφορες φάσεις της ανάπτυξης και συγκροτούν το σύστημα. Τα συστατικά στοιχεία μπορεί να είναι είτε σε ολόκληρα προϊόντα είτε σε τμήματα αυτών. Κάθε τέτοιο ξεχωριστό στοιχείο ή μια συλλογή από στοιχεία που για διαχειριστικούς λόγους τα θεωρούμε ως μία μονάδα, καλείται Στοιχείο Σχηματισμού Λογισμικού. Επομένως, στοιχείο σχηματισμού λογισμικού (Configurable Item - CI) είναι ένα ατομικό συστατικό στοιχείο λογισμικού ή μια συλλογή από τέτοια στοιχεία τα οποία σχετίζονται μεταξύ τους, ώστε να συγκροτούν μια διαχειριστική οντότητα (Βεσκούκης, 2000).

Έτσι, αν ένα στοιχείο είναι μια συλλογή από άλλα στοιχεία, ονομάζεται σύνθετο, ενώ διαφορετικά ονομάζεται απλό ή ατομικό. Τα στοιχεία σχηματισμών λογισμικού μπορεί να είναι ιεραρχικά δομημένα. Μια τέτοια ιεραρχία φαίνεται στην Εικόνα 6.15. Προφανώς, η διαχείριση ενός στοιχείου σχηματισμού λογισμικού είναι απλούστερη, όταν ανήκει στα χαμηλότερα επίπεδα της ιεραρχίας.

Η διαχείριση σχηματισμών (Humphrey, 1989) είναι μια διαδικασία καταγραφής και τεκμηρίωσης των αλλαγών του συστήματος με τη χρήση γραμμών αναφοράς (baselines), η οποία:

- Παρακολουθεί και καταγράφει τα λειτουργικά και φυσικά χαρακτηριστικά ενός αντικειμένου και ενός συστήματος, καθώς αυτό εξελίσσεται μέσα στον χρόνο.
- Ελέγχει και καταγράφει τις αλλαγές αυτών των χαρακτηριστικών μέσα στον χρόνο.
- Μπορεί να παρουσιάσει ανά πάσα στιγμή και με λεπτομέρεια την ιστορία εξέλιξης του συστήματος, καθώς και την παρούσα κατάσταση.
- Ελέγχει ότι όλα τα επιμέρους στοιχεία ενός συστήματος ακολουθούν τους κανόνες.



***Εικόνα 6.15**: Η ιεραρχία των συστατικών ενός συστήματος λογισμικού*

Οι μεταβολές που πραγματοποιούνται στο λογισμικό κατά τον κύκλο ζωής του τελικά υλοποιούνται στο επίπεδο των συστατικών λογισμικού, που αποτελεί το κατώτερο επίπεδο του δέντρου διαμόρφωσης. Κάθε μεταβολή σε κάποιο συστατικό δεν καταργεί απαραίτητα την προηγούμενη μορφή αυτού, αλλά ενδέχεται και να συνυπάρχει με αυτή. Για παράδειγμα, η τροποποίηση ενός συστατικού στοιχείου της διπροσωπίας με τον χρήστη για μια άλλη γλώσσα, δεν καταργεί το στοιχείο που τροποποίησε, αλλά συνυπάρχει με αυτό. Μπορούμε λοιπόν γενικά να θεωρούμε ότι οι μεταβολές που συμβαίνουν στα συστατικά στοιχεία λογισμικού έχουν ως αποτέλεσμα αυτά να υπάρχουν σε πολλές μορφές του συστήματος, οι οποίες ονομάζονται εκδόσεις (versions).

Η συνολική κατάσταση του συστήματος που υπόκειται σε διαχείριση σχηματισμών συνήθως προσδιορίζεται από έναν αριθμό της μορφής N.X.Y, όπου το N προσδιορίζει τη βασική αποδέσμευση (major release) ή έκδοση (version), το X την υπο-αποδέσμευση (minor release), ενώ το Y την ενδιάμεση αποδέσμευση (interim release). Μια βασική αποδέσμευση προσδιορίζει μια νέα έκδοση του συστήματος, μια υπο-αποδέσμευση προσδιορίζει μια έκδοση του συστήματος με νέα βελτιωμένα χαρακτηριστικά, ενώ μια ενδιάμεση αποδέσμευση μια έκδοση του συστήματος με διορθωμένη λειτουργικότητα. (Φιτσιλής et al., 2008).

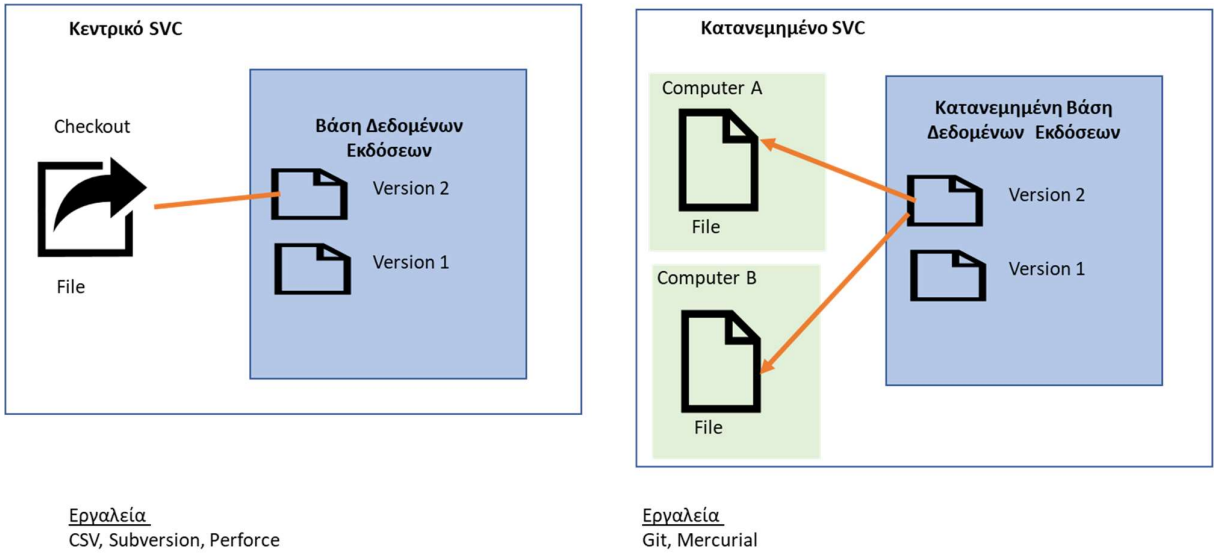
Εξαιτίας της πολυπλοκότητας της διαχείρισης των σχηματισμών λογισμικού η όλη διαδικασία γίνεται με τη χρήση ειδικών εργαλείων τα οποία επιτρέπουν την αυτοματοποίηση της διαδικασίας. Τα εργαλεία αυτά ονομάζονται Εργαλεία Ελέγχου Εκδόσεων (Software Version Control - SVC). Παραδείγματα τέτοιων εργαλεία είναι το Git, CVS, Subversion, Mercurial, Perforce, κ.λπ.

Ένα τέτοιο σύστημα επιτρέπει:

- Την καταγραφή και τη διαχείριση όλων των στοιχείων σχηματισμού (Configuration items - CIs),
- Δίνει τη δυνατότητα καταγραφής των χαρακτηριστικών του κάθε CI,
- Αποτυπώνει την ιεραρχία δόμησης του συστήματος,
- Επιτρέπει ελεγχόμενη πρόσβαση στη βάση CMDB (Configuration Management DB),

- Επιτρέπει την εισαγωγή ιστορικών στοιχείων, καθώς και
- Επιτρέπει τη δημιουργία αναφορών.

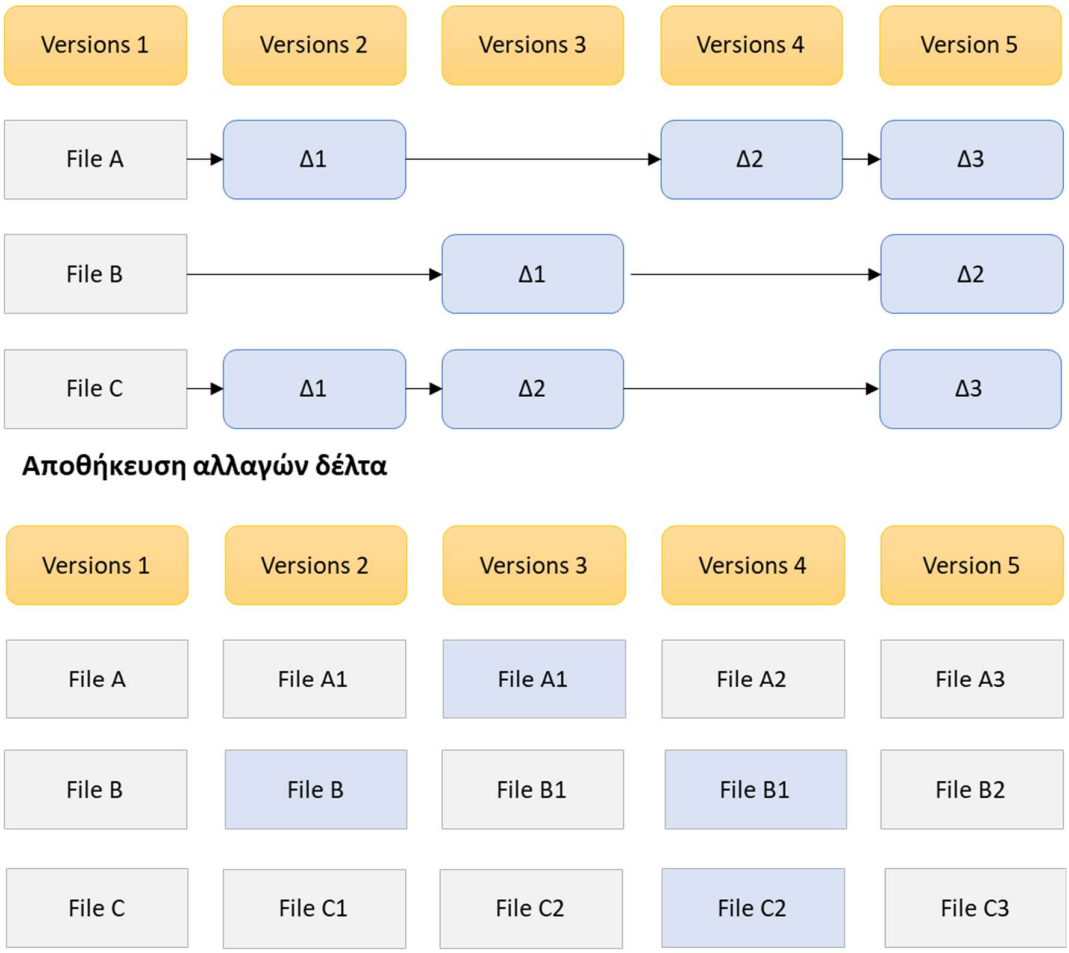
Τα συστήματα CVS ανάλογα με την αρχιτεκτονική τους, χαρακτηρίζονται ως κεντρικά συστήματα ελέγχου εκδόσεων (Centralized Version Control Systems - CVCSs) όπου γίνεται χρήση ενός κεντρικού εξυπηρετητή (π.χ. CVS, Subversion, Repforce) ο οποίος αποθηκεύει όλα τα αρχεία και ως κατανεμημένα συστήματα ελέγχου εκδόσεων (DVCS). Σε ένα σύστημα DVCS όπως το Git, το Mercurial, κ.λπ., οι υπολογιστές των χρηστών έχουν ένα αντίγραφο των αρχείων του κεντρικού αποθετηρίου, συμπεριλαμβανομένου του πλήρους ιστορικού του. Έτσι, εάν οποιοσδήποτε διακομιστής βγει εκτός λειτουργίας, οποιοδήποτε από τα τοπικά αποθετήρια πελάτη μπορεί να αντιγραφεί ξανά στον διακομιστή για την επαναφορά του, αφού κάθε κλώνος είναι πραγματικά ένα πλήρες αντίγραφο ασφαλείας όλων των δεδομένων. Οι εναλλακτικές αρχιτεκτονικές απεικονίζονται στην Εικόνα 6.16.



***Εικόνα 6.16**: Κεντρικά και κατανεμημένα συστήματα ελέγχου εκδόσεων*

Εννοιολογικά, τα περισσότερα συστήματα ελέγχου εκδόσεων, αποθηκεύουν τις αλλαγές ως μια λίστα αλλαγών. Αυτά τα άλλα συστήματα (CVS, Subversion, Perforce, κ.λπ.) θεωρούν κάθε τροποποίηση ως μια αλλαγή επί του αρχείου και αποθηκεύουν μόνο την αλλαγή με σκοπό τη βελτιστοποίηση του αποθηκευτικού χώρου. Το τελικό αρχείο παράγεται από το αρχείο βάσης (baseline) αφού εφαρμόσουμε τις αλλαγές (delta changes) που έχουν συμβεί.

Συστήματα όπως το Git ακολουθούν άλλη λογική: κάθε φορά που ένας προγραμματιστής αποθηκεύει μια αλλαγή, αποθηκεύεται εκ νέου ένα στιγμιότυπο του συστήματος συνολικά. Για λόγους βελτιστοποίησης, εάν ένα τεχνούργημα (π.χ. αρχείο) δεν έχει αλλάξει, αποθηκεύουμε ένα δείκτη προς αυτό το αρχείο (Chacon & Straub, 2014) (βλέπε Εικόνα 6.17).



.

Εικόνα 6.17*: Διαφορετικές στρατηγικές αποθήκευσης σε συστήματα ελέγχου εκδόσεων*

Αντίστοιχα, ένα αποθετήριο είναι ένας συνεργατικός ιστότοπος φιλοξενίας κώδικα όπου ο χρήστης/προγραμματιστής μπορεί να δημιουργήσει ένα σύστημα ελέγχου εκδόσεων. Παραδείγματα τέτοιων συστημάτων είναι τα συστήματα GitHub (https://github.com/), το σύστημα bitbucket (https://bitbucket.org/), κ.λπ.

Στα αποθετήρια όπως το GitHub υπάρχει η δυνατότητα δημιουργίας ατομικού ή συνεργατικού συστήματος ελέγχου εκδόσεων. Για παράδειγμα το 2002 το GitHub ανέφερε ότι είχε πάνω από 83 εκατομμύρια προγραμματιστές και περισσότερα από 200 εκατομμύρια αποθετήρια, συμπεριλαμβανομένων τουλάχιστον 28 εκατομμυρίων δημόσιων αποθετηρίων, το οποίο αποτελεί τον μεγαλύτερο κεντρικό υπολογιστή πηγαίου κώδικα από τον Νοέμβριο του 2021 (https://en.wikipedia.org/wiki/GitHub).

Σε ένα τέτοιο αποθετήριο κώδικα όπως το GitHub υποστηρίζεται μεγάλος αριθμός λειτουργιών που αυτοματοποιούν διεργασίες όπως:

- Λειτουργία "fork & pull" όπου οι προγραμματιστές δημιουργούν το δικό τους αντίγραφο ενός αποθετηρίου και υποβάλλουν ένα αίτημα έλξης όταν θέλουν ο διαχειριστής του έργου να «τραβήξει» τις αλλαγές τους στον κύριο κλάδο (baseline) του κώδικα.
- Συνεργατική αναθεώρηση κώδικα.
- Παρακολούθηση προβλημάτων λογισμικού.

- Υποστήριξη κοινωνικής δικτύωσης, όπου χρήστες μπορούν να εγγραφούν «παρακολουθώντας» έργα και «ακολουθώντας» άλλους χρήστες.

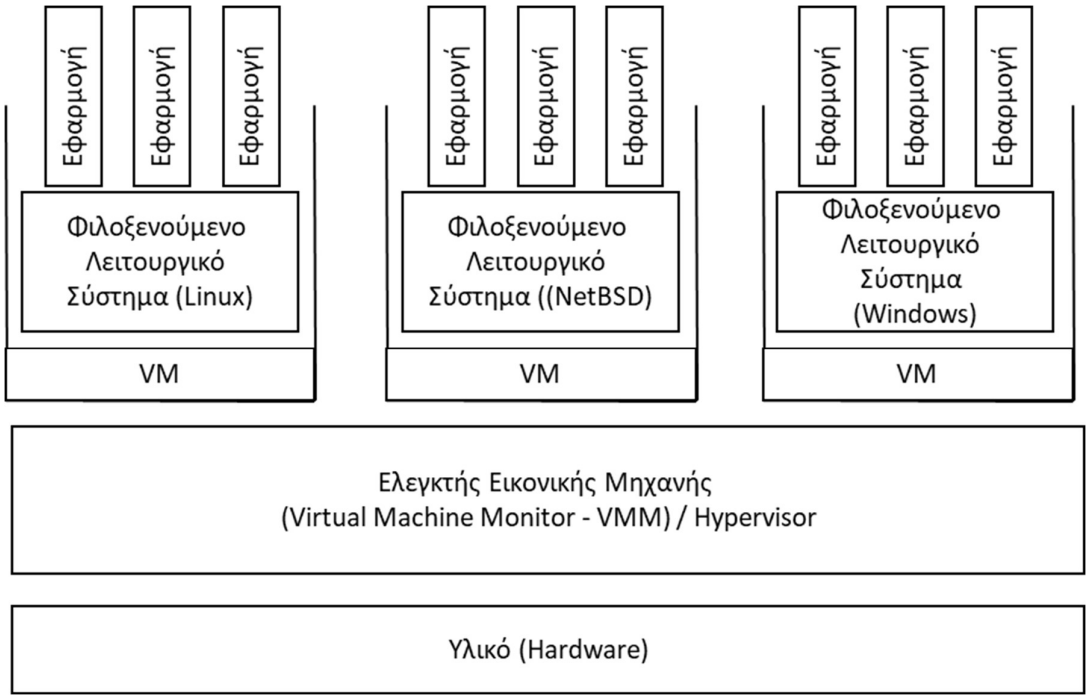
6.4.3 Containers (Περιέκτες)

Οι containers (περιέκτες) είναι ένα αρχιτεκτονικό μοτίβο που υλοποιεί μια μορφή εικονικοποίησης (virtualization). Όπως υποδηλώνει η ονομασία τους είναι "κουτιά – δοχεία" τα οποία έχουν συγκεκριμένες ιδιότητες, μέσα στα οποία, όπως θα δούμε, εκτελούνται οι εφαρμογές.

Τα containers αλλάζουν ριζικά τον τρόπο που αναπτύσσουμε, διανέμουμε και εκτελούμε το λογισμικό, διότι η ανάπτυξη του λογισμικού αποδεσμεύεται από το περιβάλλον λειτουργίας. Έτσι οι προγραμματιστές μπορούν να δημιουργούν λογισμικό με όποιον τρόπο επιθυμούν, γνωρίζοντας εξ' αρχής ότι θα εκτελείται πανομοιότυπα, ανεξάρτητα από το περιβάλλον λειτουργίας, είτε πρόκειται για ένα εξυπηρετητή, είτε για έναν φορητό υπολογιστή ή είτε για το ΥΝ. Με τον τρόπο αυτό μπορούν να επικεντρωθούν στην ανάπτυξη του λογισμικού, αντί να αφιερώνουν χρόνο στη διαμόρφωση του περιβάλλοντος λειτουργίας ή να αναγνωρίζουν και να συντηρούν τις εξαρτήσεις του συστήματος (Mouat, 2015). Τα containers είναι, λοιπόν, μια ενθυλάκωση μιας εφαρμογής με τις εξαρτήσεις της (application encapsulation) με σκοπό την ανεξάρτητη εκτέλεση.

Οι προγραμματιστές συνήθως χρησιμοποιούν container για να επιλύσουν δύο κύρια προβλήματα με τη φιλοξενία των εφαρμογών. Το πρώτο είναι ότι οι προγραμματιστές συχνά δυσκολεύονται να κάνουν τις εφαρμογές να εκτελούνται με συνέπεια σε διαφορετικά περιβάλλοντα φιλοξενίας. Ακόμη και όταν το βασικό λειτουργικό σύστημα των συστημάτων είναι το ίδιο, μικρές διαφορές στη διαμόρφωση αυτών των συστημάτων σε υλικό ή λογισμικό μπορεί να οδηγήσουν σε απροσδόκητες διαφορές στη συμπεριφορά, προκαλώντας, για παράδειγμα, την εμφάνιση ζητημάτων στην παραγωγή που δεν ήταν εμφανή κατά την ανάπτυξη του συστήματος. Η δημιουργία container για μια εφαρμογή λύνει αυτό το πρόβλημα παρέχοντας ένα συνεπές και τυποποιημένο περιβάλλον για την εκτέλεση αυτής της εφαρμογής. Το δεύτερο πρόβλημα είναι ότι, αν μια φιλοξενούμενη εφαρμογή πρέπει να απομονωθεί από τις άλλες για να εκτελείται με ασφάλεια και αξιοπιστία, η επίτευξη αυτής της απομόνωσης με φυσικούς διακομιστές είναι έντασης υπολογιστικών πόρων.

Για να γίνει κατανοητή η έννοια και η δομή ενός container, θα πρέπει να παρουσιάσουμε την έννοια της εικονικοποίησης. Η αρχιτεκτονική με χρήση εικονικών μηχανών αποσυνδέει την έννοια του εξυπηρετητή από τη φυσική μηχανή στην οποία είναι εγκατεστημένος και εκτελείται. Δηλαδή, η εικονικοποίηση αναφέρεται στην ύπαρξη εικονικών μηχανών στο ίδιο φυσικό μηχάνημα, επιτρέποντας την παράλληλη εκτέλεση πολλαπλών εξυπηρετητών και λειτουργικών συστημάτων στον ίδιο φυσικό υπολογιστή. Στην εικονικοποίηση, οι εικονικές μηχανές μπορούν να μεταφέρονται δυναμικά από τον ένα φυσικό υπολογιστή σε έναν άλλον, με αποτέλεσμα την αξιοποίηση στο έπακρο των διαθέσιμων φυσικών υπολογιστικών πόρων. Ταυτόχρονα, η εικονικοποίηση αφαιρεί τις τεχνικές πληροφορίες υλοποίησης του συστήματος από τους τελικούς χρήστες και τους προγραμματιστές. Οι εφαρμογές τρέχουν σε φυσικά συστήματα τα οποία δεν έχουν καθοριστεί, τα δεδομένα αποθηκεύονται σε άγνωστες γεωγραφικές τοποθεσίες, η διαχείριση των συστημάτων μεταβιβάζεται σε τρίτους και η πρόσβαση από τους τελικούς χρήστες είναι ευρέως διαδεδομένη. Μια συνηθισμένη τέτοιου είδους αρχιτεκτονική παρουσιάζεται στην Εικόνα 6.18.

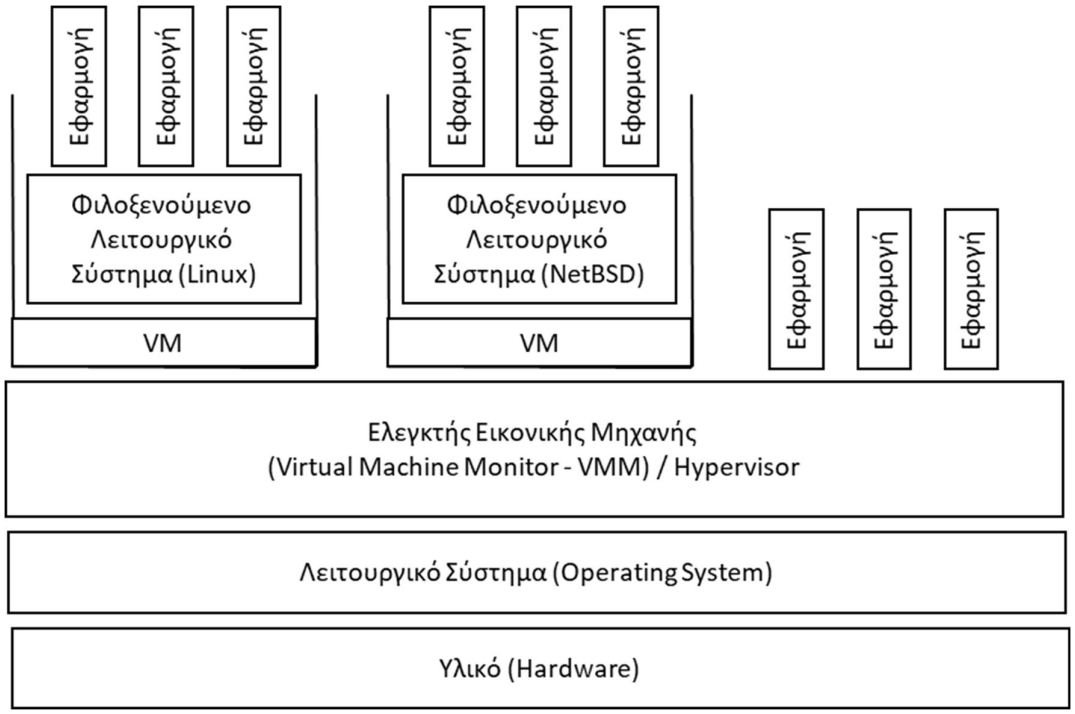


***Εικόνα 6.18**: Αρχιτεκτονική εικονικών μηχανών με hypervisor τύπου 1*

Ένας hypervisor είναι ένα επίπεδο λογισμικού που βρίσκεται ενδιάμεσα από το υλικό (μηχανικά μέρη) και τις εικονικές μηχανές (Virtual Machines - VM). Χωρίς την ύπαρξη του hypervisor, ένα λειτουργικό σύστημα θα επικοινωνούσε απευθείας με το υλικό του συστήματος. Συνεπώς, όλες οι εφαρμογές θα μπορούσαν να διαχειριστούν άμεσα από το υποσύστημα του δίσκου, τη φυσική μνήμη του συστήματος κ.λπ. Αυτό σημαίνει ότι θα υπήρχαν πολλά προβλήματα συγχρονισμού και διαμοιρασμού των υπολογιστικών πόρων με καταστροφικές συνέπειες. Το πρόβλημα αυτό καλείται να λύσει ο hypervisor, ο οποίος συντονίζει και οργανώνει τις αλληλεπιδράσεις μεταξύ της κάθε εικονικής μηχανής και του υλικού που όλες οι συνδεδεμένες εφαρμογές μοιράζονται.

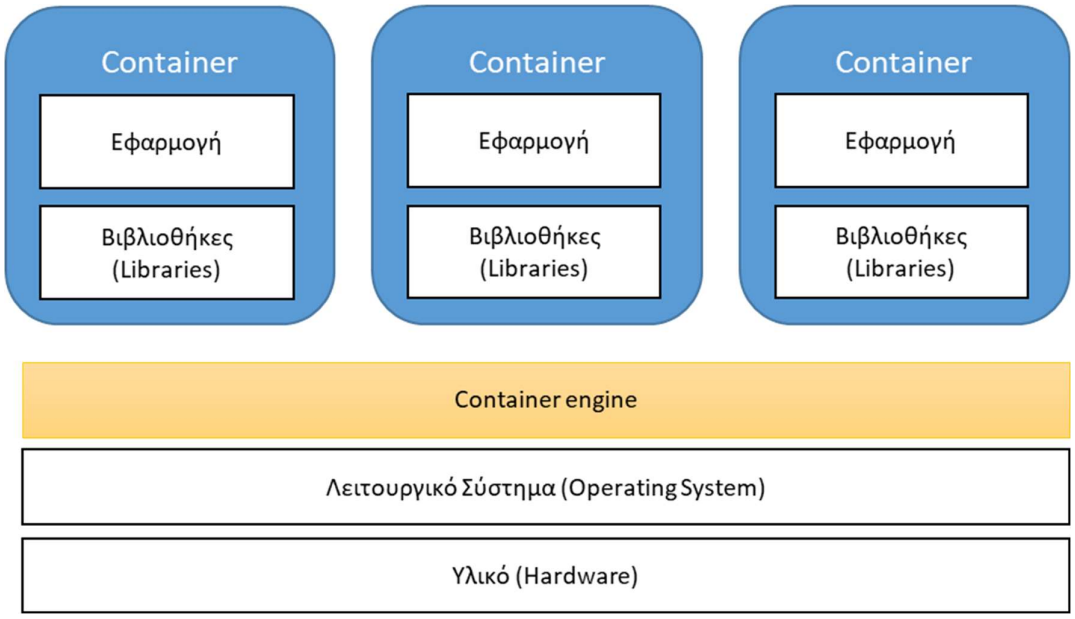
Υπάρχουν δύο κατηγορίες hypervisor: ο hypervisor τύπου 1, που χρησιμοποιείται κυρίως σε εικονικοποίηση διακομιστών, όπως είναι το παράδειγμα της Εικόνας 2.10, και ο hypervisor τύπου 2, που χρησιμοποιείται σε εικονικοποίηση επιφάνειας εργασίας. Όπως φαίνεται και από την Εικόνα 2.10, ο hypervisor τύπου 1 εκτελείται κατευθείαν στο υλικό του server χωρίς να υπάρχει λειτουργικό σύστημα ενδιάμεσα. Επίσης, παρέχει επιπλέον ασφάλεια, διότι, αν υπάρχει πρόβλημα σε κάποια εικονική μηχανή (Virtual Machine - VM), αυτή μπορεί να βλάψει μόνο τον εαυτό της, αφού το όποιο πρόβλημα δεν ξεφεύγει από τα όρια της VM.

Ένας hypervisor τύπου 2 είναι ένας hypervisor που είναι εγκατεστημένος πάνω στο λειτουργικό σύστημα (βλέπε Εικόνα 6.19). Οι hypervisor τύπου 2 υποστηρίζουν εικονικές μηχανές, συντονίζοντας/συγχρονίζοντας τα αιτήματα προς τη CPU, τη μνήμη, τον δίσκο, το δίκτυο και τους άλλους υπολογιστικούς πόρους μέσω του λειτουργικού συστήματος του φυσικού κεντρικού υπολογιστή, γεγονός που διευκολύνει τον τελικό χρήστη να λειτουργεί μια εικονική μηχανή σε έναν προσωπικό υπολογιστή.



***Εικόνα 6.19**: Αρχιτεκτονική εικονικών μηχανών με hypervisor τύπου 2*

Η container based virtualization αρχιτεκτονική που υπάρχει στα συστήματα Unix εδώ και πολλά χρόνια, επανήλθε στο προσκήνιο και έγινε ιδιαίτερα δημοφιλής με την εξάπλωση των συστημάτων ΥΝ, διότι βοηθά στο να αποφύγουμε την πτώση της απόδοσης που προσθέτει ο Hypervisor σε κάθε virtual machine. Η λογική είναι ότι όλα τα containers χρησιμοποιούν τον ίδιο kernel (τις περισσότερες φορές του Linux) με αποτέλεσμα να εμφανίζουν καλύτερη απόδοση, αν και προφανώς θυσιάζεται η ευελιξία της χρήσης διαφορετικών λειτουργικών στο ίδιο υλικό (βλέπε Εικόνα 6.20).



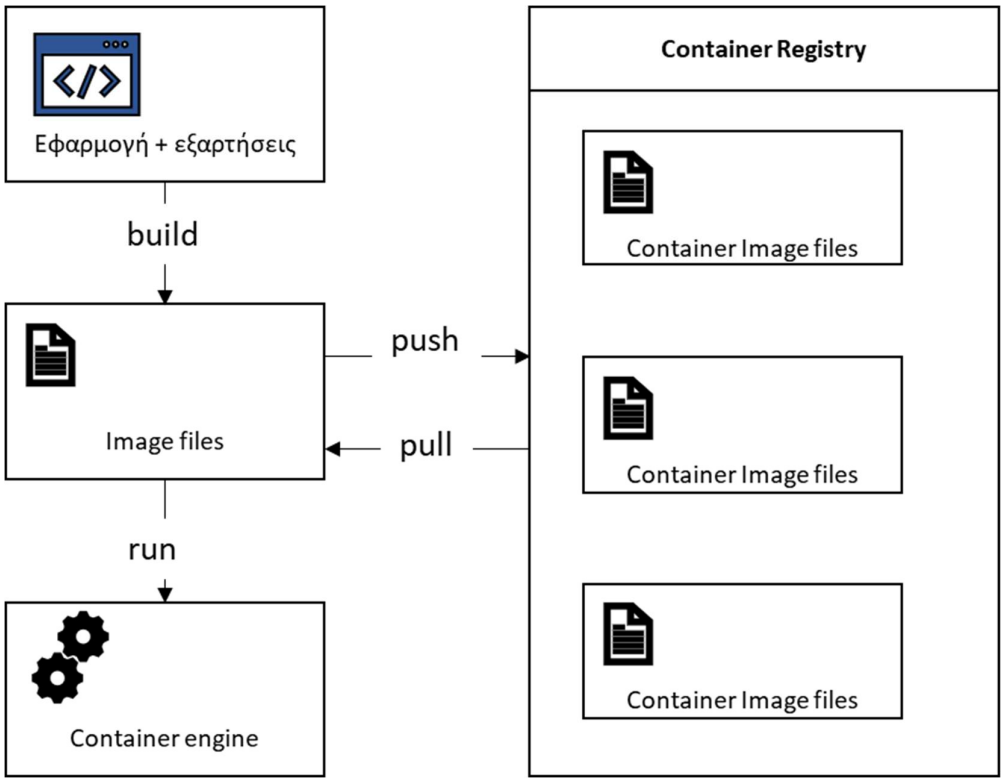
***Εικόνα 6.20**: Αρχιτεκτονική container based virtualization*

Τα containers περιέχουν τα απαραίτητα στοιχεία για την εκτέλεση του επιθυμητού λογισμικού. Αυτά τα στοιχεία περιλαμβάνουν τα αρχεία, τις μεταβλητές περιβάλλοντος, τις εξαρτήσεις και τις βιβλιοθήκες. Το λειτουργικό σύστημα κεντρικού υπολογιστή περιορίζει την πρόσβαση του container σε φυσικούς πόρους, όπως τη CPU, το σκληρό δίσκο, τη μνήμη, κ.λπ. και επομένως ένα μεμονωμένο container δεν μπορεί να καταναλώσει όλους τους φυσικούς πόρους ενός κεντρικού υπολογιστή.

Όταν ένα container δεν εκτελείται, υπάρχει μόνο ως αποθηκευμένο αρχείο που ονομάζεται container image και περιέχει αρχεία που είναι πλήρεις, στατικές και εκτελέσιμες εκδόσεις μιας εφαρμογής ή μιας υπηρεσίας που μπορεί να διαφέρουν από τη μια πλατφόρμα στην άλλη. Αυτά τα image αρχεία αποτελούνται από πολλαπλά επίπεδα, τα οποία ξεκινούν με μια βασική εικόνα που περιλαμβάνει όλες τις εξαρτήσεις που απαιτούνται για την εκτέλεση κώδικα σε ένα container. Κάθε image μπορεί να χρησιμοποιηθεί για τη δημιουργία πολλών στιγμιότυπων και για αυτόν τον λόγο, ένα image μπορεί να θεωρηθεί ως σχέδιο (template) για containers. Ένα άλλο σημαντικό σημείο που πρέπει να τονιστεί σχετικά με τα images είναι ότι μπορούν να κοινοποιηθούν σε τρίτους μέσω ενός δημόσιου ή ιδιωτικού container registry. Για την προώθηση της κοινής χρήσης και τη μεγιστοποίηση της συμβατότητας μεταξύ διαφορετικών πλατφορμών και εργαλείων, τα images δημιουργούνται συνήθως με βάση το πρότυπο Open Container Initiative (OCI) (https://opencontainers.org/). Με τον τρόπο αυτό επιτυγχάνεται υψηλή φορητότητα, επειδή κάθε image περιλαμβάνει τις εξαρτήσεις που απαιτούνται για την εκτέλεση του κώδικα σε ένα container. Για παράδειγμα, οι χρήστες containers μπορούν να εκτελέσουν την ίδια εικόνα στο YN του Amazon Web Services (AWS) κατά τη διάρκεια μιας δοκιμής και στη συνέχεια, σε έναν διακομιστή εσωτερικής εγκατάστασης για παραγωγή, χωρίς να αλλάξουν τον κώδικα εφαρμογής στο container. Τέλος, μια εφαρμογή που χρησιμοποιεί containers ονομάζεται containerized application (βλέπε Εικόνα 6.21).

Το πιο δημοφιλές εργαλείο για τη δημιουργία container images συμβατών με OCI είναι το εργαλείο Docker (https://www.docker.com/).

Ένα container engine (μηχανή) είναι ένα λογισμικό που επιτρέπει σε ένα λειτουργικό σύστημα να λειτουργεί ως σύστημα εκτέλεσης container. Μια μηχανή container δέχεται εντολές χρήστη για τη δημιουργία, εκκίνηση και διαχείριση container μέσω γραφικής διεπαφής. Επίσης, παρέχει ένα API που επιτρέπει σε εξωτερικές εφαρμογές να υποβάλλουν παρόμοια αιτήματα. Το πιο βασικό υποσύστημα μιας μηχανής container, είναι το σύστημα εκτέλεσης (runtime). Το container runtime σύστημα είναι υπεύθυνο για τη δημιουργία της τυποποιημένης πλατφόρμας στην οποία μπορούν να εκτελούνται οι εφαρμογές, για τη λειτουργία των containers και για τον χειρισμό των αναγκών αποθήκευσης container στο τοπικό σύστημα. Επίσης πολλοί οργανισμοί χρησιμοποιούν έναν χρονοπρογραμματιστή container ή/και τεχνολογία ενορχήστρωσης (π.χ. Kubernetes) για τη διαχείριση των αποδεσμεύσεων και διατάξεων.



***Εικόνα 6.21**: Η δομή του container*

Τα containers αποτελούν μια ιδιαίτερα δημοφιλή λύση τα τελευταία χρόνια, ιδιαίτερα σε σχέση με την ανάπτυξη του ΥΝ και της προσέγγισης DevOps, αφού παρουσιάζουν αρκετά πλεονεκτήματα όπως:

- Εκτέλεση πολλαπλών αντιγράφων της ίδιας εφαρμογής στην ίδια υπολογιστική μονάδα. Το κάθε container είναι απομονωμένο από το λειτουργικό σύστημα και αυτή η απομόνωση των πόρων μας δίνει την δυνατότητα να εκτελέσουμε την ίδια εφαρμογή στην ίδια ή διαφορετική έκδοση πολλές φορές.
- Ευκολία μετακίνησης εφαρμογών σε άλλη υπολογιστική μονάδα διαφορετικών προδιαγραφών.
- Γρήγορη εκκίνηση των εφαρμογών.
- Μεγάλη δυνατότητα επεκτασιμότητας

6.4.4 Η υποδομή ως κώδικας (Infrastructure as Code)

Η υποδομή ως κώδικας είναι μια καινοτόμα προσέγγιση που αντιμετωπίζει την πληροφοριακή υποδομή, τα εργαλεία και τις υπηρεσίες που διαχειρίζονται την υποδομή, ως σύστημα λογισμικού, προσαρμόζοντας σε αυτά τις πρακτικές της μηχανικής λογισμικού.

Τα στοιχεία της πληροφοριακής υποδομής σήμερα αλλάζουν συνεχώς και αυτόματα. Για παράδειγμα, οι διακομιστές ενεργοποιούνται ή απενεργοποιούνται κατά περίπτωση, ώστε η χωρητικότητα να ταιριάζει με τη ζήτηση, την ανάνηψη από βλάβες ή την ενεργοποίηση νέων υπηρεσιών. Αυτές οι αλλαγές μπορεί να προέρχονται είτε από άτομα της ομάδας DevOps, είτε ως απόκριση σε συμβάντα (π.χ. αποτυχία υλικού), είτε για λόγους αυξημένης ζήτησης υπηρεσιών.

Οι στόχοι της διαχείρισης της υποδομής με κώδικα είναι πολλαπλοί (Morris, 2016):

- Η πληροφορική υποδομή θα πρέπει να επιτρέπει τις αλλαγές χωρίς να αποτελεί εμπόδιο ή περιορισμό.

- Η ομάδα DevOps θα πρέπει να αφιερώνει τον χρόνο της σε εργασίες που απαιτούν σκέψη, δημιουργικότητα, ικανότητες και τεχνογνωσία και όχι σε καθημερινές, επαναλαμβανόμενες εργασίες.
- Οι χρήστες θα πρέπει να είναι σε θέση να διαχειρίζονται τους υπολογιστικούς πόρους που χρειάζονται, χωρίς να χρειάζονται εξειδικευμένο προσωπικό πληροφορικής για να το κάνουν (self-service).
- Οι αλλαγές στο πληροφοριακό σύστημα θα πρέπει να είναι αλλαγές ρουτίνας, και να μην προκαλούν σφάλματα ή διακοπές λειτουργίας.
- Οι βελτιώσεις πρέπει να είναι συνεχείς και μικρές και όχι μεγάλου μεγέθους που να θέτουν σε κίνδυνο την λειτουργία του συστήματος.
- Οι λύσεις στα προβλήματα θα πρέπει να εφαρμόζονται και να δοκιμάζονται και όχι να ξοδεύεται ο χρόνος της ομάδας σε ατέρμονες συζητήσεις και σε ανταλλαγή εκτενών εγγράφων.

Για την επίτευξη της διαχείρισης της υποδομής με λογισμικό θα πρέπει να βασιστούμε σε εννέα βασικές αρχές. Αυτές είναι:

- Η υποδομή θα πρέπει να μπορεί να αναπαραχθεί από το λογισμικό (reproducibility). Θα πρέπει να είναι δυνατή η αβίαστη και αξιόπιστη ανακατασκευή οποιουδήποτε στοιχείου μιας υποδομής.
- Όπως έχουμε ήδη αναφέρει, η εξασφάλιση της ικανότητας μιας επιχείρησης, σε περίπτωση καταστροφής, να επαναφέρει τις υπηρεσίες των πληροφοριακών συστημάτων σε ένα συγκεκριμένο επίπεδο λειτουργίας, το οποίο θα καλύπτει τις ελάχιστες απαραίτητες επιχειρησιακές λειτουργίες ορίζεται ως «συνέχεια της υπηρεσίας» (service continuity)
- Θα πρέπει να υπάρχει δυνατότητα αυτοελέγχου της υποδομής με την εκτέλεση καταλλήλων σεναρίων.
- Η τεκμηρίωση του προγράμματος θα πρέπει να παράγεται αυτόματα από τον υπάρχοντα κώδικα (self-documenting systems).
- Οι όποιες αλλαγές γίνονται στην πληροφοριακή υποδομή θα πρέπει να είναι σταδιακές και μικρές ώστε να μην επηρεάζουν την ομαλή λειτουργία του συστήματος, να είναι εύκολο να ελεγχθούν, κ.λπ.
- Όλα τα στοιχεία της υποδομής θα πρέπει να είναι υπό τον έλεγχο του συστήματος διαχείρισης σχηματισμών (configuration management).
- Θα πρέπει να υπάρχει συνέπεια (consistency) στη διαμόρφωση των στοιχείων της πληροφοριακής υποδομής. Για παράδειγμα, δύο εξυπηρετητές με κοινό σκοπό να διαμορφώνονται με τον ίδιο τρόπο.
- Θα πρέπει να υπάρχει δυνατότητα επαναληψιμότητας (repeatability) των ενεργειών, ειδικά για όσες εργασίες είναι περιοδικές.
- Θα πρέπει να κάνουμε την παραδοχή εργασίας ότι οποιοδήποτε στοιχείο της πληροφοριακής υποδομής μπορεί να καταστραφεί ανά πάσα στιγμή, χωρίς προειδοποίηση, όταν το υλικό αποτύχει ή όταν υπάρχει ανάγκη για μείωση της υπολογιστικής δυναμικότητας ή απλά αντικατασταθεί με άλλο στοιχείο νέας γενιάς (disposability). Θα πρέπει να σχεδιάζουμε την ψηφιακή υπηρεσία με τέτοιο τρόπο, ώστε η υποδομή να μπορεί να αντικατασταθεί απρόσκοπτα, ακόμη και με το σύστημα σε πλήρη λειτουργία (hot swap).

Υπάρχουν πολλά εργαλεία που υποστηρίζουν την προσέγγιση «υποδομή ως κώδικας», όπως τα εργαλεία Puppet (https://puppet.com/), το Chef (https://www.chef.io/), Terraform (https://www.terraform.io/), Ansimble (https://www.ansible.com/), κ.α.

6.5 Η κουλτούρα DevOps

Η ανάπτυξη λογισμικού με την προσέγγιση DevOps είναι μια σύνθετη διεργασία που παρά την πολυπλοκότητά της έχει ένα απλό στόχο: να προσφέρει την αξία όπου χρειάζεται ο πελάτης.

Συνεπώς, ο πρωταρχικός σκοπός της προσέγγισης DevOps είναι η βελτιστοποίηση της ροής αξίας από τη σύλληψη της αρχικής ιδέα ως την παράδοση της ψηφιακής υπηρεσίας στον τελικό χρήστη. Όπως τονίσαμε και στην εισαγωγή, στόχος του DevOps δεν είναι μόνο η ανάπτυξη λογισμικού, αλλά η συνεχής παράδοση λογισμικού στον πελάτη. Τελικά, αυτό που παραδίδουμε στον πελάτη δεν είναι λογισμικό αλλά μια ψηφιακή υπηρεσία, η οποία μεταβάλλεται, αλλάζει, λειτουργεί αδιάλειπτα χωρίς προβλήματα. Επιπλέον, ο πελάτης δεν γίνεται μέτοχος των σύνθετων διεργασιών και των πιθανών προβλημάτων που παρουσιάζονται κατά τη διάρκεια αυτής της αυτοματοποιημένης ροής εργασίας.

Για να επιτύχουμε τα παραπάνω, θα πρέπει να αλλάξει η κουλτούρα της επιχείρησης, αυτή που εφαρμόζει την προσέγγιση DevOps. Οι βασικές αρχές της κουλτούρας DevOps είναι (Google, 2022; Davis & Daniels, 2016). :

- Ικανοποίηση από την εργασία
- Οργανωσιακή κουλτούρα Westrum
- Κουλτούρα συνεχούς μάθησης
- Κουλτούρα μετασχηματισμού

Η έρευνα που πραγματοποιήθηκε από την ερευνητική ομάδα DORA (Google, 2022) διαπίστωσε ότι η **ικανοποίηση ενός εργαζομένου** από την εργασία του σχετίζεται άμεσα με την απόδοση του οργανισμού. Συνεπώς, έχοντας αφοσιωμένους υπαλλήλους που κάνουν ουσιαστική δουλειά, οδηγεί σε βελτίωση της απόδοσης και στη δημιουργία επιχειρηματικής αξίας. Μια εργασία που είναι δημιουργική, με νόημα, που αξιοποιεί τις δεξιότητες, τις ικανότητες και την κρίση του εργαζομένου δημιουργεί ικανοποίηση. Επιπλέον, υπάρχει ένας ενάρετος κύκλος όσον αφορά τα οφέλη της εργασιακής ικανοποίησης. Οι άνθρωποι δουλεύουν καλύτερα όταν αισθάνονται την υποστήριξη των εργοδοτών τους, όταν έχουν τα εργαλεία και τους πόρους για να κάνουν τη δουλειά τους και όταν αισθάνονται ότι η εργασία τους εκτιμάται. Αυτός ο κύκλος συνεχούς βελτίωσης και μάθησης είναι αυτό που ξεχωρίζει τις επιτυχημένες εταιρείες από τις αποτυχημένες, δίνοντάς τους τη δυνατότητα να καινοτομούν, να προηγούνται του ανταγωνισμού και να κερδίζουν πελάτες.

Το DevOps βελτιώνει την ικανοποίηση από την εργασία διότι:

- οι εργαζόμενοι έχουν ουσιαστική εργασία στην οποία μπορούν να συνεισφέρουν,
- προωθεί τη συνεχή βελτίωση και τον πειραματισμό,
- προωθεί τη συνεχή μάθηση,
- οι γραφειοκρατικές διαδικασίες ελαχιστοποιούνται,
- βελτιώνει το ομαδικό πνεύμα,
- προωθεί το πνεύμα υψηλής απόδοσης,
- παρέχει στους εργαζόμενους τα εργαλεία και τους πόρους που απαιτούνται για την εργασία τους, και
- παρέχει στους υπαλλήλους ουσιαστική εργασία που αξιοποιεί την εμπειρία τους και τις γνώσεις τους.

Η δεύτερη αρχή στην οποία βασίζεται η κουλτούρα του Devops είναι **η οργανωσιακή κουλτούρα Westrum δημιουργικού τύπου** (Westrum, 2004). Ο Westrum μελέτησε τους ανθρώπινους παράγοντες που επηρεάζουν περιπτώσεις που σχετίζονταν με ατυχήματα σε τεχνολογικούς τομείς που είναι εξαιρετικά περίπλοκοι και επικίνδυνοι, όπως η αεροπορία και η υγειονομική περίθαλψη και ανέπτυξε μια τυπολογία οργανωσιακής κουλτούρας, η οποία αποτελείται από τρεις κατηγορίες κουλτούρας:

- Οι εργαζόμενοι στις παθολογικές οργανώσεις (pathological) είναι προσανατολισμένοι στην απόκτηση εξουσίας και χαρακτηρίζονται από τον φόβο και την απειλή. Οι εργαζόμενοι φοβούνται να εκφραστούν ελεύθερα, αποκρύπτουν τις πληροφορίες ή τις διαστρεβλώνουν, ώστε να μπορέσουν να παρουσιάσουν μια καλύτερη εικόνα για το αποτέλεσμα της εργασίας τους. Επιπλέον, κάθε αποτυχία τιμωρείται.
- Οι γραφειοκρατικές οργανώσεις (bureaucratic) είναι προσανατολισμένες στους κανόνες και προστατεύουν την κατάτμηση της εργασίας σε τμήματα. Το κάθε τμήμα διεκδικεί την ανεξαρτησία του, την απομόνωσή του από τα άλλα και την εφαρμογή των δικών τους κανόνων.
- Οι δημιουργικοί οργανισμοί (generative) είναι προσανατολισμένοι στην επίτευξη της βέλτιστης απόδοσης και επικεντρώνονται στην αποστολή τους.

Ο Πίνακας 6.5 παρουσιάζει τα βασικά χαρακτηριστικά της κάθε κατηγορίας κουλτούρας.

Παθολογική κουλτούρα	**Γραφειοκρατική κουλτούρα**	**Δημιουργική κουλτούρα**
Εστίαση στη δύναμη/ισχύ	Εστιασμένη σε κανόνες	Εστιασμένη στην υψηλή απόδοση
Μικρή δυνατότητα συνεργασίας	Μέτρια δυνατότητα συνεργασίας	Στενή συνεργασία
Αποποίηση ευθυνών από τη διοίκηση	Περιορισμένες ευθύνες	Οι κίνδυνοι επιμερίζονται στην ομάδα
Η συνεργασία με άλλα τμήματα αποθαρρύνεται	Η συνεργασία είναι ανεκτή	Ενθαρρύνεται η συνεργασία
Η αποτυχία οδηγεί στην εύρεση «αποδιοπομπαίων τράγων»	Η αποτυχία αξιολογείται με δικαιοσύνη	Η αποτυχία οδηγεί σε έρευνα για τους λόγους με σκοπό τη βελτίωση
Η καινοτομία δεν είναι επιθυμητή	Η καινοτομία οδηγεί σε προβλήματα	Η καινοτομία καλωσορίζεται και γίνεται προσπάθεια υλοποίησης

Πίνακας 6.5: *Οι τρεις διαφορετικές κατηγορίες κουλτούρας στην τυπολογία Westrum*

Η προσέγγιση DevOps στοχεύει στη δημιουργική κουλτούρα και έχει ως στόχο την αύξηση της συνεργασίας με διαλειτουργικές ομάδες που περιλαμβάνουν αναλυτές, προγραμματιστές, μηχανικούς ποιότητας, επιχειρήσεις, ειδικούς ασφάλειας, κ.λπ. Όλοι μοιράζονται την ίδια ευθύνη για την παροχή της ψηφιακής υπηρεσίας, από τον απλό εργαζόμενο μέχρι και το υψηλότερο επίπεδο διοίκησης. Η πολιτική που εφαρμόζεται είναι η ακόλουθη:

- Η διοίκηση στοχεύει στην ανάπτυξη μιας δημιουργικής κουλτούρας χωρίς φόβο, όπου οι εργαζόμενοι παρουσιάζουν τα προβλήματα καθώς και την επίλυσή τους. Γίνονται συζητήσεις σχετικά με το ποια ήταν η αιτία του προβλήματος.
- Οι εργαζόμενοι μοιράζονται τους κινδύνους και τις ευθύνες.
- Καταργεί τα οργανωτικά και τα τεχνολογικά σιλό, για παράδειγμα τον διαχωρισμό σε ομάδες ανάπτυξης λογισμικού και σε ομάδα λειτουργίας της ψηφιακής υπηρεσίας, δημιουργώντας μια ενιαία ομάδα DevOps
- Αφήνει την ομάδα να πειραματιστεί και να διερευνήσει νέες ιδέες που μπορεί να οδηγήσουν σε σημαντικές βελτιώσεις.

- Ενθαρρύνει και επιβραβεύει τις βελτιώσεις της διαδικασίας και τις ιδέες που βοηθούν στην προώθηση της συνεργασίας.

Έχει αποδειχτεί πολλαπλώς ότι οι ομάδες υψηλής απόδοσης χρειάζονται για να λειτουργήσουν κλίμα εμπιστοσύνης και ψυχολογικής ασφάλειας (Forsgren et al., 2019)

Η **συνεχής μάθηση** είναι το τρίτο βασικό συστατικό της κουλτούρας DevOps. Μια οργανωσιακή κουλτούρα που στοχεύει στη συνεχή μάθηση συμβάλλει σημαντικά στη βελτίωση της απόδοσης στην παράδοση λογισμικού. Μερικοί τρόποι που συμβάλουν στη συνεχή μάθηση περιλαμβάνουν:

- Τα μέλη της ομάδας θα πρέπει να έχουν διαθέσιμο χρόνο και πόρους για να ασχοληθούν με την άτυπη μάθηση και να μπορούν να εξερευνήσουν νέες ιδέες ή τεχνολογίες.
- Εάν η αποτυχία τιμωρείται, οι εργαζόμενοι δεν θα επιχειρούν να δοκιμάσουν νέες τεχνολογίες, αρχιτεκτονικές, εργαλεία κ.λπ.. Συνεπώς, η επιχείρηση θα πρέπει να αντιμετωπίζει τις αποτυχίες ως ευκαιρίες για μάθηση και θα πρέπει οι εργαζόμενοι να ενθαρρύνονται να αναλαμβάνουν εύλογα ρίσκα. Ο υπολογισμένος κίνδυνος είναι αναγκαίο συστατικό μιας κουλτούρας καινοτομίας.
- Η επιχείρηση θα πρέπει να δημιουργήσει ευκαιρίες και χώρους στους οποίους οι εργαζόμενοι μπορούν να μοιράζονται πληροφορίες. Για παράδειγμα, μπορούν να διοργανωθούν εβδομαδιαίες συναντήσεις προσωπικού, εβδομαδιαίες παρουσιάσεις νέων προϊόντων/τεχνολογιών, οργάνωση κοινωνικών εκδηλώσεων από την επιχείρηση, κ.λπ.
- Οι εργαζόμενοι θα πρέπει να ενθαρρύνονται να λάβουν μέρος σε συνέδρια, συναντήσεις εργασίας, να λαμβάνουν πιστοποιήσεις κ.λπ.
- Η επιχείρηση θα πρέπει να διαθέτει οικονομικούς και άλλους πόρους για συνεχή εκπαίδευση.

Το τέταρτο συστατικό της κουλτούρας DevOps είναι η **κουλτούρα μετασχηματισμού/αλλαγής** καθώς και η ηγεσία αλλαγής που θα πρέπει να υπάρχει. Ένας ηγέτης που θέλει να μετασχηματίσει την επιχείρηση θα πρέπει να έχει (Raffert & Griffin, 2004).:

- Όραμα και να κατανοεί με ξεκάθαρο τρόπο τους στόχους της ομάδας σε ένα εύλογο χρονικό ορίζοντα.
- Να διαθέτει ικανότητα εμπνευσμένης επικοινωνίας που να επιτυγχάνει να κάνει τους εργαζόμενους περήφανους που είναι μέλη της επιχείρησης.
- Να προκαλεί τα μέλη της ομάδας να σκεφτούν παλιά προβλήματα με νέους τρόπους και να επανεξετάσουν μερικές από τις βασικές υποθέσεις τους σχετικά με τη δουλειά τους.
- Να λαμβάνει υπόψη τα συναισθήματα και τις προσωπικές ανάγκες των άλλων πριν ενεργήσει.
- Να επαινεί τα μέλη της ομάδας όταν κάνουν μια δουλειά καλύτερη από το συνηθισμένο ή καλύτερα από τον μέσο όρο.

Αυτά τα πέντε χαρακτηριστικά συσχετίζονται σε μεγάλο βαθμό με τη βελτίωση της απόδοση της ομάδας DevOps.

Βιβλιογραφία/Αναφορές

Azure Pipelines. (2022, July 13). Microsoft Docs. Retrieved September 6, 2022, from https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/key-pipelines-concepts?view=azure-devops

Chacon, S., & Straub, B. (2014). *Pro git* (p. 456). Springer Nature.

Davis, J., & Daniels, R. (2016). Effective DevOps: building a culture of collaboration, affinity, and tooling at scale. " O'Reilly Media, Inc.".

McEwen, I. K. (2004). *Vitruvius: writing the body of architecture*. MIT press.

Fitsilis, P. (2021). DevOps basic concepts, culture and practices. https://smartdevops.eu

Fowler, M., & Foemmel, M. (2006). Continuous integration.

Fowler, M. (2006). Continuous Integration, Ανακτήθηκε 10η Αυγούστου, 2022 από https://martinfowler.com/articles/continuousIntegration.html.

Freeman, E. (2019). *DevOps For Dummies*. John Wiley & Sons.

Forsgren, N., Smith, D., Humble, J., & Frazelle, J. (2019). 2019 accelerate state of devops report.

Garvin, D, Edmondson, A., & Gino, G. (2008). Is Yours a Learning Organization? Ανακτήθηκε 5η Σεπτεμβρίου, 2022 από https://hbr.org/2008/03/is-yours-a-learning-organization.

Google (2022), *Cultural DevOps capabilities | DORA*, Ανακτήθηκε 5η Σεπτεμβρίου, 2022 από https://cloud.google.com/architecture/devops/culture

Hernantes, J., Ebert, C., Gallardo, G., & Serrano, N. (2016). Devops. *IEEE Software*, *33*(3), 94-100.

Humble, J., & Farley, D. (2010). *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education.

Humble, J., & Kim, G. (2018). *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations*. IT Revolution.

IEEE Computer Society. Software Engineering Technical Committee. (1983). *IEEE Standard Glossary of Software Engineering Terminology: An American National Standard*. IEEE.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. Ανακτήθηκε 22/06/2022 από https://csrc.nist.gov/publications/detail/sp/800-145/final

Morris, K. (2016). *Infrastructure as code: managing servers in the cloud*. " O'Reilly Media, Inc.".

Mouat, A. (2015). *Using docker: developing and deploying software with containers*. " O'Reilly Media, Inc.".

Nurdiani, I., Börstler, J., & Fricker, S. A. (2018). Literature review of flexibility attributes: A flexibility framework for software developing organization. *Journal of software: Evolution and Process*, *30*(9), e1937.

Rahman, A., Mahdavi-Hezaveh, R., & Williams, L. (2019). A systematic mapping study of infrastructure as code research. *Information and Software Technology*, 108, 65-77.

Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review. *Information and Software Technology*, *141*, 106700.

Rafferty, A. E., & Griffin, M. A. (2004). Dimensions of transformational leadership: Conceptual and empirical extensions. *The leadership quarterly*, *15*(3), 329-354.

Ribeiro, M., (2021). *Learning DevSecOps*. O'Reilly Media.

Sharma, S. (2017). *The DevOps adoption playbook: a guide to adopting DevOps in a multi-speed IT enterprise*. John Wiley & Sons.

Varanasi, B., & Belida, S. (2014). *Introducing Maven*. Apress.

Verona, J., Duffy, M., & Swartout, P. (2016). Learning DevOps: Continuously Deliver Better Software. Packt Publishing Ltd.

Westrum, R. (2004). A typology of organisational cultures. *BMJ Quality & Safety*, *13*(suppl 2), ii22-ii27.

Wilson, G. (2020). *DevSecOps: A Leader's Guide to Producing Secure Software Without Compromising Flow, Feedback and Continuous Improvement*. Rethink Press.

Βεσκούκης, Β. (2000). *Τεχνολογία Λογισμικού ΙΙ*. Ελληνικό Ανοικτό Πανεπιστήμιο.

Φιτσιλής, Π., Σταμέλος, Ι. & Ξένος, Μ. (2008). *Προγραμματισμός Έργων Πληροφορικής – Αντικειμενοστρεφείς Μεθοδολογίες.* Ελληνικό Ανοικτό Πανεπιστήμιο.

Φιτσιλής, Π. (2018). *Σύγχρονα πληροφοριακά συστήματα επιχειρήσεων (2η έκδοση).* Broken Hill Publishers.

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Με βάση τις μετρήσεις η διαθεσιμότητα ενός συστήματος για τον μήνα Αύγουστο ήταν 730 ώρες. Ποιο είναι το ποσοστό διαθεσιμότητας; Τι συμπεράσματα βγάζουμε αν η απαιτούμενη διαθεσιμότητα είναι 99,99%;

Απάντηση/Λύση

Για να αξιολογήσουμε τι θα συμβεί θα πρέπει να υπολογίσουμε τη διαθεσιμότητα του συστήματος με βάση τον τύπο που δίνεται. Έτσι,

$$Δ = 100 * \frac{Απαιτούμενος\ χρόνος\ λειτουργίας\ -\ Χρόνος\ δυσλειτουργίας}{Απαιτούμενος\ χρόνος\ λειτουργίας}$$

$$= 100 * \frac{744\ -(744-730)}{744} = 100 * \frac{730}{744} = 98{,}19\%$$

Για τις ψηφιακές υπηρεσίες απαιτείται διαθεσιμότητα 99,99 % που σημαίνει ότι:

Α) δεν έχουμε επιτύχει τον στόχο διαθεσιμότητας και

Β) το σύστημα μπορεί να δυσλειτουργεί για 4,5 λεπτά μηνιαίως. Ο χρόνος αυτός υπολογίζεται αν πολλαπλασιάσουμε τις ώρες του Αυγούστου με το 99,99%.

Συνεπώς, για τον μήνα Αύγουστο έχουμε υπερβεί κατά 13 ώρες και 55,5 λεπτά τη διαθεσιμότητα-στόχο.

Κριτήριο αξιολόγησης 2

Πότε επιβάλλεται η χρήση ενός αυτοματοποιημένου εργαλείου (π.χ. Maven) για την κατασκευή του συστήματος (build)

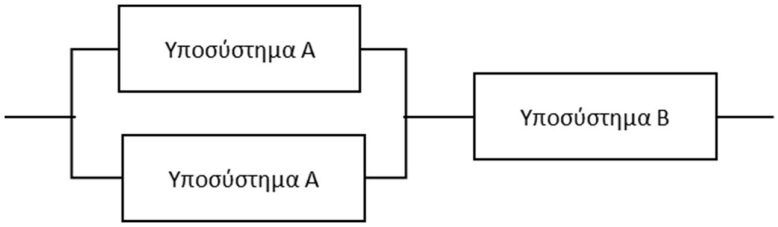
Απάντηση/Λύση

Ένα εργαλείο κατασκευής κώδικα όπως το Maven πρέπει να χρησιμοποιείται όταν συντρέχουν οι ακόλουθες συνθήκες:

- Όταν το έργο έχει μεγάλο αριθμό εξαρτήσεων. Με τη χρήση εργαλείων μπορούμε να διαχειριστούμε τις εξαρτήσεις πιο εύκολα.
- Όταν οι εκδόσεις των αναγκαίων συστατικών αλλάζουν συχνά. Για να ενημερώσουμε τις εξαρτήσεις, απλώς ενημερώνουμε το αναγνωριστικό έκδοσης στο αρχείο POM.
- Όταν υπάρχει ανάγκη για καθημερινή ή συχνότερη κατασκευή, ενσωμάτωση και έλεγχο.
- Όταν χρειαζόμαστε ένα γρήγορο τρόπο δημιουργίας τεκμηρίωσης από τον πηγαίο κώδικα.

Κριτήριο αξιολόγησης 3

Δίνεται η παρακάτω τοπολογία ενός πληροφοριακού συστήματος. Αν η διαθεσιμότητα του υποσυστήματος Α είναι 97% ενώ αυτή του συστήματος Β είναι 99%, ποια είναι η συνολική διαθεσιμότητα του συστήματος;



Απάντηση/Λύση

Η Διαθεσιμότητα (Δ) σε παράλληλα υποσυστήματα υπολογίζεται από τον τύπο:

$$\Delta = 1 - (1 - \Delta_A)^2 = 1 - (1 - 0{,}97)^2 = 1 - 0{,}03^2 = 1 - 0{,}0009 = 0{,}9991$$

Η διαθεσιμότητα σε σειριακά υποσυστήματα δίνεται από το γινόμενο των διαθεσιμοτήτων των υποσυστημάτων. Συνεπώς έχουμε:

$$\Delta_{\Sigma\upsilon\nu o\lambda\iota\kappa\acute{\eta}} = 0{,}9991 * \Delta_B = 0{,}9991 * 0{,}99 = 0{,}9891 = 98{,}91\%$$

Το συμπέρασμα είναι ότι ενώ το υποσύστημα Α έχει μόνο 97% διαθεσιμότητα, αν παράλληλα τοποθετήσουμε μια εφεδρεία, αυξάνουμε τη διαθεσιμότητα στο 99,91%, που αποτελεί πολύ σημαντική βελτίωση. Επίσης, η σειριακή τοποθέτηση υποσυστημάτων υποβιβάζει τη διαθεσιμότητα σε αυτή του υποσυστήματος με τη χαμηλότερη διαθεσιμότητα.

Κριτήριο αξιολόγησης 4

Περιγράψτε τα βασικά χαρακτηριστικά του υπολογιστικού νέφους.

Απάντηση/Λύση

Ο ορισμός του NIST (National Institute of Standards and Technology) απαριθμεί πέντε βασικά χαρακτηριστικά του ΥΝ:

- Είναι διαθέσιμο κατ' απαίτηση (on-demand), η επιχείρηση δηλαδή μπορεί να προμηθευτεί υπολογιστικούς πόρους όπως π.χ. χρόνο στον διακομιστή ή χώρο αποθήκευσης όποτε το χρειαστεί, αυτομάτως, χωρίς να απαιτείται η παρέμβαση από τον πάροχο της κάθε υπηρεσίας.
- Οι χρήστες αυτοεξυπηρετούνται (self-service).
- Παρέχεται ευρυζωνική πρόσβαση στο διαδίκτυο (broadband).
- Οι υπολογιστικοί πόροι είναι συγκεντρωμένοι κεντρικά.
- Παρέχει ελαστικότητα στην παροχή των πόρων, ενώ οι παρεχόμενες υπηρεσίες παρέχονται σε εγγυημένο επίπεδο (χρήση SLA). Για παράδειγμα, όποτε διαπιστώνεται αυξημένη χρήση μιας υπηρεσίας, μέσω του υπολογιστικού νέφους είναι πολύ απλό να προστεθεί επιπλέον δυναμικό σε

αυτή, κάτι για το οποίο θα απαιτείτο πολύ περισσότερος χρόνος ή και θα ήταν αδύνατο εάν μια εταιρεία υποχρεωνόταν να εγκαταστήσει νέες μηχανές στο δικό της κέντρο δεδομένων άμεσα.

Κριτήριο αξιολόγησης 5

Ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα των κεντρικών και των κατανεμημένων συστημάτων ελέγχου εκδόσεων;

Απάντηση/Λύση

Τα κεντρικά συστήματα ελέγχου εκδόσεων αναπτύχθηκαν για να καταγράφουν αλλαγές σε ένα κεντρικό σύστημα και να επιτρέπουν στους προγραμματιστές να συνεργάζονται. Τα κεντρικά συστήματα ελέγχου εκδόσεων έχουν πολλά θετικά σημεία, ωστόσο έχουν και ορισμένα σοβαρά μειονεκτήματα.

Πλεονεκτήματα των κεντρικών συστημάτων ελέγχου εκδόσεων:

- Είναι απλά στη ρύθμισή τους.
- Παρέχουν διαφάνεια στις αλλαγές.
- Επιτρέπουν στο διαχειριστή να ελέγξει εύκολα τη ροή εργασίας.

Μειονεκτήματα των κεντρικών συστημάτων ελέγχου εκδόσεων:

- Εάν ο διακομιστής έχει βλάβη, οι προγραμματιστές δεν μπορούν να αποθηκεύσουν τις αλλαγές που έχουν κάνει.
- Εάν ο διακομιστής είναι απομακρυσμένος και η ταχύτητα του δικτύου είναι αργή, η αποθήκευση των αρχείων (commit) είναι αργή.
- Οι ανεπιθύμητες αλλαγές μπορεί να δημιουργήσουν προβλήματα.
- Εάν η κεντρική βάση δεδομένων καταστραφεί, μπορεί να χαθεί ολόκληρο το ιστορικό.

Τα κατανεμημένα συστήματα ελέγχου εκδόσεων (DVCS) δεν βασίζονται σε κεντρικό διακομιστή. Επιτρέπουν στους προγραμματιστές να κλωνοποιήσουν το αποθετήριο και να εργαστούν σε αυτήν την τοπική έκδοση. Οι προγραμματιστές θα έχουν ολόκληρο το ιστορικό του έργου στο δικό τους υπολογιστή.

Πλεονεκτήματα κατανεμημένων συστημάτων ελέγχου εκδόσεων:

- Λόγω της τοπικής αποθήκευσης (local commit), το πλήρες ιστορικό είναι πάντα διαθέσιμο.
- Δεν χρειάζεται πρόσβαση σε απομακρυσμένο διακομιστή και συνεπώς υπάρχει γρηγορότερη πρόσβαση.
- Υπάρχει η δυνατότητα να προωθείτε τις αλλαγές σας συνεχώς.
- Είναι καλό για έργα όπου οι προγραμματιστές δεν δουλεύουν στον ίδιο χώρο.

Μειονεκτήματα κατανεμημένων συστημάτων ελέγχου εκδόσεων:

- Μπορεί να μην είναι πάντα κατανοητό/γνωστό ποιος έκανε την πιο πρόσφατη αλλαγή.
- Το κλείδωμα αρχείων δεν επιτρέπει σε διαφορετικούς προγραμματιστές να εργάζονται στο ίδιο κομμάτι κώδικα ταυτόχρονα. Αυτό βοηθά στην αποφυγή συγκρούσεων συγχώνευσης, αλλά επιβραδύνει την ανάπτυξη του λογισμικού.
- Το DVCS σάς δίνει τη δυνατότητα να κλωνοποιήσετε το αποθετήριο γεγονός που μπορεί να αποτελεί πρόβλημα ασφαλείας.

- Η εργασία με πολλά δυαδικά αρχεία απαιτεί τεράστιο χώρο και οι προγραμματιστές δεν μπορούν να δουν τις διαφορές με αυτοματοποιημένο τρόπο.

Κριτήριο αξιολόγησης 6

Τι είναι ο hypervisor σε μια αρχιτεκτονική με εικονικές μηχανές,; Με τη χρήση του διαδικτύου βρείτε και δώστε παραδείγματα υλοποιήσεων hypervisor.

Απάντηση/Λύση

Ο hypervisor ή ελεγκτής των εικονικών μηχανών (Virtual Machine Monitor - VMM) είναι λογισμικό ή firmware ή hardware το οποίο είναι υπεύθυνο για τη λειτουργία και εκτέλεση των εικονικών μηχανών. Ο υπολογιστής ο οποίος χρησιμοποιείται για να εκτελεστεί ο hypervisor είναι η μηχανή φιλοξενίας, ενώ οι εικονικές μηχανές (virtual machines) είναι οι φιλοξενούμενες μηχανές.

Παραδείγματα hypervisor: VMware ESXi, Citrix XenServer και Microsoft Hyper-V hypervisor.

Κριτήριο αξιολόγησης 7

Ποιες είναι οι διαφορές των αρχιτεκτονικών του virtualization και του containerization;

Απάντηση/Λύση

Οι διαφορές μεταξύ virtualization και containerization παρουσιάζονται στον παρακάτω πίνακα:

Περιοχή	**Virtualization**	**Containerization**
Απομόνωση (Isolation)	Παρέχει πλήρη απομόνωση από το λειτουργικό σύστημα του κεντρικού υπολογιστή αλλά και τις άλλες εικονικές μηχανές.	Συνήθως παρέχει «ελαφριά» απομόνωση στον κεντρικό υπολογιστή και από άλλα container, αλλά δεν παρέχει τόσο ισχυρή ασφάλεια όσο ένας εικονικός υπολογιστής.
Λειτουργικό σύστημα	Εκτελεί ένα πλήρες λειτουργικό σύστημα, συμπεριλαμβανομένου και του πυρήνα αυτού. Απαιτεί όμως περισσότερους πόρους συστήματος όπως CPU, μνήμη και αποθήκευση.	Εκτελεί ένα περιορισμένο τμήμα του λειτουργικού συστήματος αλλά μπορεί να προσαρμοστεί ώστε να περιέχει μόνο τις απαραίτητες υπηρεσίες για την εφαρμογή χρησιμοποιώντας λιγότερους πόρους συστήματος.
Συμβατότητα (combatibility)	Εκτελεί σχεδόν οποιοδήποτε λειτουργικό σύστημα μέσα στην εικονική μηχανή.	Εκτελεί την ίδια έκδοση του λειτουργικού συστήματος με τον κεντρικό υπολογιστή.
Διάταξη (Deployment)	Διατάσσει διαφορετικές VMs χρησιμοποιώντας το λογισμικού του Hypervisor.	Διατάσσει μεμονωμένα container χρησιμοποιώντας εργαλεία όπως το Docker ή πολλαπλά container χρησιμοποιώντας έναν ενορχηστρωτή όπως το Kubernetes.
Αποθήκευση (Persistence)	Χρήση εικονικών δίσκων.	Χρήση τοπικών δίσκων.
Εξισορρόπηση φόρτου (Load balancing)	Η εξισορρόπηση φόρτου εικονικής μηχανής γίνεται εκτελώντας εικονικές μηχανές σε άλλους διακομιστές.	Ένας ενορχηστρωτής μπορεί να ξεκινήσει ή να σταματήσει αυτόματα containers σε κόμβους του υπολογιστικού κόμβου για να διαχειριστεί τις αλλαγές στο φορτίο και τη διαθεσιμότητα.
Δικτύωση	Χρησιμοποιεί εικονικούς προσαρμογείς δικτύου.	Χρησιμοποιεί μια απομονωμένη προβολή ενός εικονικού προσαρμογέα δικτύου.

Περιοχή	Virtualization	Containerization
Ταχύτητα επεξεργασίας	Οι εικονικές μηχανές έχουν μεγαλύτερη υπολογιστική πολυπλοκότητα και συνεπώς τρέχουν πιο αργά.	Είναι γρηγορότερη τεχνολογία από τις εικονικές μηχανές.
Στοχευόμενο στοιχείο	Στοχεύουν στην εικονικοποίηση του hardware.	Στοχεύουν στην εικονικοποίηση των εφαρμογών.

Μέρος Γ – Ειδικά θέματα ευέλικτων μεθόδων

Πρακτικά θέματα ευελιξίας

"Tell me how you will measure me and
I will tell you how I will behave."

Eli Goldratt

Σύνοψη
Το έβδομο κεφάλαιο παρουσιάζει πρακτικά θέματα και πρακτικές που αφορούν όλες τις ευέλικτες μεθόδους. Πιο συγκεκριμένα παρουσιάζουμε πώς καταγράφουμε τις ιστορίες χρηστών, πώς γίνεται ο υπολογισμός της απαιτούμενης προσπάθειας για την ανάπτυξη του συστήματος, πώς γίνεται η ιεράρχηση των απαιτήσεων αλλά και τις βασικές μετρικές που χρησιμοποιούνται στα ευέλικτα έργα. Δίνονται παραδείγματα ώστε να γίνει κατανοητός ο τρόπος εργασίας.

7 Πρακτικά θέματα ευελιξίας

7.1 Οι ιστορίες χρηστών

Οι *ιστορίες χρηστών* αποτελούν κείμενα που περιγράφουν και αποτυπώνουν την αλληλεπίδραση του χρήστη με το σύστημα. Στόχος τους είναι να παρουσιάσουν την προστιθέμενη αξία που λαμβάνει ο χρήστης από το σύστημα στη συγκεκριμένη περίπτωση. Στην πράξη αποτελούν περιγραφή των εργασιών που πρέπει να γίνουν.

Μια ιστορία χρήστη περιγράφει λειτουργίες που είναι χρήσιμες για έναν χρήστη ή για τον αγοραστή του λογισμικού. Οι ιστορίες χρηστών αποτελούνται από τρία μέρη:

- μια γραπτή περιγραφή της ιστορίας
- συζητήσεις για την ιστορία που χρησιμεύουν για να δώσουν τις λεπτομέρειες της ιστορίας
- ελέγχους που απαιτούνται για να θεωρηθεί η υλοποίηση της ιστορίας ως ολοκληρωμένη

Παρόλο που οι ιστορίες χρηστών μοιάζουν αρκετά με τις προδιαγραφές απαιτήσεων λογισμικού και τις περιπτώσεις χρήσης, διαφέρουν από αυτές σε μερικά λεπτά αλλά σημαντικά σημεία (Leffingwell, 2010).

- Δεν είναι λεπτομερείς προδιαγραφές απαιτήσεων (κάτι που πρέπει να κάνει ένα σύστημα), αλλά είναι μάλλον διαπραγματεύσιμες εκφράσεις πρόθεσης (πρέπει να κάνει κάτι τέτοιο).
- Είναι σύντομες, ευανάγνωστες και κατανοητές από την ομάδα έργου αλλά και από όλους τους συμμετέχοντες.
- Αντιπροσωπεύουν μικρές επαυξήσεις λειτουργικότητας του συστήματος που μπορούν να αναπτυχθούν σε περίοδο ημερών έως εβδομάδων.
- Θα πρέπει να είναι σχετικά εύκολο να εκτιμηθούν, επομένως η απαιτούμενη προσπάθεια για την υλοποίηση της λειτουργικότητας να μπορεί να προσδιοριστεί γρήγορα.
- Αναπτύσσονται σε λίστες που μπορούμε να τις διαχειριστούμε εύκολα και δεν είναι αναγκαίο να έχουμε εκτενή έγγραφα.
- Προσθέτουμε λεπτομέρειες κατά τη διάρκεια του έργου και δεν χρειάζεται να υπάρχουν όλες οι λεπτομέρειες από την αρχή.

Το πιο κοινό πρότυπο για τη σύνταξη μιας ιστορίας χρήστη είναι αυτή που παρουσιάστηκε από τον M. Cohn (2004):

> **As a** <user type>, **I want** <some goal> **so that** <some reason>
>
> ή σε ελληνική μετάφραση,
>
> **Ως ένας** <τύπος χρήστη>, **θέλω να υλοποιήσω** <κάποιους στόχους>
> **έτσι ώστε** <να επιτύχω έναν στόχο>.

Με άλλα λόγια, σε μια ιστορία χρήστη απαντούμε στις ακόλουθες ερωτήσεις:

Ποιος: προς ποιου όφελος εκτελούμε αυτές τις ενέργειες;

Τι: Τι κάνουμε;

Γιατί: Γιατί το κάνουμε αυτό;

Για μια πιο αναλυτική περιγραφή των ιστοριών χρηστών χρησιμοποιούμε τις κάρτες χρηστών. Στην περίπτωση αυτή, για κάθε ιστορία χρήστη φτιάχνουμε μια κάρτα η οποία περιέχει τις ακόλουθες πληροφορίες:

- Τίτλος: <Ο τίτλος της ιστορίας χρήστη>
- Χρήστης: <Ποιος χρειάζεται την ιστορία>
- Περιγραφή: <Η περιγραφή της ιστορίας σε απλά βήματα>
- Στόχος: <Ποιος είναι ο βασικός στόχος της ιστορίας>

Το επόμενο βήμα είναι να συνδέσουμε τις ιστορίες χρηστών με τις *συνθήκες προς ικανοποίηση* (conditions of satisfaction) ή αλλιώς κριτήρια αποδοχής (acceptance criteria). Οι συνθήκες αυτές πρέπει να ικανοποιηθούν, ώστε η λειτουργία που περιγράφει η ιστορία χρήστη να γίνει αποδεκτή από τον χρήστη. Αποτελούν σενάρια (επιτυχίας, αλλά και αποτυχίας) ή επεξηγήσεις αναφορικά με το τί πρέπει να γίνει. Είναι δυνατό να υπάρχουν πολλαπλά κριτήρια αποδοχής ανά ιστορία χρήστη. Οι συνθήκες αυτές συνήθως προσθέτουν περιορισμούς, διευκρινίζουν την ιστορία χρήστη και γενικότερα διευκολύνουν τον έλεγχο του συστήματος. Η καταγραφή των ιστοριών χρηστών είναι μια ιδιαίτερα χρονοβόρα διαδικασία, αφού στις περισσότερες περιπτώσεις υπάρχει μεγάλος αριθμός τέτοιων συνθηκών. Αντίστοιχα, τα κριτήρια αποδοχής μπορούμε να τα δομήσουμε ώστε να είναι η δουλειά μας πιο εύκολη. Για παράδειγμα, θα μπορούσαμε να τα δομήσουμε με την παρακάτω μορφή.

Δεδομένης της κατάστασης (Given) <Περιγράφουμε την κατάσταση στην οποία εφαρμόζονται τα κριτήρια αποδοχής>

Πότε συμβαίνει (When) <Περιγράφουμε την ενέργεια που εκτελούμε>

Τότε <Then> <περιγράφουμε ένα σύνολο παρατηρήσιμων συμπεριφορών >

Ένα σύντομο παράδειγμα της περιγραφής μιας ιστορίας χρήστη είναι το ακόλουθο.

Τίτλος: Προβολή πληροφοριών άρθρου	**Προτεραιότητα:** Πολύ μεγάλη	**Εκτίμηση προσπάθειας:** 5
Ως (**As a**) Δημοσιογράφος **Θέλω να** (**I want to**) δω πληροφορίες σχετικά με το άρθρο που διαβάζω **Ώστε** (**so that I can**) να έρθω σε επαφή με το γραφείο τύπου του εκδότη		
Κριτήρια Αποδοχής (Acceptance criteria) **Δεδομένης της κατάστασης (Given)** να έρθω σε επαφή με το γραφείο τύπου του εκδότη **Πότε συμβαίνει (When)** να δω τα στοιχεία επαφής του εκδότη **Τότε <Then>** Τα στοιχεία επαφής του εκδότη θα πρέπει να περιλαμβάνουν: • ένα έγκυρο e-mail, • **ΚΑΙ** ένα έγκυρο τηλέφωνο, • **ΚΑΙ** μια διεύθυνση, • **ΚΑΙ** έναν έγκυρο ταχυδρομικό κώδικα.		

Πίνακας 7.1: *Παράδειγμα ιστορίας χρήστη*

Θα δώσουμε ένα δεύτερο παράδειγμα στο οποίο φαίνεται πώς μπορούν να συνδυαστούν τα μέρη των κριτηρίων αποδοχής ώστε η περιγραφή τους να γίνει πιο συγκεκριμένη. Στη γενική περίπτωση η κατάσταση μπορεί να περιγράφεται με πολλά κριτήρια. Το παράδειγμα αφορά την ανάληψη μετρητών από μια τράπεζα (βλέπε Πίνακα 7.2).

Τίτλος: Ανάληψη μετρητών από τράπεζα	**Προτεραιότητα:** Πολύ μεγάλη	**Εκτίμηση προσπάθειας:** 5
Ως (**As a**) χρήστης **Θέλω να** (**I want to**) κάνω ανάληψη από το λογαριασμό μου από ένα ATM **Ώστε** (**so that I can**) να μπορώ να πάρω μετρητά από το λογαριασμό μου γρήγορα και από οπουδήποτε.		
Κριτήρια Αποδοχής (Acceptance criteria)		

Δεδομένης της κατάστασης (Given)		ανάληψη μετρητών από ATM
	ΚΑΙ	ο λογαριασμός είναι έγκυρος
	ΚΑΙ	η κάρτα είναι έγκυρη
	ΚΑΙ	το ATM έχει ικανή ποσότητα χαρτονομισμάτων
Πότε συμβαίνει (When)		ο πελάτης ζητά μετρητά
Τότε <Then>		εξασφαλίζουμε ότι έγινε η χρέωση στο λογαριασμό,
•	**ΚΑΙ**	το ATM έδωσε τα χρήματα στον πελάτη,
•	ΚΑΙ	η κάρτα μετά το τέλος της συναλλαγής επιστράφηκε

Πίνακας 7.2: *Παράδειγμα ιστορίας χρήστη*

Οι ιστορίες χρηστών είναι απλές στη συγγραφή, καθώς και στην κατανόησή τους, τόσο από το τεχνικό προσωπικό όσο και από τους χρήστες, καθώς δεν υποδηλώνουν μια συγκεκριμένη λύση ή μια αρχιτεκτονική. Αυτό το στοιχείο είναι σημαντικό, επειδή οι απαιτήσεις υψηλού επιπέδου θα πρέπει να περιγράφουν το πρόβλημα και όχι τη λύση.

Ο τίτλος της ιστορίας χρήστη θα πρέπει να είναι σύντομος, μοναδικός και σαφής. Μερικά καλά παραδείγματα τίτλων είναι τα ακόλουθα:

- Προσθήκη των στοιχείων επικοινωνίας στην αρχική σελίδα
- Αποδοχή χρεωστικής κάρτας στο καλάθι αγορών

Μερικά όχι τόσο επιτυχημένα παραδείγματα:

- Πραγματοποίηση αλλαγών (είναι ασαφής και όχι συγκεκριμένη)
- Αίτηση πελάτη (είναι ασαφής, δεν αναφέρει σε ποια αίτηση αναφέρεται)

7.1.1 Το ακρώνυμο INVEST σε καλές ιστορίες χρηστών

Οι ευέλικτες ομάδες αφιερώνουν σημαντικό χρόνο στον εντοπισμό, την επεξεργασία και κατανόηση των ιστοριών των χρηστών, την καταγραφή αυτών και των ελέγχων αποδοχής. Οι ιστορίες χρηστών είναι η βάση για την ανάπτυξη του συστήματος και συνεπώς η σωστή ανάπτυξη αυτών είναι θεμελιώδους σημασίας. Ο Bill Wake (2013) επινόησε το αρκτικόλεξο INVEST για να περιγράψει τα χαρακτηριστικά μιας καλής ιστορίας χρήστη.

- Ανεξάρτητη (Independent)
- Διαπραγματεύσιμη (Negotiable)
- Πολύτιμη (Valuable)
- Εκτιμώμενη (Estimable)
- Μικρή σε μέγεθος (Small)
- Ελέγξιμη (Testable)

Το μοντέλο INVEST είναι αρκετά διαδεδομένο και πολλές ευέλικτες ομάδες αξιολογούν τις ιστορίες τους σε σχέση με αυτές τις ιδιότητες.

Ανεξάρτητες

Μια ιστορία χρήστη είναι ανεξάρτητη όταν η ομάδα ανάπτυξης μπορεί να την υλοποιήσει, να την ελέγξει και να την παραδώσει ανεξάρτητα από άλλες ιστορίες χρήστη. Βέβαια είναι αρκετά συνηθισμένο μια ιστορία χρήστη να έχει φυσικές εξαρτήσεις με άλλες ιστορίες χρήστη. Το βασικό χαρακτηριστικό είναι να μπορούν να υλοποιηθούν ανεξάρτητα και να μη δημιουργούν περιορισμούς στη σειρά υλοποίησης. Το χαρακτηριστικό αυτό είναι σημαντικό γιατί επιτρέπει την πραγματική ιεράρχηση της κάθε ιστορίας χρήστη. Όταν υπεισέρχονται εξαρτήσεις, μπορεί να μην είναι δυνατή η υλοποίηση μιας ιστορίας χρήστη (η οποία μπορεί να είναι σημαντική για τους χρήστες) χωρίς την υλοποίηση άλλων ιστοριών (πολύ λιγότερο σημαντικών για τους χρήστες.

Για παράδειγμα, μπορούμε να παρουσιάσουμε μια λίστα προϊόντων σε μια οθόνη ή ένα μεμονωμένο προϊόν, μια ταξινομημένη λίστα, ή τη δυνατότητα να φιλτράρουμε τα προϊόντα με βάση κριτήρια. Όλες αυτές οι ιστορίες χρήστη έχουν αξία για τον πελάτη αλλά ταυτόχρονα σχετίζονται μεταξύ τους. Όμως κάθε sprint μπορεί να συμπεριλάβει οποιαδήποτε από αυτές στο sprint backlog και συνεπώς είναι ανεξάρτητες. Ωστόσο, πολλές εξαρτήσεις είναι χωρίς αξία, και δημιουργούν περιορισμούς που μας καθυστερούν και συνεπώς θα πρέπει να βρούμε τρόπους να τις εξαλείψουμε. Το ακόλουθο παράδειγμα είναι ένα παράδειγμα λειτουργικής εξάρτησης χωρίς αξία:

> Ως διαχειριστής, μπορώ να ορίσω τους κανόνες για την ασφαλή δημιουργία κωδικού πρόσβασης του καταναλωτή έτσι ώστε οι χρήστες να απαιτείται να δημιουργούν ασφαλείς κωδικούς πρόσβασης, διατηρώντας το σύστημα ασφαλές.
>
> Ως πελάτης, οφείλω να ακολουθήσω τους κανόνες ασφαλείας δημιουργίας κωδικού πρόσβασης που ορίζονται από το διαχειριστή ώστε να μπορώ να διατηρήσω υψηλή ασφάλεια στον λογαριασμό μου.

Σε αυτό το παράδειγμα, η ιστορία του πελάτη εξαρτάται από την ιστορία του διαχειριστή. Επιπλέον, η ιστορία του διαχειριστή μπορεί να δοκιμαστεί μόνο στο τμήμα που αφορά στη ρύθμιση της πολιτικής και όχι στο κομμάτι της λειτουργίας, διότι η λειτουργία εξαρτάται από την ιστορία του πελάτη. Συνεπώς οι δύο αυτές ιστορίες χρήστη είναι εξαρτημένες και επιπλέον μας εμποδίζουν από μια πιθανή αποδέσμευση του προϊόντος που δεν εμπεριέχει και τις δύο αυτές ιστορίες χρήστη. Μια αναδόμηση αυτών, ώστε η ρύθμιση να συνδέεται άμεσα και με την επιβολή των κανόνων πιθανώς να έλυνε αυτό το θέμα. Για παράδειγμα:

> Ως διαχειριστής, μπορώ να ορίσω την περίοδο λήξης του κωδικού πρόσβασης έτσι ώστε οι χρήστες να αναγκάζονται να αλλάζουν περιοδικά τους κωδικούς πρόσβασης.
>
> Ως διαχειριστής, μπορώ να ορίσω τα χαρακτηριστικά ισχύος του κωδικού πρόσβασης έτσι ώστε οι χρήστες να δημιουργούν δύσκολους κωδικούς πρόσβασης.

Σε αυτή τη μορφή, η κάθε ιστορία χρήστη μπορεί να σταθεί μόνη της, μπορεί να αναπτυχθεί, να δοκιμαστεί και να παραδοθεί ανεξάρτητα.

Διαπραγματεύσιμη (Negotiable)

Μια ιστορία χρήστη δεν είναι συμβόλαιο αλλά ούτε και αποτελεί μέρος αυτού. Μια ιστορία χρήστη είναι μια αποτύπωση της επιθυμητής λειτουργικότητας και υπόκειται σε συζήτηση. Αυτή η συζήτηση μεταξύ του πελάτη και της ομάδας ανάπτυξης θα πρέπει να βασίζεται στην παραδοχή ότι οι ανάγκες της επιχείρησης είναι πάντα ιδιαίτερα σημαντικές και συνεπώς θα πρέπει πάντα να καταγράφονται και να τυγχάνουν της απαραίτητης σημασίας. Η ύπαρξη αυτής της δυναμικής και η έλλειψη περιορισμών και πολύ λεπτομερών απαιτήσεων ενισχύει την ικανότητα της ομάδας και της επιχείρησης να κάνουν αντισταθμίσεις μεταξύ λειτουργικότητας και ημερομηνιών παράδοσης.

Πολύτιμη (Valuable)

Ο στόχος μιας ευέλικτης ομάδας είναι να παράγει τη μεγαλύτερη δυνατή αξία, δεδομένων των υπαρχόντων περιορισμών χρόνου και πόρων. Επομένως, η αξία είναι το πιο σημαντικό χαρακτηριστικό στο μοντέλο INVEST και κάθε ιστορία χρήστη πρέπει να παρέχει κάποια αξία, έστω και μικρή, στον χρήστη ή τον πελάτη.

Μια πρόκληση που αντιμετωπίζουν οι ομάδες έργου είναι να μάθουν πως να γράφουν μικρές ιστορίες χρηστών που ταυτόχρονα προσφέρουν αξία. Για το λόγο αυτό οι ιστορίες χρηστών πρέπει να είναι γραμμένες με τέτοιο τρόπο ώστε η προσφερόμενη αξία να είναι εμφανής. Αυτό επιτυγχάνεται με την κατάτμηση της λειτουργικότητας με κάθετο τρόπο. Η κάθετη κατάτμηση σημαίνει ότι η κατασκευή του συστήματος θα πρέπει να γίνει προσφέροντας ολοκληρωμένες λειτουργίες, έστω και μικρές, αντί να κατασκευάζουμε επίπεδα λογισμικού (software layers), τα οποία πολλές φορές δεν έχει εμφανή αξία για τον τελικό χρήστη. Για παράδειγμα, η ανάπτυξη του επιπέδου της επιχειρηματικής λογικής (business logic layer), αν και χρήσιμη, δεν έχει εμφανή αξία για τον πελάτη παρά μόνο αν υπάρχει και η αντίστοιχη γραφική διεπαφή.

Αν και οι ιστορίες χρηστών εστιάζονται στον χρήστη που αλληλοεπιδρά με το σύστημα, μερικές φορές είναι καλύτερο να εστιαστούμε σε ένα βασικό συμμετέχοντα ώστε να γίνει εμφανής η παραγόμενη αξία. Για παράδειγμα:

> Ως καταναλωτής, μπορώ να δω προγράμματα τιμολόγησης ενέργειας που με ενδιαφέρουν, ώστε να μπορώ να εγγραφώ σε ένα πρόγραμμα που ταιριάζει καλύτερα στον τρόπο ζωής μου.

Η ιστορία χρήστη θα μπορούσε να γραφεί με τον παρακάτω τρόπο ώστε να είναι καλύτερα εμφανής η αξία της ιστορίας.

> Ως διευθυντής μάρκετινγκ κοινής ωφέλειας, μπορώ να παρουσιάσω στους πελάτες νέα προγράμματα τιμολόγησης, ώστε να παραμείνουν πελάτες της εταιρείας μου.

Ένα άλλο δύσκολο σημείο που αντιμετωπίζουν οι ομάδες έργου κατά τη συγγραφή ιστοριών χρήστη είναι στο να παρουσιάσουν την προσφερόμενη αξία που προκύπτει από τεχνικές ιστορίες χρήστη, όπως για παράδειγμα η ανακατασκευή κώδικα, η αναβάθμιση συστατικών του συστήματος κ.λπ. Για παράδειγμα, η ιστορία χρήστη

> Αναδιαμορφώστε το σύστημα καταγραφής σφαλμάτων.

θα μπορούσε να παρουσιαστεί με καλύτερο τρόπο ώστε να γίνεται εμφανής η αξία αυτής. Συνεπώς θα μπορούσε να ξαναγραφεί ως:

> Ως καταναλωτής, μπορώ να λάβω ένα συνεπές και ξεκάθαρο μήνυμα λάθους οπουδήποτε στο προϊόν, ώστε να γνωρίζω καλύτερα στο πώς να αντιμετωπίσω το συγκεκριμένο πρόβλημα.

ή

> Ως μέλος της τεχνικής υποστήριξης, θέλω ο χρήστης να λαμβάνει ένα συνεπές και ξεκάθαρο μήνυμα οπουδήποτε στην εφαρμογή, ώστε να μπορεί να διορθώσει το πρόβλημα χωρίς να καλέσει την τεχνική υποστήριξη.

Σε αυτά τα τελευταία παραδείγματα, η αξία που δίνεται από τις ιστορίες χρήστη είναι ξεκάθαρη.

Εκτιμώμενη (Estimable)

Μια καλή ιστορία χρήστη θα πρέπει να μπορεί να εκτιμηθεί ή να διαστασιολογηθεί, ώστε να μπορεί να ιεραρχηθεί σωστά. Αν και μπορούμε να έχουμε ιστορίες χρήστη οποιουδήποτε μεγέθους, προκειμένου μια ιστορία χρήστη να ενταχθεί στο sprint backlog, θα πρέπει να είμαστε σε θέση να παρέχουμε μια κατά προσέγγιση εκτίμηση της πολυπλοκότητάς της και του όγκου της απαιτούμενης εργασίας για την ολοκλήρωσή της. Το μεγαλύτερο μέγεθος μια ιστορίας χρήστη είναι αυτό που μπορεί να υλοποιηθεί στη διάρκεια ενός sprint.

Εάν η ομάδα έργου δεν είναι σε θέση να εκτιμήσει μια ιστορία χρήστη, υποδηλώνει γενικά ότι η ιστορία είναι πολύ μεγάλη ή ότι υπάρχει αβεβαιότητα. Εάν η ιστορία χρήστη είναι πολύ μεγάλη για να εκτιμηθεί, θα πρέπει να χωριστεί σε μικρότερες ιστορίες χρήστη. Εάν η ιστορία χρήστη έχει πολλές ασάφειες για να εκτιμηθεί, τότε μπορεί να χρησιμοποιηθεί μια τεχνική (technical spike) ή λειτουργική ιστορία αιχμής (functional spike)[11] για τη μείωση της αβεβαιότητας, ώστε να προκύψουν μία ή περισσότερες εκτιμώμενες ιστορίες χρηστών. Ένα από τα βασικά πλεονεκτήματα της εκτίμησης των ιστοριών των χρηστών δεν είναι μόνο η εξαγωγή ενός ακριβούς μεγέθους, αλλά κυρίως η ανακάλυψη των κρυφών υποθέσεων και κριτηρίων αποδοχής που λείπουν και η καλύτερη κατανόηση της ιστορίας.

Μικρή σε μέγεθος (Small)

Οι ιστορίες χρήστη πρέπει να είναι αρκετά μικρές ώστε να μπορούν να ολοκληρωθούν σε μια επανάληψη. Δύο είναι οι κύριοι λόγοι: αυξημένη απόδοση και μειωμένη πολυπλοκότητα.

Οι μικρές ιστορίες χρήστη οδηγούν σε αυξημένη απόδοση για πολλούς λόγους. Μερικοί από τους λόγους είναι:

- Είναι πιο κατανοητές από την ομάδα
- Είναι πιο εύκολο να τις διαχειριστούμε και να τις εντάξουμε σε ένα sprint
- Είναι πιο εύκολο να ελέγξουμε την ορθότητά τους

Ο δεύτερος λόγος που προτιμούμε μικρές ιστορίες χρηστών είναι η πολυπλοκότητα. Οι μικρότερες ιστορίες χρήστη ολοκληρώνονται γρηγορότερα λόγω της μειωμένης πολυπλοκότητάς τους. Θα πρέπει να τονίσουμε ότι η πολυπλοκότητα έχει μια μη γραμμική σχέση με το μέγεθος. Αυτό είναι πιο εύκολα κατανοητό στον έλεγχο λογισμικού (testing), όπου ο αριθμός των ελέγχων που απαιτούνται για την επαλήθευση της σωστής λειτουργίας του συστήματος αυξάνεται εκθετικά σε σχέση με την πολυπλοκότητα.

Ελέγξιμες (Testable)

Στην ευέλικτη προσέγγιση αποτελεί προτεραιότητα ο συνεχής έλεγχος του κώδικα. Ο λόγος είναι ότι σε κάθε sprint παραδίδουμε μια επαύξηση του συστήματος η οποία θα πρέπει να είναι λειτουργική για τον τελικό χρήστη. Αυτό σημαίνει ότι όλες οι ιστορίες χρήστη θα πρέπει να είναι ελέγξιμες. Αν μια ιστορία χρήστη δε μπορεί να δοκιμαστεί, τότε η ιστορία χρήστη δεν έχει γραφεί σωστά ή είναι υπερβολικά περίπλοκη, ή ίσως εξαρτάται από άλλες ιστορίες χρήστη.

Πολλές φορές η δυνατότητα δοκιμής μια ιστορίας χρήστη περιορίζεται όταν περιέχει μη λειτουργικές απαιτήσεις[12]. Για παράδειγμα, οι παρακάτω ιστορίες χρήστη

Ένας χρήστης πρέπει να βρει το λογισμικό εύκολο στη χρήση.

Ένας χρήστης δεν πρέπει ποτέ να περιμένει πολλή ώρα για να εμφανιστεί οποιαδήποτε οθόνη.

Αυτές οι ιστορίες δεν μπορούν να δοκιμαστούν. Επίσης θα πρέπει να στοχεύουμε στην αυτοματοποίηση των δοκιμών όταν αυτό είναι εφικτό.

Ένας δεκάλογος για τη συγγραφή καλών ιστοριών χρήστη είναι ο ακόλουθος:

[11] Τα Spikes είναι μια ιδέα του Extreme Programming (XP) και είναι ένας ειδικός τύπος ιστορίας χρήστη που χρησιμοποιείται για την απόκτηση των απαραίτητων γνώσεων για τη μείωση του κινδύνου μιας τεχνικής προσέγγισης, την καλύτερη κατανόηση μιας απαίτησης ή την αύξηση της αξιοπιστίας της εκτίμησης μιας ιστορίας χρήστη. Ένα spike έχει μέγιστη διάρκεια ενός sprint. Το Spike είναι ένας πολύ καλός τρόπος για τον μετριασμό των κινδύνων έγκαιρα και επιτρέπει στην ομάδα έργου να κατανοήσει τις τεχνικές δυσκολίες και την πολυπλοκότητα ενός PBI.

[12] Οι μη-λειτουργικές περιγράφουν ιδιότητες του πληροφοριακού συστήματος που έχουν άμεση σχέση με τα παρακάτω χαρακτηριστικά: Απόδοση (performance), Χρηστικότητα (usability), Ασφάλεια (security), Νομιμότητα (legislative), Ιδιωτικότητα (privacy).

1. Για να εντοπίσουμε τις ιστορίες χρήστη, ξεκινούμε εξετάζοντας τους στόχους κάθε ρόλου χρήστη στη χρήση του συστήματος.
2. Οι ιστορίες χρήστη θα πρέπει να αφορούν μόνο έναν χρήστη.
3. Όταν δημιουργούμε μια ιστορία χρήστη, προσπαθούμε να κατατμήσουμε κάθετα τη λειτουργικότητα, ώστε να δημιουργήσουμε ιστορίες που διατρέχουν όλα τα επίπεδα της εφαρμογής.
4. Θα πρέπει να καταγράφουμε τους περιορισμούς κάθε ιστορίας χρήστη μαζί με τους αναγκαίους ελέγχους που να διασφαλίζουν ότι δεν παραβιάζονται οι περιορισμοί.
5. Προσπαθούμε να αποφύγουμε την περιγραφή της διεπαφής χρήστη εντός της ιστορίας χρήστη όσο το δυνατόν περισσότερο.
6. Θα πρέπει να συμπεριλαμβάνουμε το ρόλο του χρήστη κατά την καταγραφή της ιστορίας χρήστη. Ένας χρήστης μπορεί να έχει πολλούς ρόλους.
7. Οι ιστορίες χρηστών θα πρέπει να γράφονται σε ενεργητική φωνή. Για παράδειγμα, είναι καλύτερα να γράψουμε «Ένας άνεργος που αναζητά εργασία μπορεί να δημοσιεύσει ένα βιογραφικό» αντί «Ένα βιογραφικό μπορεί να αναρτηθεί από έναν άνεργο».
8. Οι ιστορίες χρήστη θα πρέπει να γράφονται από την οπτική του πελάτη και όχι από αυτή του προγραμματιστή.
9. Δεν πρέπει να αριθμούμε τις ιστορίες χρηστών, διότι οι αριθμοί μάλλον κάνουν την καταγραφή πολύπλοκη.
10. Οι ιστορίες χρηστών θα πρέπει να είναι μικρές αλλά όχι πολύ μικρές. Θα πρέπει πάντα να είναι διακριτή η αξία που προσφέρουν στον πελάτη.

7.1.2 Οι μέθοδοι κατάτμησης των ιστοριών χρηστών

Οι ιστορίες χρηστών συχνά οργανώνονται σε epics και features, τα οποία περιγράφουν μεγάλα κομμάτια λειτουργικότητας των χρηστών. Συνήθως ένα epic είναι πολύ μεγάλο για να απεικονιστεί με μια ιστορία χρήστη και θα πρέπει να κατατμηθεί σε μικρότερα κομμάτια λειτουργικότητας που αντιστοιχούν σε ιστορίες χρηστών. Υπάρχει ένας αριθμός τεχνικών-προτύπων που μπορεί να χρησιμοποιηθεί για τη δημιουργία κατάλληλων περιπτώσεων χρήσης. Τα πρότυπα αυτά εργασίας παρουσιάζονται στις επόμενες παραγράφους (Lawrence, 2021).

Με χρήση βημάτων ροής εργασιών (workflow steps)

Προσδιορίζουμε τα συγκεκριμένα βήματα που κάνει ένας χρήστης για να ολοκληρώσει μια συγκεκριμένη ροή εργασίας και, στη συνέχεια, εφαρμόζουμε τη ροή εργασίας για να παράγουμε τις ιστορίες χρηστών.

Για παράδειγμα ένα epic ή μια μεγάλη ιστορία χρήστη

Ως διαχειριστής περιεχομένου, μπορώ να δημοσιεύσω μια είδηση στον εταιρικό ιστότοπο.

Μπορεί να αναλυθεί σε ιστορίες χρήστη που αφορούν τα επιμέρους βήματα

Ως διαχειριστής περιεχομένου, μπορώ να δημοσιεύσω μια είδηση απευθείας στον εταιρικό ιστότοπο.

Ως διαχειριστής περιεχομένου, μπορώ να δημοσιεύσω μια είδηση μετά από έγκριση του συντάκτη.

Ως διαχειριστής περιεχομένου, μπορώ να δημοσιεύσω μια είδηση μετά από νομικό έλεγχο.

Κ.λπ.

Με βάση τις λειτουργίες (π.χ. CRUD)

Είναι ίσως ο πιο φυσικός τρόπος για να διαχωρίσουμε τις ιστορίες χρηστών. Για παράδειγμα ένα epic ή μια μεγάλη ιστορία χρήστη

Ως χρήστης, μπορώ να διαχειριστώ τον λογαριασμό μου.

γίνεται

Ως χρήστης, μπορώ να εγγραφώ για δημιουργία λογαριασμού.

Ως χρήστης, μπορώ να επεξεργαστώ τις ρυθμίσεις του λογαριασμού μου.

Ως χρήστης, μπορώ να αλλάξω τον κωδικό μου.

Ως χρήστης, μπορώ να ακυρώσω τον λογαριασμό μου.

Με βάση παραλλαγές επιχειρηματικών κανόνων

Χρησιμοποιείται στην περίπτωση που ιστορίες χρηστών έχουν την ίδια δομή αλλά αλλάζουν οι επιχειρηματικοί κανόνες που εφαρμόζονται κάθε φορά.

Για παράδειγμα η παρακάτω ιστορία χρήστη

Ως χρήστης, μπορώ να αναζητήσω πτήσεις με ευέλικτες ημερομηνίες.

Μπορεί να μετασχηματιστεί σε

Ως χρήστης, μπορώ να αναζητήσω πτήσεις με ευέλικτες ημερομηνίες ως 10 ημέρες μεταξύ x και y."

Ως χρήστης, μπορώ να αναζητήσω πτήσεις με ευέλικτες ημερομηνίες ένα Σαββατοκύριακο τον Δεκέμβριο.

Με βάση παραλλαγές στα δεδομένα

Η πολυπλοκότητα σε μια ιστορία χρήστη μπορεί να προέρχεται από τις παραλλαγές στα δεδομένα.

Για παράδειγμα η ιστορία χρήστη

Ως υπάλληλος τηλεπικοινωνιακής εταιρείας, μπορώ να στείλω μηνύματα στους πελάτες μου.

Μπορεί να μετασχηματιστεί σε

Ως υπάλληλος τηλεπικοινωνιακής εταιρείας, μπορώ να στείλω μηνύματα στα ελληνικά στους πελάτες μου.

Ως υπάλληλος τηλεπικοινωνιακής εταιρείας, μπορώ να στείλω μηνύματα στα αγγλικά στους πελάτες μου.

Με βάση τη μέθοδο εισαγωγής δεδομένων

Η πολυπλοκότητα μερικές φορές βρίσκεται στη διεπαφή χρήστη παρά στην ίδια τη λειτουργικότητα. Σε αυτήν την περίπτωση, θα πρέπει να διαχωρίσουμε την ιστορία χρήστη σε δύο: αυτή που περιέχει τη λειτουργικότητα και την απλούστερη δυνατή διεπαφή χρήστη και σε αυτή που περιέχει τη διεπαφή χρήστη.

Για παράδειγμα η ιστορία χρήστη

Ως χρήστης, μπορώ να αναζητήσω πτήσεις μεταξύ δύο προορισμών.

μπορεί να χωριστεί σε

Ως χρήστης, μπορώ να αναζητήσω πτήσεις μεταξύ δύο προορισμών χρησιμοποιώντας απλή εισαγωγή ημερομηνίας.

Ως χρήστης, μπορώ να αναζητήσω πτήσεις μεταξύ δύο προορισμών με μια διεπαφή χρήστη με βάση το ημερολόγιο.

Με βάση την μεγαλύτερη απαιτούμενη προσπάθεια

Μερικές φορές μια ιστορία χρήστη μπορεί να χωριστεί σε πολλά μέρη όπου το μεγαλύτερο μέρος της απαιτούμενης προσπάθειας προορίζεται για την υλοποίηση του πρώτου μέρους. Για παράδειγμα, αυτή η ιστορία χρήστη με αντικείμενο την επεξεργασία πιστωτικών καρτών.

Ως χρήστης, μπορώ να πληρώσω για την πτήση μου με VISA, MasterCard, Diners Club ή American Express.

θα μπορούσε να χωριστεί σε τέσσερις ιστορίες χρήστη, μία για κάθε τύπο κάρτας. Η πρώτη ιστορία χρήστη θα περιέχει την αναγκαία υποδομή επεξεργασίας πιστωτικών καρτών και θα υλοποιηθεί με την πρώτη ιστορία χρήστη. Η προσθήκη περισσότερων τύπων καρτών απαιτεί σημαντικά λιγότερη προσπάθεια. Συνεπώς η παραπάνω ιστορία χρήστη μετασχηματίζεται σε

Ως χρήστης, μπορώ να πληρώσω με έναν τύπο πιστωτικής κάρτας (VISA, MC, DC, AMEX).

Ως χρήστης, μπορώ να πληρώσω και με τους τέσσερις τύπους πιστωτικών καρτών (VISA, MC, DC, AMEX)

Με βάση τις απλές – σύνθετες ιστορίες χρηστών

Μια ιστορία χρήστη που μπορεί να έχει πολλά εναλλακτικά σενάρια δημιουργούμε ιστορίες χρηστών για κάθε ένα από τα εναλλακτικά σενάρια. Για παράδειγμα:

Ως χρήστης, μπορώ να αναζητήσω πτήσεις μεταξύ δύο προορισμών.

παραμένει ως η βασική ιστορία χρήστη αλλά διαχωρίζουμε και τις παραλλαγές της δημιουργώντας νέες ιστορίες χρηστών όπως,

Ως χρήστης, μπορώ να αναζητήσω πτήσεις μεταξύ δύο προορισμών καθορίζοντας έναν μέγιστο αριθμό στάσεων.

Ως χρήστης, μπορώ να αναζητήσω πτήσεις μεταξύ δύο προορισμών χρησιμοποιώντας ευέλικτες ημερομηνίες.

Κ.λπ.

Με βάση τη δημιουργία της πιο απλής συμπεριφοράς και την αναβολή ειδικών χαρακτηριστικών

Μερικές φορές, ένα μεγάλο μέρος της προσπάθειας οφείλεται στην ανάγκη δημιουργίας ενός ποιοτικού/μη-λειτουργικού χαρακτηριστικού, αφού η αρχική υλοποίηση της λειτουργικότητας δεν είναι τόσο δύσκολη. Σε αυτήν την περίπτωση, η ιστορία χρήστη διαχωρίζεται σε δύο, όπου η πρώτη αφορά τη λειτουργία και η δεύτερη το μη λειτουργικό χαρακτηριστικό. Για παράδειγμα αν η πρώτη ιστορία χρήστη είναι

Ως χρήστης, μπορώ να αναζητήσω πτήσεις μεταξύ δύο προορισμών.

Προσθέτουμε μια δεύτερη ιστορία χρήστη η οποία θα μπορούσε να είναι

Ως χρήστης, μπορώ να αναζητήσω πτήσεις μεταξύ δύο προορισμών σε λιγότερο από 5 δευτερόλεπτα.

Αυτή η προσέγγιση μπορεί να χρησιμοποιηθεί για οποιαδήποτε μη λειτουργική απαίτηση, και όχι μόνο για την απόδοση του συστήματος.

7.2 Εκτίμηση προσπάθειας και ιεράρχηση απαιτήσεων

Η εκτίμηση κόστους λογισμικού είναι μία επιστημονική περιοχή που άρχισε να αναπτύσσεται την εποχή που το λογισμικό άρχισε να διαδίδεται ραγδαία, βρίσκοντας ευρεία εφαρμογή σε κλάδους όπως εμπορικές εφαρμογές, εφαρμογές διαχείρισης δεδομένων, παράλληλα με τις επιστημονικές εφαρμογές από τις οποίες και ξεκίνησε. Με τον όρο εκτίμηση κόστους εννοούμε το σύνολο των δραστηριοτήτων που αποσκοπούν στον προσδιορισμό του απαιτούμενου κατασκευαστικού κόστους για την ολοκλήρωση ενός έργου λογισμικού.

Πρόκειται για μία αναγκαία διοικητική ενέργεια που αποτελεί την αρχική δραστηριότητα για τον χρονοπρογραμματισμό και τη στελέχωση του έργου. Στα έργα λογισμικού η κυριότερη συνιστώσα κόστους είναι η ανθρώπινη προσπάθεια που απαιτείται για την εκτέλεση του έργου. Επομένως η εκτίμηση κόστους σε μεγάλο βαθμό ανάγεται σε εκτίμηση απαιτούμενης προσπάθειας (effort estimation).

Για την εκτίμηση της απαιτούμενης προσπάθειας μας απασχολούν ερωτήματα όπως πόσο μεγάλο θα είναι το έργο, πόσο θα κοστίσει, πόσο θα διαρκέσει η ανάπτυξή του, από ποιους παράγοντες θα επηρεασθούν όλα αυτά και σε ποιο βαθμό.

Οι εκτιμήσεις της αναγκαίας προσπάθειας αποτελούν μία πηγή πολύτιμης πληροφορίας για τη διοίκηση ενός έργου λογισμικού. Πιο συγκεκριμένα είναι απαραίτητες για τις εξής λειτουργίες της διοίκησης:

- Κατανομή Πόρων: Ανάλογα με το προϋπολογιζόμενο κόστος θα ανατεθούν πόροι σε διάφορες δραστηριότητες του έργου (π.χ. ανθρώπινοι πόροι στη διαδικασία συλλογής απαιτήσεων, συγγραφή κώδικα, διασφάλιση ποιότητας). Επίσης θα επιλεγούν και άτομα με τα κατάλληλα προσόντα που θα στελεχώσουν τις διάφορες ομάδες του έργου (π.χ. εκτίμηση υψηλού κόστους για τη φάση της ανάλυσης θα οδηγήσει σε ανάγκη στελέχωσης του έργου με αναλυτές λογισμικού).

- Χρονοπρογραμματισμός Έργου: Η εκτίμηση του ανθρώπινου φόρτου, εκτός από τον αριθμό των εργαζόμενων και την ένταση της προσπάθειάς τους οδηγεί και σε εκτίμηση της χρονικής διάρκειας του έργου και στη συνέχεια της χρονικής διάρκειας κάθε φάσης και επιμέρους εργασίας. Επομένως τα αποτελέσματα της εκτίμησης μπορούν να αποτελέσουν τη βάση με την οποία θα τοποθετηθούν οι φάσεις του έργου στο σχετικό χρονοδιάγραμμα.

- Ικανοποίηση Χρονικών και Οικονομικών Περιορισμών: Η εκτίμηση του κόστους από την πλευρά του κατασκευαστή του δίνει τη δυνατότητα (α) να διεκδικήσει την ανάθεση έργων με ρεαλιστικό, αποδοτικό και ανταγωνιστικό τίμημα (β) να αιτιολογήσει τα οικονομικά αιτήματά του προς το χρηματοδότη οργανισμό και (γ) να καταρτίζει ρεαλιστικά χρονοδιαγράμματα σε σχέση με τους χρονικούς περιορισμούς που τίθενται (π.χ. να συμπιέσει σε κάποιο βαθμό τη διάρκεια εκτέλεσης του έργου αυξάνοντας τον απαιτούμενο φόρτο εργασίας). Ανάλογα, η εκτίμηση του κόστους από την πλευρά του χρηματοδότη / πελάτη του λογισμικού του δίνει τη δυνατότητα (α) να αξιολογεί τις προσφορές των κατασκευαστών, (β) να ορίζει ρεαλιστικά επίπεδα προϋπολογισμών για τα έργα για τα οποία πρόκειται να ζητήσει προσφορές από κατασκευαστές λογισμικού, και (γ) να καταρτίζει ρεαλιστικούς χρονικούς στόχους.

Η εκτίμηση του λογισμικού είναι μια σύνθετη και δύσκολη εργασία για τις περισσότερες ομάδες ανάπτυξης λογισμικού. Πολλές φορές, τα μέλη της ομάδας ανάπτυξης καταναλώνουν σημαντικό χρόνο για να αναλύσουν τα χαρακτηριστικά του υπό ανάπτυξη συστήματος, ώστε να δημιουργήσουν ακριβείς εκτιμήσεις. Η βασική λογική είναι ότι θα πρέπει να καταγράψουμε όλες τις αναγκαίες εργασίες, στην μεγαλύτερη δυνατή λεπτομέρεια και στη συνέχεια να συνδέσουμε αυτές τις εργασίες με μια εκτίμηση χρόνου. Δυστυχώς όμως κάθε κομμάτι λογισμικού είναι μοναδικό και είναι δύσκολο να εκτιμήσουμε πόσος χρόνος απαιτείται για κάτι που κατασκευάζεται για πρώτη φορά. Είναι επίσης δύσκολο να δούμε τους ανθρώπους, σαν μηχανές, που παράγουν για παράδειγμα 100 γραμμές κώδικα καθημερινά.

Διακρίνουμε τρεις μεγάλες κατηγορίες τεχνικών εκτίμησης κόστους:

1. Εκτίμηση από ειδικούς
2. Εκτίμηση με παραμετρικά μοντέλα
3. Εκτίμηση με βάση τις αναλογίες
4. Εκτίμηση με χρήση τεχνικών μηχανικής μάθησης.

Στην εκτίμηση από ειδικούς, οι εκτιμήσεις παράγονται με βάση αποκλειστικά την εμπειρία ενός ή περισσοτέρων ‘ειδικών’, δηλ. σχετικά έμπειρων στελεχών και τεχνολόγων λογισμικού. Δε χρησιμοποιούνται ιδιαίτερα καταγραμμένα στοιχεία για προγενέστερα, κοστολογημένα έργα λογισμικού (λέγονται ιστορικά

έργα λογισμικού). Η τεχνική εστιάζεται στο πως θα συνδυασθούν οι εκτιμήσεις περισσοτέρων του ενός ειδικών. Μία (αρκετά παλιά) σχετική τεχνική είναι αυτή των Δελφών. Η επιτυχία της τεχνικής των ειδικών εξαρτάται πολύ από τη διαθεσιμότητα του ειδικού ή των ειδικών και από την αξιοποίηση της εμπειρίας των ειδικών και τη σωστή υποκειμενική αξιολόγηση του υπό εκτίμηση έργου. Είναι η πιο διαδομένη μέθοδος στην πράξη, αλλά συνήθως εφαρμόζεται με μη συστηματικό τρόπο και οδηγεί συχνά σε πολύ λανθασμένες εκτιμήσεις.

Στην εκτίμηση με παραμετρικά μοντέλα, χρησιμοποιούνται μεμονωμένες εξισώσεις ή ομάδες εξισώσεων, οι οποίες, με βάση τιμές που δίδονται σε επιλεγμένα χαρακτηριστικά (παραμέτρους) του προς εκτίμηση έργου, υπολογίζουν το κόστος του. Οι εξισώσεις αυτές βασίζονται σε ένα σύνολο από ιστορικά έργα τα οποία έχουν χρησιμοποιηθεί για να επιλεγεί το είδος της σχέσης του κόστους με τις παραμέτρους του (π.χ. γραμμική, εκθετική σχέση) και οι συντελεστές βαρύτητας με τους οποίους κάθε παράμετρος συμμετέχει στον υπολογισμό του κόστους. Διακρίνουμε δύο κατηγορίες, τα μοντέλα παλινδρόμησης (regression models) και τα άτυπα μοντέλα (ad hoc models).

Η μέθοδος με αναλογίες στηρίζεται επίσης σε ένα σύνολο από ιστορικά έργα. Το προς εκτίμηση έργο συγκρίνεται με κάθε ένα ιστορικό έργο ξεχωριστά. Επιλέγονται ένα ή περισσότερα ιστορικά έργα ως "ανάλογα" με το προς εκτίμηση, και με βάση το καταγραμμένο κόστος τους γίνεται η εκτίμηση του κόστους του νέου έργου. Κρίσιμο σημείο της μεθόδου είναι η ύπαρξη έργου ή έργων που είναι παρόμοια με το προς εκτίμηση. Η μέθοδος των αναλογιών είναι η λιγότερο διαδομένη στην πράξη από τις τρεις βασικές τεχνικές.

Τέλος, η εκτίμηση με μοντέλα μηχανικής μάθησης μοιάζει με αυτή των παραμετρικών μοντέλων, με τη διαφορά ότι την εκτίμηση του νέου έργου αναλαμβάνουν τώρα δίκτυα (νευρωνικά, Bayes), κανόνες ή δένδρα κατηγοριοποίησης που έχουν εκπαιδευθεί προηγουμένως πάνω στα ιστορικά στοιχεία των έργων (Φιτσιλής et al., 2008; Sandeep, 2020).

Σε ένα παραδοσιακό έργο που ακολουθεί τον κύκλο ζωής καταρράκτη, η εργασία που πρέπει να γίνει αναλύεται σε δραστηριότητες, χρησιμοποιώντας μια δομή ανάλυσης εργασιών (Work Breakdown Structure). Στη συνέχεια για κάθε δραστηριότητα υπολογίζεται η αναγκαία προσπάθεια. Συνεπώς, ένα παραδοσιακό έργο σχεδιάζεται πλήρως στην έναρξη του έργου.

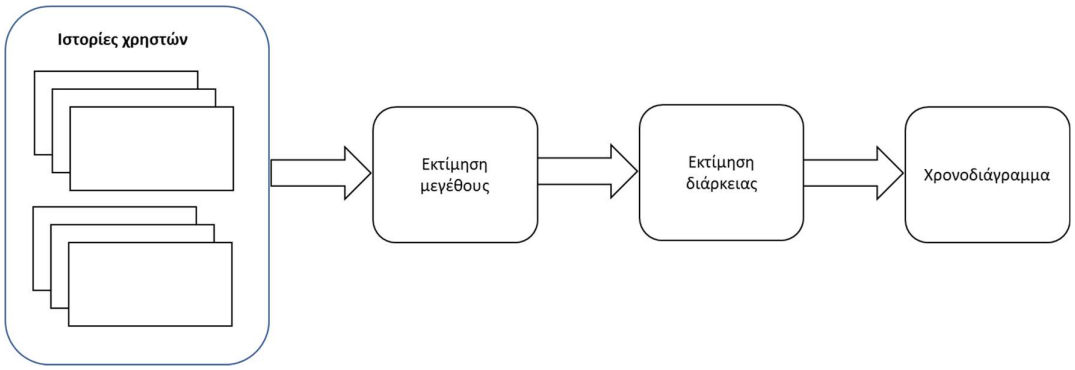
Αντίθετα σ' ένα έργο που ακολουθεί την ευέλικτη προσέγγιση, ο σχεδιασμός και η εκτίμηση της προσπάθειας είναι μια επαναληπτική διαδικασία, κατά την οποία τα στοιχεία του καταλόγου των απαιτήσεων εκτιμώνται με μεγαλύτερη ακρίβεια όταν είναι διαθέσιμες περισσότερες πληροφορίες.

Μια ευέλικτη ομάδα διαχωρίζει τις εκτιμήσεις μεγέθους από τις εκτιμήσεις της διάρκειας. Για να καταλάβουμε τη διαφορά, ας υποθέσουμε ότι πρέπει να μεταφέρουμε ένα αριθμό κιβωτίων από ένα κτήριο σε ένα άλλο. Θα μπορούσαμε να κοιτάξουμε τη στοίβα με τα κιβώτια, να αξιολογήσουμε τα εργαλεία που χρειαζόμαστε (π.χ. ένα καρότσι, ένα φορτηγό) και να υπολογίσουμε απευθείας τη δουλειά που χρειάζεται σε πέντε ώρες. Φτάνοντας σε αυτήν την εκτίμηση, έχουμε παρακάμψει οποιαδήποτε εκτίμηση μεγέθους και πήγαμε απευθείας σε μια εκτίμηση της διάρκειας.

Ας υποθέσουμε ότι κοιτάζουμε τη στοίβα των κιβωτίων και μετράμε τον αριθμό τους. Με βάση τις διαστάσεις τους υπολογίζουμε ότι η στοίβα έχει μέγεθος 10 κυβικά μέτρα συνολικά. Αυτή είναι η εκτίμησή μας για το μέγεθος αυτού του έργου. Όμως, η εκτίμηση του μεγέθους από μόνη της δεν είναι χρήσιμη σε αυτή την περίπτωση. Θέλουμε να μάθουμε πόσο καιρό θα χρειαστεί για να μετακινήσουμε τα κιβώτια. Πρέπει να μετατρέψουμε την εκτίμηση του μεγέθους (10 κυβικά μέτρα) σε εκτίμηση της διάρκειας.

Το διαθέσιμο φορτηγό έχει χωρητικότητα 2 κυβικά μέτρα. Συνεπώς θα χρειαστούμε 5 διαδρομές με το φορτηγό. Υπολογίζουμε ότι κάθε διαδρομή απαιτεί 20 λεπτά για να φορτωθεί το φορτηγό, 30 λεπτά για να μετακινηθεί, 20 λεπτά για την αποφόρτωση και 30 λεπτά για την επιστροφή. Ο συνολικός χρόνος ταξιδιού θα είναι 100 λεπτά. Εφόσον προβλέπουμε ότι θα απαιτηθούν 5 διαδρομές, η εκτίμησή μας για τη διάρκεια είναι 500 λεπτά.

Η διαδικασία αυτή παρουσιάζεται γραφικά στην Εικόνα 7.1



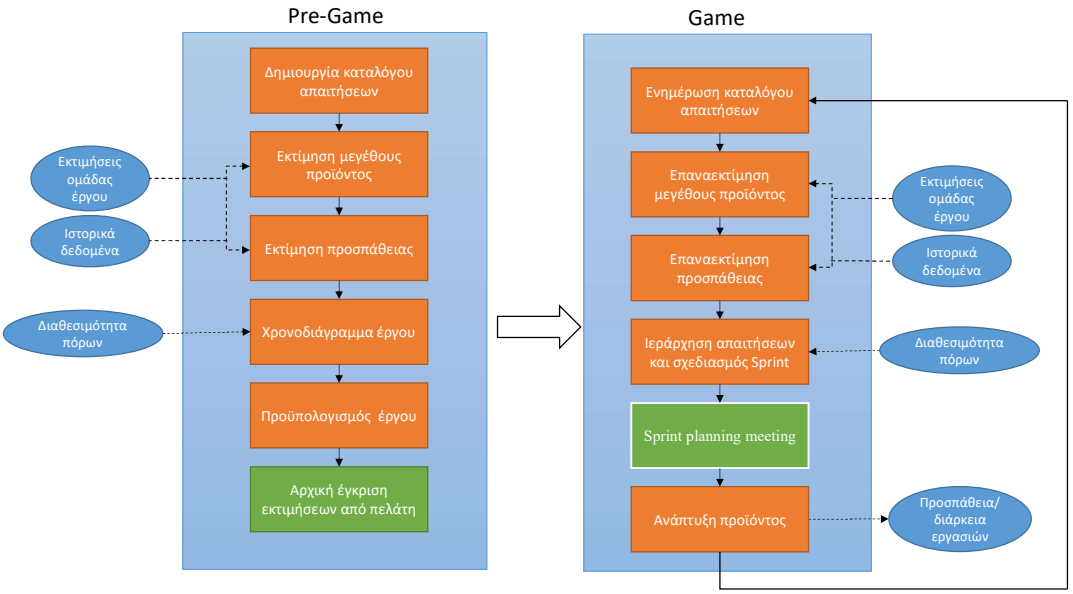
Εικόνα 7.1: *Η εκτίμηση της διάρκειας ενός έργου με το μέγεθος των ιστοριών χρηστών*

7.2.1 Τα αναγκαία βήματα για την εκτίμηση της προσπάθειας

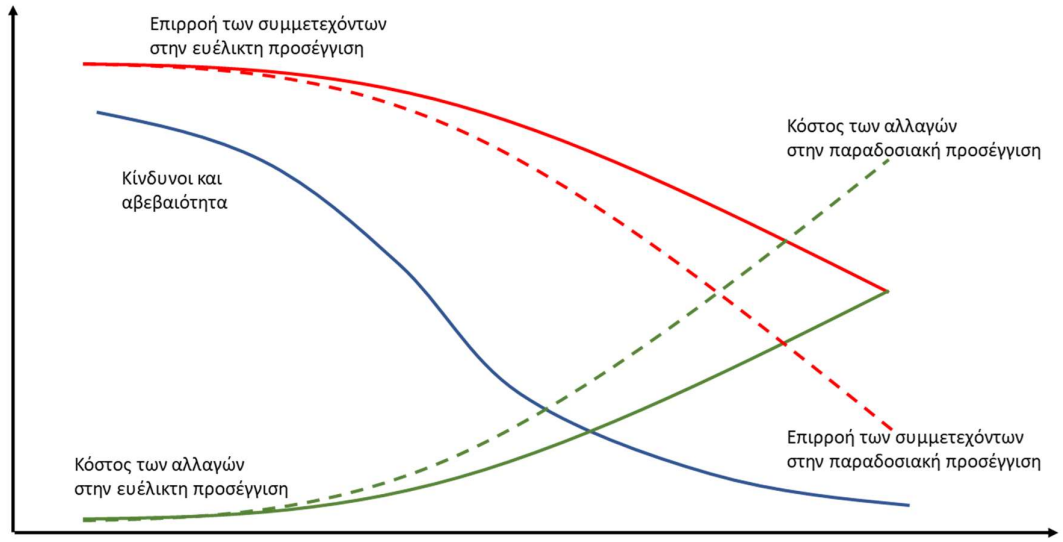
Όταν ξεκινά ένα έργο, η εικόνα που έχουμε για το έργο είναι περιορισμένη. Έχουμε μιλήσει αναλυτικά για τη φάση της της αρχικής διερεύνησης (pre-game) της διεργασίας Scrum. Θα υπενθυμίσουμε ότι η αρχική διερεύνηση στοχεύει στην ανάλυση του συστήματος και στον υψηλού επιπέδου σχεδιασμό. Κατά τη φάση της ανάλυσης (Βλέπε Εικόνα 4.3) γίνεται ο καθορισμός των προδιαγραφών του συστήματος καθώς και των απαιτήσεων του χρήστη από αυτό. Σημαντικό ρόλο στη διεργασία αυτή διαδραματίζει ο πελάτης, ο οποίος βρίσκεται σε διαρκή επικοινωνία με την ομάδα έργου και πρέπει να ορίσει το ποια θα είναι η λειτουργία του συστήματος. Το αποτέλεσμα αυτής της συνεργασίας μεταξύ πελάτη και ομάδας έργου είναι ένας κατάλογος χαρακτηριστικών του προϊόντος (product backlog list) που περιέχει όλες τις γνωστές, μέχρι την παρούσα στιγμή, απαιτήσεις. Προφανώς, η ύπαρξη ενός πλήρους καταλόγου χαρακτηριστικών είναι βασική προϋπόθεση για την εκτίμηση της προσπάθειας.

Η εκτίμηση είναι μια επαναληπτική διεργασία η οποία ξεκινά στην αρχική διερεύνηση και γίνεται κατά τη διάρκεια του sprint planning meeting. Αρχικά, θα πρέπει να εκτιμηθεί το μέγεθος του υπό κατασκευή προϊόντος με βάση τις απαιτήσεις, τα ιστορικά στοιχεία από άλλα έργα, αλλά και λαμβάνοντας υπόψη την εμπειρία των μελών της ομάδας έργου. Η αρχική διαστασιολόγηση του έργου είναι απολύτως αναγκαία καθώς ο πελάτης θα πρέπει να δεσμεύσει τους αναγκαίους πόρους (βλέπε Εικόνα 7.2).

Στη συνέχεια, η εκτίμηση του μεγέθους γίνεται κατά τη διάρκεια του κάθε sprint αφού όπως έχουμε πει ο κατάλογος των χαρακτηριστικών είναι κάτι που μπορεί να αλλάζει έως και το τέλος του έργου. Όμως καθώς το έργο προχωρά, η γνώση της ομάδας έργου μεγαλώνει ενώ αντίστοιχα μειώνεται η αβεβαιότητα. Στην Εικόνα 7.3 παρουσιάζεται η εξέλιξη των κινδύνων και της αβεβαιότητας μέσα στο έργο, η οποία μεταβάλλεται κατά τη διάρκεια του έργου.



Εικόνα 7.2: *Η διαδικασία της ευέλικτης διοίκησης*



Εικόνα 7.3: *Οι κίνδυνοι και η αβεβαιότητα μέσα στο έργο*

7.2.2 Η ιεράρχηση των απαιτήσεων

Η προτεραιότητα μιας απαίτησης υποδηλώνει τη σημασία της απαίτησης σε σχέση με ένα ή περισσότερα κριτήρια ιεράρχησης. Οι προτεραιότητες των απαιτήσεων μπορούν να προσδιοριστούν είτε για κάθε μια απαίτηση είτε με κατά ζεύγη συγκρίσεις απαιτήσεων. Το αποτέλεσμα της ιεράρχησης είναι μία ταξινόμηση των απαιτήσεων με βάση τα κριτήρια που επιλέχθηκαν στην αρχική φάση της ιεράρχησης. Μετά τη διαδικασία της ιεράρχησης, κάθε απαίτηση ανατίθεται σε μία «κλάση» προτεραιοτήτων. Στην κάθε κλάση προτεραιοτήτων δεν ορίζεται περαιτέρω ταξινόμηση (Pohl, 2010).

Οι λόγοι για τους οποίους ιεραρχούνται οι απαιτήσεις ποικίλλουν, και μπορεί να είναι, για παράδειγμα, ο καθορισμός της σειράς με την οποία θα εκτελεστούν οι απαιτήσεις, ή της σειράς με την οποία θα υλοποιηθούν απαιτήσεις διαφορετικών συμμετεχόντων-χρηστών (Εξάρχου, 2014).

Σύμφωνα με το Σώμα της Γνώσης για τους επιχειρηματικούς αναλυτές, το λεγόμενο Business Analysis Body of Knowledge (BABOK Guide 3.0) υπάρχουν οκτώ παράγοντες που επηρεάζουν την ιεράρχηση των απαιτήσεων. Πιο συγκεκριμένα:

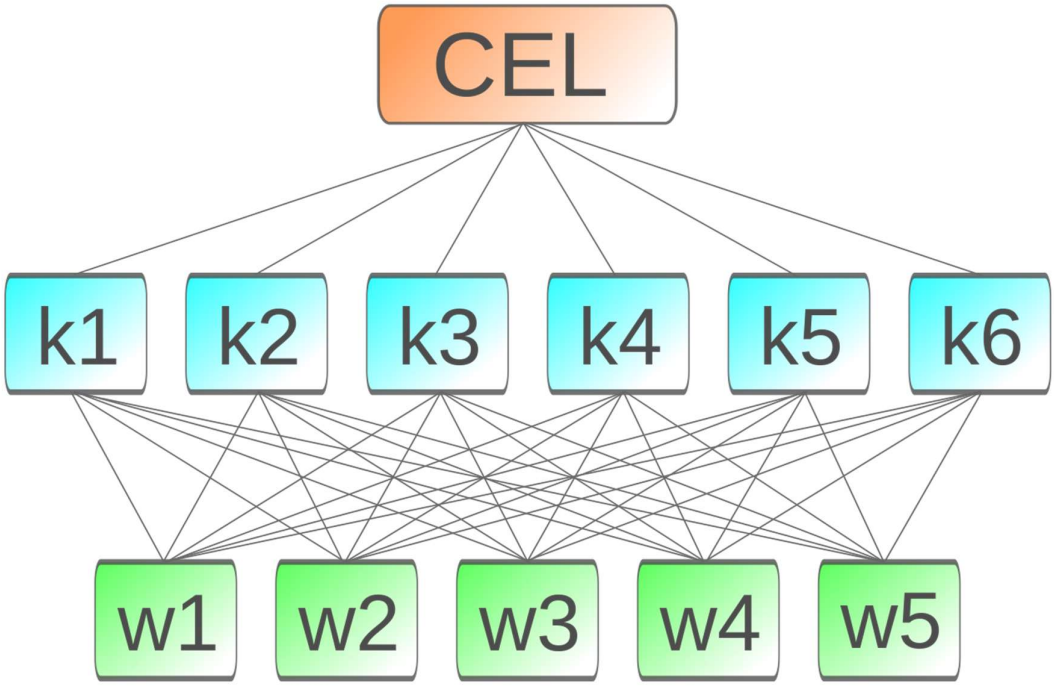
- Όφελος (benefit) – Αυτό είναι το όφελος που θα αποκομίσει η επιχείρηση εάν ικανοποιηθεί πρώτα και κύρια μια συγκεκριμένη ανάγκη. Το όφελος από την υλοποίηση μιας συγκεκριμένης απαίτησης όσον αφορά την ικανότητα, την ποιότητα, την τιμή, τον χρόνο και τις προμήθειες που γίνονται με πιο αποτελεσματικό τρόπο ή για την επίτευξη των εταιρικών στόχων.
- Ποινή (penalty) – Το μειονέκτημα ή η συνέπεια της μη συμμόρφωσης με μια απαίτηση αναφέρεται ως ποινή. Η μη ιεράρχηση των βασικών αναγκών μπορεί να έχει αρνητικό αντίκτυπο στο έργο και στο τελικό προϊόν. Ως αποτέλεσμα, η ποινή που πρέπει να καταβάλουν οι ομάδες του έργου και οι ενδιαφερόμενοι είναι ένα κρίσιμο αποφασιστικό στοιχείο για την ιεράρχηση των απαιτήσεων. Η ποινή δεν έχει πάντα οικονομικό χαρακτήρα αλλά θα μπορούσε να είναι η δυσαρέσκεια του πελάτη, η σπατάλη ή αντίστοιχα η υπερβολική δαπάνη των πόρων του έργου, η κακή λειτουργικότητα/χρηστικότητα του προϊόντος, κ.λπ..
- Κίνδυνος (risk)– Ο κίνδυνος παραγωγής της αναμενόμενης αξίας από την υλοποίηση της απαίτησης. Μια απαίτηση μπορεί να μην υλοποιηθεί πλήρως για διάφορους λόγους, όπως προβλήματα κατανόησης της απαίτησης, ελλιπής υλοποίηση, κ.λπ.
- Εξαρτήσεις (dependencies) – Η εξάρτηση είναι μια σχέση μεταξύ δύο απαιτήσεων στις οποίες η μία δεν μπορεί να ολοκληρωθεί ή να υλοποιηθεί σωστά χωρίς την άλλη. Συνεπώς, πριν από την έναρξη της ιεράρχησης των απαιτήσεων είναι αναγκαίο να έχουμε χάρτη/πίνακα εξαρτήσεων.
- Ευαισθησία στον χρόνο (time sensitivity) – Όταν πρόκειται για την ιεράρχηση των απαιτήσεων, το πιο σημαντικό ζήτημα είναι συνήθως τα χρονικά όρια που είναι διαθέσιμα. Ορισμένες απαιτήσεις είναι αρκετά επείγουσες και πρέπει να υλοποιηθούν πριν από άλλες. Αυτό είναι ιδιαίτερα σημαντικό στις περιπτώσεις έργων ή/και προϊόντων που έχουν εποχικές απαιτήσεις. Σε τέτοιες περιπτώσεις, η εκπλήρωση συγκεκριμένων απαιτήσεων μέχρι μια συγκεκριμένη ημερομηνία ή ώρα είναι κρίσιμη για την επιτυχία του έργου.
- Σταθερότητα (stability)– Ένα σημαντικό στοιχείο που επηρεάζει την ιεράρχηση των απαιτήσεων είναι η πιθανότητα αλλαγής της απαίτησης. Απαιτήσεις που δεν είναι σταθερές ή των οποίων ο ορισμός αλλάζει συχνά έχουν χαμηλότερη προτεραιότητα για να υλοποιηθούν στην επόμενη επανάληψη, διότι η εκ νέου επεξεργασία της απαίτησης αποτελεί σπατάλη χρόνου και πόρων.
- Κανονιστική / Συμμόρφωση με την Πολιτική (Regulatory/Policy Compliance) – Προκειμένου μια επιχείρηση ή τα ενδιαφερόμενα μέρη να συμμορφωθούν με τις αναγκαίες κανονιστικές ή με νομικές διατάξεις, θα πρέπει το υπο-ανάπτυξη προϊόν να υλοποιεί σχετικές απαιτήσεις. Γενικότερα, η συμμόρφωση με την πολιτική αναφέρεται στη συμμόρφωση ενός οργανισμού ή μιας επιχείρησης με τους νόμους, τους εμπορικούς κανονισμούς και άλλες απαιτήσεις εταιρικής πολιτικής στις καθημερινές του λειτουργίες.

Στη βιβλιογραφία υπάρχουν δεκάδες μέθοδοι ιεράρχησης των απαιτήσεων, η κάθε μια με τα επιμέρους χαρακτηριστικά της. Μερικές από τις πλέον γνωστές μέθοδοι είναι (Bukhsh et al., 2020):

- Αναλυτική Ιεραρχική Διεργασία - Analytical Hierarchical Process (AHP)
- Η μέθοδος των 100 δολαρίων
- Ταξινόμηση (ranking)
- Η αριθμητική ταξινόμηση (numerical grouping)
- Η μέθοδος του Wieger
- Η μέθοδος MoSCoW

- Κ.λπ.

Η **Αναλυτική Ιεραρχική Διεργασία** (AHP) αναπτύχθηκε από τον Saaty (1987) και είναι μία πολύ-κριτηριακή μέθοδος λήψης αποφάσεων που χρησιμοποιεί πίνακα δυαδικών συγκρίσεων για κάθε ζεύγος απαιτήσεων, ώστε να υπολογίσει τη σχετική αξία (relative importance) κάθε απαίτησης, σε σχέση με μία άλλη. Θεωρείται ότι αποτελεί μία από τις πιο διαδεδομένες μεθόδους πολύ-κριτηριακής λήψης αποφάσεων. Στη μέθοδο AHP πρώτα καθορίζεται ο στόχος και η δομή των στοιχείων σε σύνολα κριτηρίων, υποκριτηρίων και εναλλακτικών αξιών. Στη συνέχεια πραγματοποιούνται οι δυαδικές συγκρίσεις των στοιχείων και γίνεται η εκχώρηση των αξιών των συγκρίσεων, π.χ. εκχωρείται ο αριθμός 1 εάν το Α και το Β είναι της ίδιας σημαντικότητας, 9 εάν το Α είναι 9 φορές σημαντικότερο από το Β και 1/9 εάν το Α είναι το 9 φορές μικρότερης σημασίας από το Β. Τα αποτελέσματα τοποθετούνται σε ένα πίνακα, και υπολογίζεται το κανονικοποιημένο άνυσμα Eigen του πίνακα. Υπολογίζεται το βάρος και η κλίμακα συνέπειας και αξιολογούνται οι εναλλακτικές αξίες σε σχέση με το βάρος, για να ληφθεί η τελική βαθμολογία. Ένα σύντομο παράδειγμα παρουσιάζεται στην Εικόνα 7.4 όπου κ1, κ2, …, κ6 αναπαριστούν τα κριτήρια ενώ τα w1, w2, …, w5 τα εναλλακτικά ενδεχόμενα, τα οποία στην περίπτωσή μας είναι οι ιστορίες χρηστών. Το βασικό μειονέκτημα της μεθόδου είναι ότι απαιτείται μεγάλος αριθμός συγκρίσεων για την ιεράρχηση των απαιτήσεων.



Εικόνα 7.4: *Ενδεικτική εικόνα της μεθόδου AHP*

Η **δοκιμή των 100 δολαρίων** (100 dollar test) είναι μια πολύ απλή τεχνική ιεράρχησης προτεραιοτήτων όπου δίνονται στους συμμετέχοντες 100 φανταστικές μονάδες (χρήματα, ώρες κ.λπ.) για να τις κατανείμουν μεταξύ των απαιτήσεων. Το αποτέλεσμα της ιεράρχησης παρουσιάζεται με μια αναλογική κλίμακα (ratio scale). Ένα πρόβλημα της τεχνικής είναι στην περίπτωση που υπάρχουν πάρα πολλές απαιτήσεις για ιεράρχηση. Για παράδειγμα, εάν υπάρχουν 25 απαιτήσεις, και συνεπώς υπάρχουν κατά μέσο όρο τέσσερις βαθμοί για να διανείμουμε σε κάθε απαίτηση. Στην περίπτωση αυτή χρησιμοποιούμε ένα μεγαλύτερο πλασματικό ποσό π.χ. 100.000 δολαρίων ώστε οι συμμετέχοντες να έχουν περισσότερη ελευθερία στις προτεραιότητες. Ένα άλλο πρόβλημα της μεθόδου είναι η χρήση μεγάλου ποσού σε λίγες απαιτήσεις με σκοπό δώσουν στις «αγαπημένες τους» απαιτήσεις κορυφαία προτεραιότητα. Η λύση στο πρόβλημα αυτό είναι ο περιορισμός του ποσού που δαπανάται για μεμονωμένες απαιτήσεις.

Η απλή ταξινόμηση (ranking) των απαιτήσεων βασίζεται σε μια διατεταγμένη κλίμακα (ordinal scale) όπου οι απαιτήσεις κατατάσσονται χωρίς ισοβαθμίες στην κατάταξη. Αυτό σημαίνει ότι η πιο σημαντική απαίτηση κατατάσσεται στην 1η θέση και η λιγότερο σημαντική στη θέση n (για n απαιτήσεις).

Κάθε απαίτηση έχει μια μοναδική θέση στην κατάταξη, αλλά δεν είναι δυνατό να φανεί η σχετική διαφορά μεταξύ δύο απαιτήσεων (όπως μπορεί να γίνει σε άλλες μεθόδους όπως με την AHP ή στη δοκιμή των 100 δολαρίων). Η κατάταξη μπορεί να γίνει με διάφορους τρόπους, όπως για παράδειγμα με τη χρήση αλγορίθμων ταξινόμησης με φυσαλίδες (bubble sort) ή με χρήση δυαδικού δέντρου αναζήτησης (binary search tree) (Karlsson et al., 1998). Ανεξάρτητα από τον αλγόριθμο ταξινόμησης, η απλή ταξινόμηση φαίνεται να είναι πιο κατάλληλη όταν υπάρχει ένας συμμετέχων/χρήστης, επειδή είναι δύσκολο να ευθυγραμμιστούν οι απόψεις πολλών διαφορετικών συμμετεχόντων.

Η **αριθμητική ταξινόμηση (numerical grouping)** είναι μια από τις πιο διαδεδομένες τεχνικές και βασίζεται στην ομαδοποίηση των απαιτήσεων σε διαφορετικές ομάδες προτεραιότητας. Ο αριθμός των ομάδων προτεραιότητας μπορεί να ποικίλλει, αλλά στην πράξη, η ύπαρξη τριών ομάδων είναι η πιο συνηθισμένη περίπτωση. Για παράδειγμα, οι απαιτήσεις μπορούν να ομαδοποιηθούν σε τρεις ομάδες, όπως:

- κρίσιμη προτεραιότητα,
- κανονική προτεραιότητα και
- προαιρετικές.

Εναλλακτικά, οι ενδιαφερόμενοι μπορούν να ταξινομήσουν τις απαιτήσεις ως υποχρεωτικές, πολύ σημαντικές, μάλλον σημαντικές, μη σημαντικές. Η μέθοδος δεν απαιτεί την τεκμηρίωση των επιλογών των συμμετεχόντων. Αντίστοιχα, η χρήση όρων όπως υψηλή, μεσαία και χαμηλή προτεραιότητα δεν αποτελεί καλή επιλογή αφού μπορεί να προκαλέσει σύγχυση στους ενδιαφερόμενους.

Η **μέθοδος ιεράρχησης με πίνακα (matrix prioritization)** είναι μια τεχνική ιεράρχησης πολλαπλών παραγόντων, γενικής χρήσης, που έχει ως αποτέλεσμα μια αναλογική κλίμακας (ratio scale)**.** Η πιο κοινή εφαρμογή αυτής της τεχνικής είναι πιθανώς ο Wiegers Prioritization Matrix (WPM) (Wieger, 1999; Wieger, 2005).

Ο πίνακας ιεράρχησης (Prioritization Matrix) αποτελείται από πέντε συστατικά, τα οποία είναι:

- Ένα **σύνολο αντικειμένων** προς αξιολόγηση με σκοπό την ιεράρχηση των προτεραιοτήτων. Τα αντικείμενα προς ιεράρχηση μπορεί να είναι έργα, χαρακτηριστικά, απαιτήσεις, ιστορίες χρηστών, κ.λπ.
- Τα διαφορετικά **κριτήρια** που θα χρησιμοποιηθούν για την αξιολόγηση του κάθε στοιχείου. Όλα τα κριτήρια ισχύουν/πρέπει να έχουν εφαρμογή για όλα τα αξιολογούμενα στοιχεία. Τα κριτήρια πρέπει να είναι κατάλληλα αλλά και σημαντικά για τα αντικείμενα που αξιολογούμε. Η παρακάτω λίστα περιέχει μερικά από τα κοινά κριτήρια που αναφέρονται στη βιβλιογραφία:
    - Σχετικό όφελος (για την επιχείρηση, τον πελάτη, τα έσοδα κ.λπ.)
    - Σχετική ποινή
    - Σχετικό κόστος (υλοποίησης)
    - Σχετικός κίνδυνος (εφαρμογής, σε σχέση με το κόστος)
    - Σχετική Σημασία / Κρισιμότητα (προς επιτυχία, για την κάλυψη των απαιτήσεων του κανονιστικού πλαισίου κ.λπ.)
    - Βαθμός Στρατηγικής Ευθυγράμμισης (πόσο καλά ταιριάζει το αντικείμενο με τους τακτικούς ή στρατηγικούς στόχους του οργανισμού)
    - Απαιτούμενος χρόνος υλοποίησης
    - % των χρηστών που επηρεάστηκαν

- Απαιτούμενοι πόροι υλοποίησης
- Μεταβλητότητα (πιθανότητα αλλαγής του στοιχείου ή πιθανότητα αλλαγής βαθμολογίας στοιχείου, π.χ. αλλάζει η αξία της απαίτησης για την επιχείρηση)
- Αποτελεί επείγουσα ανάγκη (πόσο σύντομα χρειάζεται το στοιχείο;)

- **Κλίμακες τιμών** - Πρόκειται για την κλίμακα των πιθανών τιμών που μπορούν να χρησιμοποιηθούν για τον καθορισμό της σχέσης ενός στοιχείου με τα κριτήρια που αξιολογούνται. Οι κλίμακες τιμών χρησιμοποιούνται για την αξιολόγηση κάθε στοιχείου σε σχέση με ένα μεμονωμένο κριτήριο. Η πιο κοινή κλίμακα είναι μια διατεταγμένη κλίμακα από το 1 έως το 9 ή από το 0 έως το 5. Τα βασικά πράγματα που πρέπει να λαμβάνει κανείς υπόψη σε σχέση με τις κλίμακες τιμών είναι:
  - Ο αριθμός των επιλογών θα πρέπει να είναι σχετικά μικρός. Ένα ευρύ φάσμα επιλογών οδηγεί σε συζήτηση για μικρές διαφορές.
  - Κάθε τιμή στην κλίμακα τιμών θα πρέπει να συνδέεται με έναν σχετικά συγκεκριμένο ορισμό του τι σημαίνει αυτή η τιμή στο πλαίσιο των κριτηρίων που αξιολογούνται.

  Για παράδειγμα, η παρακάτω κλίμακα αναφέρεται στην αξιολόγηση του κριτηρίου *σχετικό όφελος*:
  - 0 = Κανένας χρήστης δεν θα θεωρούσε χρήσιμη αυτή τη δυνατότητα
  - 1 = Αυτή η δυνατότητα θα ήταν κάπως χρήσιμη σε μικρό αριθμό χρηστών
  - 2 = Αυτή η δυνατότητα θα ήταν κάπως χρήσιμη σε πολλούς χρήστες
  - 3 = Αυτή η δυνατότητα θα ήταν χρήσιμη σε πολλούς χρήστες
  - 4 = Αυτή η δυνατότητα θα ήταν πολύ χρήσιμη σε ορισμένους χρήστες
  - 5 = Αυτή η δυνατότητα θα ήταν πολύ χρήσιμη σε πολλούς χρήστες

  Στην ιδανική περίπτωση, θα πρέπει να ορίσουμε συγκεκριμένα κριτήρια ώστε να αποφασίσουμε τι σημαίνει μια συγκεκριμένη τιμή για μια δεδομένη κατηγορία. Αυτό ισχύει ιδιαίτερα για κριτήρια όπως η προσπάθεια υλοποίησης ή το κόστος υλοποίησης, όπου είναι εύκολο να χρησιμοποιηθούν αριθμητικές τιμές. Για παράδειγμα για την προσπάθεια υλοποίησης θα μπορούσαμε να χρησιμοποιήσουμε κόστος σε ευρώ, ανθρωποώρες ή τον αριθμό των αναγκαίων πόρων. Αν χρησιμοποιήσουμε ανθρωποώρες μπορούμε να αντιστοιχίσουμε για παράδειγμα το 0 σε 0-50 ανθρωποώρες, το 1 σε 51-150 ανθρωποώρες κ.λπ.

- **Σταθμίσεις/βάρη** - Η σχετική στάθμιση/βάρος του κάθε κριτήριου. Τα βάρη μας επιτρέπουν να δίνουμε προτεραιότητα σε ορισμένα κριτήρια σε σχέση με άλλα ή εάν υπάρχουν ξεχωριστές αξιολογήσεις από πολλούς συμμετέχοντες, μπορούμε να σταθμίσουμε τη σχετική βαρύτητα της άποψης του κάθε συμμετέχοντα, έτσι ώστε ορισμένοι να έχουν μεγαλύτερη επιρροή από άλλους στο τελικό αποτέλεσμα.

  Ο Wiegers δίνει το απλό παράδειγμα της στάθμισης του οφέλους που προκύπτει από την υλοποίηση μιας απαίτησης ως διπλάσιο από την ποινή που προκύπτει από την έλλειψη αυτής και συνεπώς τα οφέλη λαμβάνουν στάθμιση 2 ενώ οι ποινές 1. Μια εναλλακτική προσέγγιση είναι η χρήση ποσοστών σε συνδυασμό με τη διασφάλιση ότι οι συντελεστές στάθμισης για όλα τα κριτήρια συνοψίζονται στο 100%. Η ιδέα είναι ότι στα κριτήρια που είναι πιο σημαντικά δίνεται μεγαλύτερη βαρύτητα, έτσι ώστε να έχουν μεγαλύτερο αντίκτυπο στον υπολογισμό της ιεράρχησης.

- **Μαθηματικός τύπος/φόρμουλα** - Δεν υπάρχει συγκεκριμένος τύπος που χρησιμοποιείται για την ιεράρχηση πίνακα. Ο Wiegers στη μέθοδό του προτείνει τον τύπο:

$$Προτεραιότητα = \frac{\text{Σταθμισμένο όφελος \%} + \text{Σταθμισμένη ποινή \%}}{\text{Σταθμισμένο κόστος \%} + \text{Σταθμισμένος κίνδυνος \%}}$$

Μια άλλη προσέγγιση είναι η ιδέα της αξιολόγησης της εμπιστοσύνης ή της μεταβλητότητας για οποιαδήποτε μεταβλητή που αφορά την υλοποίηση, όπως για παράδειγμα το κόστος, τον κίνδυνο, τις απαιτήσεις σε πόρους κ.λπ. Αυτό θα επέτρεπε στην ομάδα υλοποίησης να αξιολογήσει ότι το κόστος είναι πιθανώς 5 (ό,τι σημαίνει αυτό για την κλίμακα αξιολόγησης), αλλά η εμπιστοσύνη της ότι αυτή είναι η σωστή βαθμολογία να είναι μόνο 1. Εάν αυτή η τιμή εμπιστοσύνης αντιστοιχεί σε ποσοστό (οπότε η βαθμολογία 1 σημαίνει 10% εμπιστοσύνη) και πολλαπλασιάσουμε τις δύο τιμές, οι ιστορίες χρήστη υψηλότερης εμπιστοσύνης έχουν υψηλότερη προτεραιότητα.

Ακριβώς όπως και με τα κριτήρια, τη στάθμιση και τις κλίμακες αξιολόγησης, η ιδέα είναι να καταλήξουμε σε ένα μαθηματικό τύπο με τον οποίο συμφωνούν όλα τα μέρη.

Η μέθοδος Wieger είναι μια μέθοδος που ενσωματώνει δύο οπτικές γωνίες: α) την οπτική γωνία των χρηστών και β) την οπτική γωνία της ομάδας ανάπτυξης Η οπτική γωνία των χρηστών περιλαμβάνει τις διαστάσεις:

- του Επιχειρηματικού Οφέλους (Benefit)
- της Επιχειρηματικής Ποινής (Penalty)

Η οπτική γωνία της ομάδας ανάπτυξης περιλαμβάνει τις διαστάσεις

- του Σχετικού Κόστους (Relative Cost) και
- του Σχετικού Κινδύνου (Relative Risk)

Για την εφαρμογή του μοντέλου ακολουθήστε τα οκτώ παρακάτω βήματα:

Βήμα 1. Φτιάξτε τον κατάλογο των απαιτήσεων του προϊόντος.

Βήμα 2. Εκτιμήστε το σχετικό επιχειρηματικό όφελος (benefit) της κάθε απαίτησης, που η απαίτηση αυτή παρέχει στον πελάτη ή στην επιχείρηση σε μια κλίμακα 1-9, (με το 1 να δείχνει πολύ μικρό όφελος και 9 να είναι το μέγιστο δυνατό όφελος). Εάν η αξιολόγηση γίνεται από περισσότερους του ενός πελάτες/χρήστες, τότε ο κάθε ένας από αυτούς θα πρέπει να παρέχει μια ανεξάρτητη ατομική αξιολόγηση. Στη γενική περίπτωση κάθε μέλος της ομάδας αξιολόγησης μπορεί να έχει διαφορετική βαρύτητα. Αυτό μπορεί να γίνει αλλάζοντας το σχετικό βάρος του κάθε χρήστη.

Βήμα 3. Υπολογίστε τη σχετική επιχειρηματική ποινή (penalty) που ο πελάτης ή η επιχείρηση θα έχουν εάν η δυνατότητα/απαίτηση δεν συμπεριληφθεί στην επανάληψη. Και πάλι, χρησιμοποιούμε την ίδια κλίμακα από 1-9, όπου το 1 σημαίνει ουσιαστικά καμία ποινή και 9 δείχνει ένα πολύ σοβαρό μειονέκτημα.

Βήμα 4. Η συνολική αξία (value) είναι το άθροισμα του σχετικού επιχειρηματικού οφέλους και της σχετικής επιχειρηματικής ποινής. Συνήθως, τα οφέλη σταθμίζονται με 2, ενώ η ποινή με 1. Εναλλακτικά, και ανάλογα με το είδος του έργου θα μπορούσαν να σταθμίζονται εξίσου.

$$\text{ΣΣΟ} = \text{Συντελεστής Στάθμησης Οφέλους}$$

$$\text{ΣΣΠ} = \text{Συντελεστής Στάθμησης Ποινής}$$

$$Αξία = \text{όφελος \%} * \text{ΣΣΟ} + \text{ποινή \%} * \text{ΣΣΠ}$$

Βήμα 5. Εκτιμήστε το σχετικό κόστος (relative cost) της κάθε απαίτησης, και πάλι σε μια κλίμακα που κυμαίνεται από το χαμηλό του 1 έως το υψηλό των 9. Η αξιολόγηση αυτή γίνεται από την

ομάδα έργου και βασίζεται σε παράγοντες όπως η πολυπλοκότητα, η έκταση της εργασίας, η διεπαφή χρήστη που απαιτείται, η ικανότητα επαναχρησιμοποίησης κ.α. Πολλές φορές ως κόστος μπορεί να χρησιμοποιηθούν τα σημεία ιστορίας (βλέπε επόμενη παράγραφο).

Βήμα 6. Η ομάδα έργου εκτιμά τους τεχνικούς ή άλλους κινδύνους (relative risk) που συνδέονται με κάθε απαίτηση σε μια κλίμακα από το 1 έως το 9. Μια εκτίμηση του 1 σημαίνει ότι είναι πολύ εύκολο να υλοποιηθεί, ενώ 9 δείχνει σοβαρές ανησυχίες για τη σκοπιμότητα, την ύπαρξη της απαιτούμενης τεχνογνωσίας, τη χρήση των απαιτούμενων εργαλείων, κ.α.

Συνήθως, το κόστος και ο κίνδυνος έχουν την ίδια βαρύτητα με τις διαστάσεις του επιχειρηματικού οφέλους και της επιχειρηματικής ποινής.

Βήμα 7. Υπολογισμός της προτεραιότητας για κάθε χαρακτηριστικό ή ιστορία χρήστη. Ο τύπος για την προτεραιότητα είναι:

$$\text{ΣΣΚ} = \text{Συντελεστής Στάθμησης Κόστους}$$

$$\text{ΣΣΚι} = \text{Συντελεστής Στάθμησης Κινδύνου}$$

$$Προτεραιότητα = \frac{\text{Αξία}}{\text{Κόστος \%} * \text{ΣΣΚ} + \text{Κίνδυνος \%} * \text{ΣΣΚι}}$$

Βήμα 8. Ταξινομήστε τη λίστα των χαρακτηριστικών/ιστοριών χρηστών σε φθίνουσα τάξη, ώστε να υπολογίσετε την κάθε προτεραιότητα. Οι απαιτήσεις στην κορυφή της λίστας έχουν τον πιο ευνοϊκό συνδυασμό αξίας, ποινής, κόστους και κινδύνου, και ως εκ τούτου θα πρέπει να έχουν μεγαλύτερη προτεραιότητα στην υλοποίηση.

Παράδειγμα της μεθόδου Wieger

Στην παρακάτω εικόνα παρουσιάζεται ένα σύνολο απαιτήσεων, η αξία των οποίων αξιολογείται από τρεις χρήστες/πελάτες:

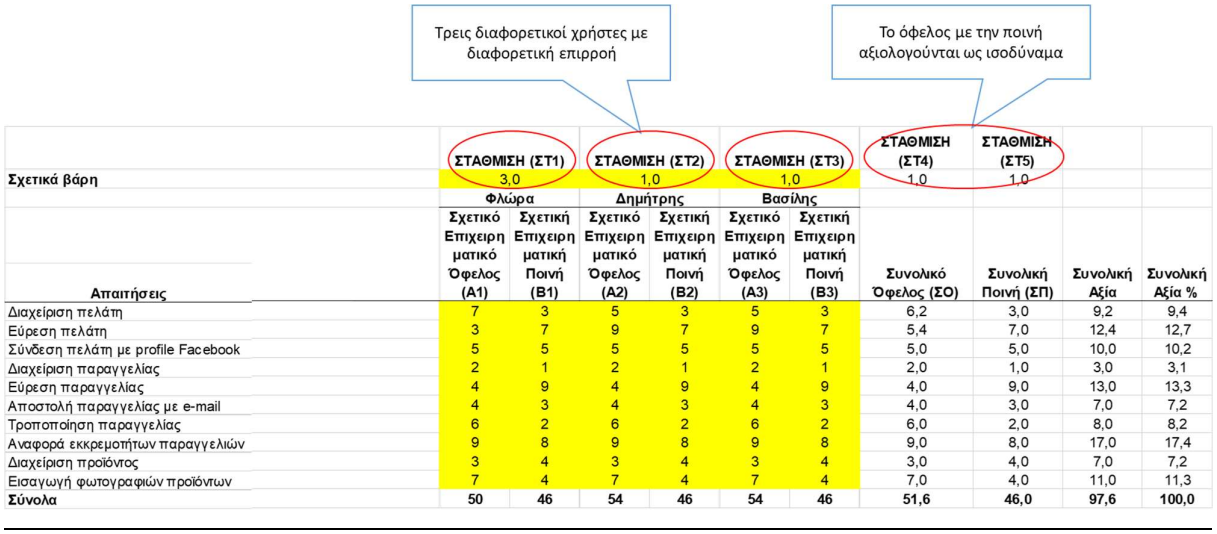
Α) Την Φλώρα που έχει σχετική επιρροή 3

Β) Το Δημήτρη και το Βασίλη με σχετική επιρροή 1

Η αξιολόγηση γίνεται χρησιμοποιώντας μια σχετική κλίμακα από 1 έως 9, όπου το 9 αντιπροσωπεύει την ισχυρότερη επιρροή. Το συνολικό όφελος και η συνολική ποινή προκύπτει από το συνδυασμό των βαθμολογιών των τριών πελατών προκύπτει από τους ακόλουθους τύπους αντίστοιχα (βλέπε Εικόνα 7.5):

$$\text{Συνολικό όφελος} = \frac{(A1 * ΣT1) + (A2 * ΣT2) + (A3 * ΣT3)}{ΣT1 + ΣT2 + ΣT3}$$

$$\text{Συνολική ποινή} = \frac{(B1 * ΣT1) + (B2 * ΣT2) + (B3 * ΣT3)}{ΣT1 + ΣT2 + ΣT3}$$



	ΣΤΑΘΜΙΣΗ (ΣΤ1)		ΣΤΑΘΜΙΣΗ (ΣΤ2)		ΣΤΑΘΜΙΣΗ (ΣΤ3)		ΣΤΑΘΜΙΣΗ (ΣΤ4)	ΣΤΑΘΜΙΣΗ (ΣΤ5)		
Σχετικά βάρη	3,0		1,0		1,0		1,0	1,0		
	Φλώρα		Δημήτρης		Βασίλης					
Απαιτήσεις	Σχετικό Επιχειρηματικό Όφελος (Α1)	Σχετική Επιχειρηματική Ποινή (Β1)	Σχετικό Επιχειρηματικό Όφελος (Α2)	Σχετική Επιχειρηματική Ποινή (Β2)	Σχετικό Επιχειρηματικό Όφελος (Α3)	Σχετική Επιχειρηματική Ποινή (Β3)	Συνολικό Όφελος (ΣΟ)	Συνολική Ποινή (ΣΠ)	Συνολική Αξία	Συνολική Αξία %
Διαχείριση πελάτη	7	3	5	3	5	3	6,2	3,0	9,2	9,4
Εύρεση πελάτη	3	7	9	7	9	7	5,4	7,0	12,4	12,7
Σύνδεση πελάτη με profile Facebook	5	5	5	5	5	5	5,0	5,0	10,0	10,2
Διαχείριση παραγγελίας	2	1	2	1	2	1	2,0	1,0	3,0	3,1
Εύρεση παραγγελίας	4	9	4	9	4	9	4,0	9,0	13,0	13,3
Αποστολή παραγγελίας με e-mail	4	3	4	3	4	3	4,0	3,0	7,0	7,2
Τροποποίηση παραγγελίας	6	2	6	2	6	2	6,0	2,0	8,0	8,2
Αναφορά εκκρεμοτήτων παραγγελιών	9	8	9	8	9	8	9,0	8,0	17,0	17,4
Διαχείριση προϊόντος	3	4	3	4	3	4	3,0	4,0	7,0	7,2
Εισαγωγή φωτογραφιών προϊόντων	7	4	7	4	7	4	7,0	4,0	11,0	11,3
Σύνολα	**50**	**46**	**54**	**46**	**54**	**46**	**51,6**	**46,0**	**97,6**	**100,0**

Εικόνα 7.5: *Υπολογισμός συνολικού οφέλους και ποινής στη μέθοδο Wieger*

Στη συνέχεια υπολογίζουμε το σχετικό κόστος και το σχετικό κίνδυνο χρησιμοποιώντας την ίδια λογική (βλέπε Εικόνα 7.6). Ο υπολογισμός του κόστους μπορεί να γίνει είτε ατομικά είτε από όλα τα μέλη της ομάδας ανάπτυξης.

	ΣΤΑΘΜΙΣΗ (ΣΤ6)		ΣΤΑΘΜΙΣΗ (ΣΤ7)		
Σχετικά βάρη	1,0		1,0		
Απαιτήσεις	**Σχετικό Κόστος**	**Κόστος %**	**Σχετικός Κίνδυνος**	**Κίνδυνος %**	**Προτεραιότητα**
Διαχείριση πελάτη	2	4,8	1	3,0	1,21
Εύρεση πελάτη	5	11,9	3	9,1	0,61
Σύνδεση πελάτη με profile Facebook	3	7,1	2	6,1	0,78
Διαχείριση παραγγελίας	1	2,4	1	3,0	0,57
Εύρεση παραγγελίας	4	9,5	4	12,1	0,62
Αποστολή παραγγελίας με e-mail	3	7,1	2	6,1	0,54
Τροποποίηση παραγγελίας	4	9,5	3	9,1	0,44
Αναφορά εκκρεμοτήτων παραγγελιών	7	16,7	8	24,2	0,43
Διαχείριση προϊόντος	4	9,5	2	6,1	0,46
Εισαγωγή φωτογραφιών προϊόντων	9	21,4	7	21,2	0,26
Σύνολα	**42**	**100,0**	**33**	**100,0**	

Εικόνα 7.6: *Υπολογισμός προτεραιότητας στη μέθοδο Wieger*

Η **μέθοδος MoSCoW** είναι μια τεχνική που είναι παρόμοια με την αριθμητική ταξινόμηση (Kravchenko, et al., 2022). Στη μέθοδο MoSCoW οι απαιτήσεις κατηγοριοποιούνται σε τέσσερις κατηγορίες:

Α. Πρέπει να υπάρχει-υλοποιηθεί (**Must have**). Παρέχει το ελάχιστα αποδεκτό υποσύνολο απαιτήσεων (Minimum Usable Subset - MUS) που είναι αναγκαίο να υλοποιηθούν με την παράδοση του προϊόντος και περιέχει τις ιστορίες χρηστών με τη μεγαλύτερη αξία για τους χρήστες. Παραδείγματα τέτοιων περιπτώσεων είναι απαιτήσεις που αν παραληφθούν το σύστημα δεν είναι ασφαλές, δεν καλύπτει τις νομικές δεσμεύσεις που απαιτούνται, δεν μπορεί να γίνει η αναγκαία εργασία, κ.λπ. Για να αποφασίσουμε αν η απαίτηση είναι Must have θα πρέπει να απαντήσουμε στη στοιχειώδη ερώτηση «Τι θα συμβεί στην περίπτωση ΜΗ υλοποίησης της

απαίτησης». Εάν η απάντηση στο ερώτημα είναι η "ακύρωση του έργου αφού δεν έχει νόημα να υλοποιηθεί μια λύση που δεν πληροί αυτήν την απαίτηση", τότε είμαστε βέβαιοι ότι η απαίτηση είναι Must Have. Εάν όμως υπάρχει κάποιος τρόπος για να γίνει η εργασία στην οποία αναφέρεται η ιστορία χρήστη, ακόμα με χειροκίνητο τρόπο, τότε η απαίτηση μπορεί να υποβαθμιστεί σε Should Have ή σε Could Have. Η υποβάθμιση μιας απαίτησης σε Should Have ή σε Could Have, δεν σημαίνει αναγκαστικά ότι δεν θα υλοποιηθεί, αλλά ότι η υλοποίηση δεν είναι εγγυημένη.

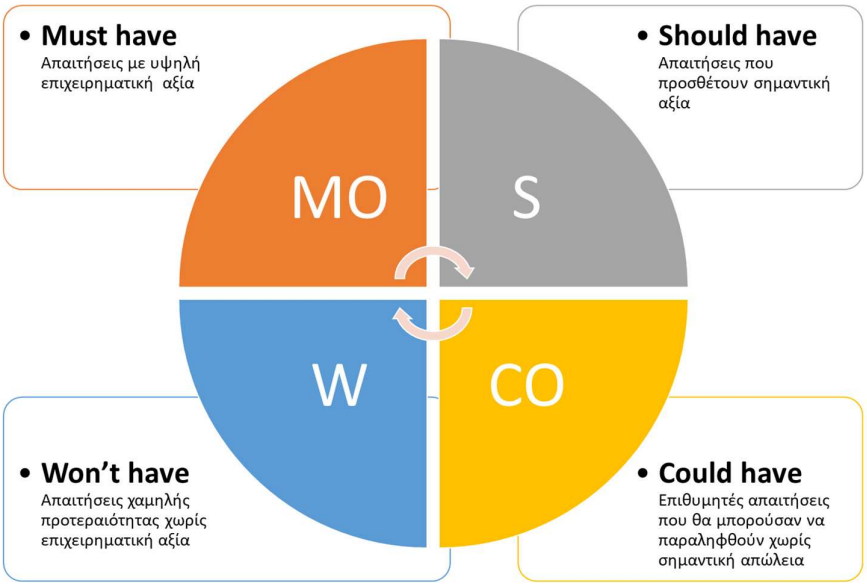
Β. Θα έπρεπε να υπάρχει-υλοποιηθεί (**Should have**). Οι απαιτήσεις αυτής της κατηγορίας αποτελούν σημαντικές απαιτήσεις, αλλά δεν είναι ζωτικής σημασίας στην υλοποίηση. Η μη υλοποίησή τους έχει κόστος, είναι επώδυνη για τους χρήστες, αλλά το έργο παραμένει βιώσιμο καθότι η ποινή/κόστος της μη υλοποίησης μπορεί να απορροφηθεί, σε όρους εμπορικής, επιχειρηματικής αξίας.

Γ. Θα μπορούσε να υπάρχει-υλοποιηθεί (**Could have**). Η απαίτηση είναι επιθυμητή, αλλά η υλοποίησή της θα πρέπει να αξιολογηθεί σε σχέση με το κόστος υλοποίησης.

Δ. Δεν θα υπάρχει-υλοποιηθεί (**Won't have**). Αποτελούν απαιτήσεις που δεν προσθέτουν αξία για την επιχείρηση, δηλαδή όλες οι απαιτήσεις οι οποίες δεν χρειάζεται να υλοποιηθούν τόσο στο πλαίσιο του συγκεκριμένου έργου. Η κατηγορία αυτή βοηθά σημαντικά στον περιορισμό των απαιτήσεων των χρηστών και βοηθά στην καλύτερη διαχείριση των προσδοκιών σχετικά με τι θα συμπεριληφθεί ή όχι σε κάθε επαναληπτικό κύκλο υλοποίησης. Επίσης προφυλάσσει το έργο από πιθανό εκτροχιασμό (scope creep).

Θα πρέπει να σημειωθεί ότι η τεχνική MoSCoW δεν είναι τμήμα των ευέλικτων μεθόδων, αλλά χρησιμοποιείται ευρύτατα σε αυτά (Agile).

Η τεχνική είναι εύκολη και γρήγορη στην εφαρμογή της, παρουσιάζει υψηλό βαθμό συνέπειας αποτελεσμάτων, αλλά η ευχρηστία της αφορά μόνο τα πρώτα στάδια ενός έργου, όταν δηλαδή δεν έχουν καθορισθεί οι απαιτήσεις με περισσότερη λεπτομέρεια. Επίσης μειονεκτεί στο γεγονός ότι το τελικό αποτέλεσμα είναι η ταξινόμηση των απαιτήσεων σε ομάδες, καθώς εντός αυτών των συνόλων-κατηγοριών, όλες οι απαιτήσεις θεωρούνται ίδιας ιεράρχησης-προτεραιότητας. Δηλαδή, δεν βοηθά στην επιλογή μεταξύ πολλαπλών απαιτήσεων με την ίδια προτεραιότητα.



Εικόνα 7.7: *Η μέθοδος ιεράρχησης MoSCoW*

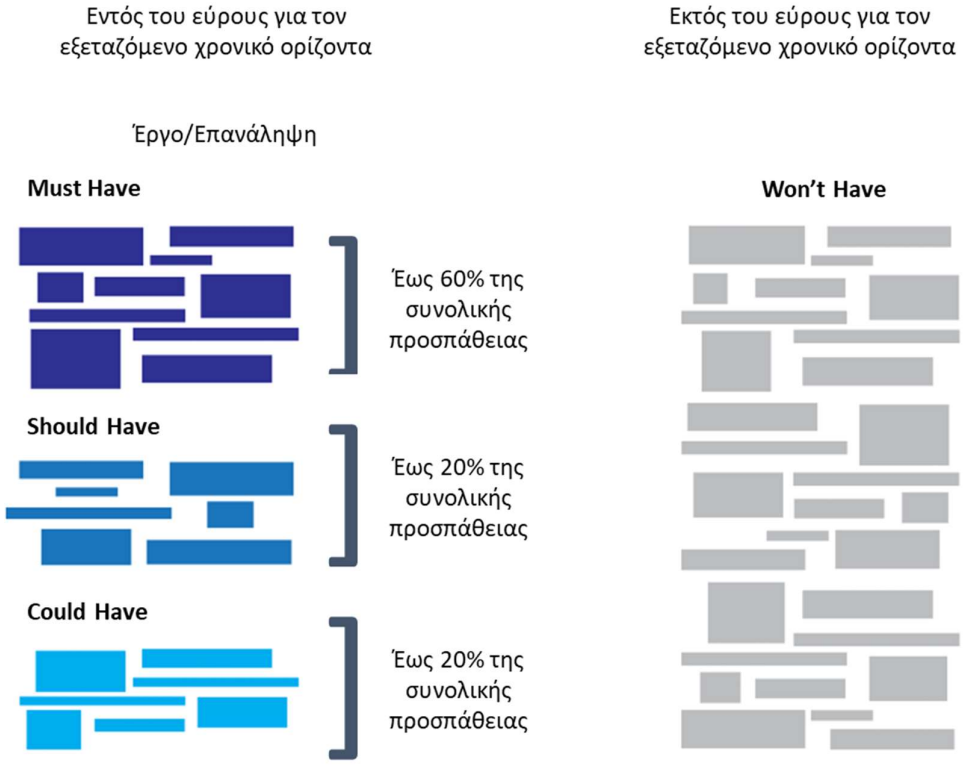
Σε ένα παραδοσιακό έργο, όλες οι απαιτήσεις αντιμετωπίζονται ως Must Have, καθώς υπάρχει η προσδοκία ότι θα παραδοθούν όλες οι απαιτήσεις ακόμη και αν χρειαστεί να καθυστερήσει το έργο.

Στην πραγματικότητα και επειδή οι απαιτήσεις Must Have ορίζουν το ελάχιστα αποδεκτό υποσύνολο απαιτήσεων θα πρέπει να αποτελούν το μεγαλύτερο ποσοστό των απαιτήσεων που περιλαμβάνονται σε κάθε απαίτηση. Συνήθως αυτό το ποσοστό ορίζεται σε 60% της συνολικής προσπάθειας.

Ποσοστά μεγαλύτερα του 60%, για τις απαιτήσεις must have, εισάγουν έναν κίνδυνο αποτυχίας στο έργο, εκτός και εάν η το έργο έχει: α) εκτιμήσεις που είναι γνωστό ότι είναι ακριβείς, β) τεχνική προσέγγιση πολύ καλά κατανοητή και γ) περιβάλλον χαμηλού κινδύνου όσον αφορά τη δυνατότητα εξωτερικών παραγόντων να προκαλέσουν αλλαγές ή καθυστερήσεις. Σε ορισμένες περιπτώσεις έργων το ποσοστό της προσπάθειας που σχετίζεται με απαιτήσεις must have θα πρέπει να είναι μικρότερο από 60%, με αντίστοιχη αύξηση των should haves, ώστε η επιχείρηση να έχει τη μεγαλύτερη δυνατή ευελιξία για τη βελτιστοποίηση της αξίας.

Επιπλέον σε κάθε επανάληψη θα πρέπει να υπάρχει ένα σημαντικό ποσοστό απαιτήσεων could have, συνήθως περίπου το 20% της συνολικής προσπάθειας. Στην καλύτερη περίπτωση οι απαιτήσεις αυτές θα παραδοθούν στο τέλος της επανάληψης, αλλά σε περίπτωση προβλημάτων δημιουργούν ένα περιθώριο ασφαλείας, ώστε η ομάδα να μπορέσει να παραδώσει τις απαιτήσεις must have και should have.

Η λογική αυτή παρουσιάζεται εποπτικά στην Εικόνα 7.8. (Messenger, 2014).



Εικόνα 7.8: *Η κατανομή των απαιτήσεων ανά κατηγορία στη μέθοδο MoSCoW*

7.2.3 Τα σημεία ιστορίας (story points)

Τα σημεία ιστορίας (story points) είναι μια μονάδα μέτρησης του συνολικού μεγέθους μιας ιστορίας χρήστη. Όταν υπολογίζουμε με σημεία ιστορίας, εκχωρούμε ένα αριθμό σημείων για κάθε ιστορία χρήστη. Η τιμή που εκχωρούμε δεν έχει ιδιαίτερη σημασία αφού το σημαντικό είναι η σχετική αξία που προσδίδουμε στις ιστορίες χρηστών. Μια ιστορία στην οποία ανατίθεται ένα δύο θα πρέπει να είναι διπλάσια από μια ιστορία στην οποία ανατίθεται ένα. Θα πρέπει επίσης να είναι τα δύο τρίτα μιας ιστορίας που υπολογίζεται σε τρία σημεία ιστορίας.

Ο αριθμός των σημείων ιστορίας που σχετίζονται με μια ιστορία χρήστη αντιπροσωπεύει το συνολικό μέγεθος της ιστορίας χρήστη. Δεν υπάρχει καθορισμένος τύπος για τον καθορισμό του μεγέθους μιας ιστορίας χρήστη. Μια εκτίμηση σημείων ιστορίας είναι το αποτέλεσμα του συνδυασμού:

- της προσπάθειας που απαιτείται για την ανάπτυξη του χαρακτηριστικού,
- της πολυπλοκότητας ανάπτυξής του,
- του εγγενούς κινδύνου σε αυτό,
- κ.λπ.

Δύο είναι οι πιο συνηθισμένοι τρόποι για να γίνει ο υπολογισμός της προσπάθειας. Ο πρώτος τρόπος είναι να επιλέξουμε μια από τις μικρότερες ιστορίες χρήστη που υπάρχουν και να θεωρήσουμε ότι το μέγεθος της ιστορίας χρήστη είναι 1 σημείο ιστορίας. Η δεύτερη προσέγγιση είναι να επιλέξουμε μια ιστορία που είναι μεσαίου μεγέθους και να της δώσουμε έναν αριθμό κάπου στη μέση του εύρους της κλίμακας που θα χρησιμοποιηθεί Εάν η κλίμακα είναι από το 1-10 να επιλέξουμε 5 σημεία ιστορίας. Μόλις αντιστοιχίσουμε «αυθαίρετα» μια τιμή σημείου ιστορίας στην πρώτη ιστορία χρήστη, κάθε πρόσθετη ιστορία χρήστη εκτιμάται συγκρίνοντάς την με την πρώτη ιστορία ή με οποιαδήποτε άλλη που έχει ήδη εκτιμηθεί.

7.2.4 Οι ιδανικές ημέρες (ideal days)

Οι ιδανικές ημέρες είναι μια εκτίμηση του αριθμού των ημερών που θα χρειαζόταν μια ομάδα για να ολοκληρώσει ένα έργο εάν δεν δούλευε σε τίποτα άλλο και ήταν απερίσπαστη. Σε αυτή τη λογική, «διακοπή εργασιών» σημαίνει οτιδήποτε αναγκάζει την ομάδα να σταματήσει να εργάζεται στο έργο, συμπεριλαμβανομένων των συναντήσεων της ομάδας, της εκπαίδευσης, εργασίας σε άλλα έργα, ασθένειες, διακοπές για αναψυχή, κ.λπ.

Οι παράγοντες που επηρεάζουν μια ιδανική ημέρα είναι οι ακόλουθοι:

- Τεχνική υποστήριξη της τρέχουσας έκδοσης λογισμικού.
- Διόρθωση σφαλμάτων στις τρέχουσες εκδόσεις.
- Συμμετοχή μελών της ομάδας έργου σε εκπαίδευση.
- Ημέρες ασθενείας.
- Χρήση του Email.
- Χρήση των κοινωνικών δικτύων και του διαδικτύου.
- Συναντήσεις εκτός των προκαθορισμένων της ευέλικτης τελετουργίας.
- Συνεντεύξεις υποψηφίων.
- Επιδείξεις λογισμικού εκτός των προκαθορισμένων.
- Προσωπικά θέματα.
- Εναλλαγή εργασιών (multitasking).
- Τηλεφωνικές κλήσεις
- Αξιολογήσεις/ερωτηματολόγια

Σύμφωνα με τη μελέτη των (Meyer et al. 2019) με τίτλο «Today was a good day: The daily life of software developers» που συμπεριέλαβε 5.971 απαντήσεις επαγγελματιών προγραμματιστών στη Microsoft, οι οποίοι παρουσίασαν, το πώς κατανάλωναν τον χρόνο τους σε διάφορες δραστηριότητες στην εργασία τους, κατέδειξε ότι είχαν περιορισμένο έλεγχο στη διάθεση του χρόνου τους. Πιο συγκεκριμένα τα στοιχεία που προέκυψαν παρουσιάζονται στον ακόλουθο Πίνακα. Οι δραστηριότητες με γκρι χρώμα δεν θα πρέπει να περιλαμβάνονται σε μια ιδανική ημέρα.

Δραστηριότητα	% συνολικού χρόνου	Χρόνος σε λεπτά
Κωδικοποίηση	15	84
Διόρθωση σφαλμάτων	14	74
Έλεγχος	8	41
Προδιαγραφές/απαιτήσεις	4	20
Αξιολόγηση κώδικα (review)	5	25
Τεκμηρίωση	2	9
Συναντήσεις	15	85
Email	10	53
Διακοπές	4	24
Βοήθεια προς άλλους συναδέλφους	5	26
Δικτύωση με άλλους συνεργάτες	2	10
Μελέτη – μάθηση	3	17
Διαχειριστικές δραστηριότητες	2	12
Διαλλείματα	8	44
Διάφορα (ταξίδια, υποδομή, κ.λπ.)	3	21
Σύνολο	**100 %**	**545 λεπτά**
Σύνολο δραστηριοτήτων που ανήκουν σε ιδανικές ημέρες	**34%**	**179 λεπτά**
Σύνολο δραστηριοτήτων που δεν ανήκουν σε ιδανικές ημέρες	**66%**	**366 λεπτά**

Πίνακας 7.3: *Δραστηριότητες και χρόνος προγραμματιστή*

Για παράδειγμα, μια ομάδα τριών προγραμματιστών χρειάζεται να εργαστεί συνολικά 45 ιδανικές ημέρες για να ολοκληρώσει ένα έργο. Εάν δεν είχαν «διακοπές» και δεν δούλευαν τίποτα άλλο, αυτό θα αντιστοιχούσε ένα χρονοδιάγραμμα 15 ημερών σύμφωνα με τα στατιστικά στοιχεία του Πίνακα 7.1.

Δυστυχώς, δεν υπάρχει ομάδα που να δουλεύει απερίσπαστη χωρίς τουλάχιστον κάποιες διακοπές. Για αυτόν τον λόγο, για το προηγούμενο σενάριο θα ήταν πιο ρεαλιστικό να υπολογίσουμε ένα χρονοδιάγραμμα 20 ημερών.

Το βασικό πλεονέκτημα των ιδανικών ημερών είναι ότι είναι εύκολο να παραχθούν και είναι κατανοητές σε όλους. Αντίθετα, είναι επιστημονικά αποδεδειγμένο ότι οι εργαζόμενοι έχουν την τάση να χρησιμοποιούν όλο το διαθέσιμο χρόνο και όχι τον αναγκαίο, όπως επίσης ο ορισμός της «ιδανικής ημέρας» δεν είναι η ίδια για όλους.

7.2.5 Το πόκερ σχεδιασμού

Μια από τις πιο γνωστές μεθόδους εκτίμησης προσπάθειας στη μέθοδο Scrum είναι η επονομαζόμενη πόκερ σχεδιασμού (planning poker). Η μέθοδος παρουσιάστηκε για πρώτη φορά και ονομάστηκε έτσι από τον James Grenning το 2002 (https://wingman-sw.com/articles/planning-poker), ενώ έγινε δημοφιλής από τον Mike Cohn όταν παρουσιάστηκε στο βιβλίο του Agile Estimation and Planning (Cohn, M. 2005).

Το πόκερ προγραμματισμού ανήκει σε ένα σύνολο τεχνικών ομαδικής εκτίμησης που βασίζονται στην αρχή ότι οι ομάδες μπορούν να προβλέπουν με μεγαλύτερη ακρίβεια σε σχέση με μεμονωμένα άτομα. Η έννοια αυτή έγινε ευρύτερα γνωστή με τον όρο «η σοφία του πλήθους» (wisdom of the crowds) από τον James Surowiecki στο ομώνυμο βιβλίο του το 2004. Το βιβλίο εξετάζει πώς μεγάλες ομάδες ατόμων έχουν λάβει καλύτερες αποφάσεις σε μια πληθώρα προβλημάτων.

Η ιδέα της σοφίας του πλήθους μπορεί να αναχθεί στη θεωρία της συλλογικής κρίσης του Αριστοτέλη, όπως παρουσιάζεται στο έργο του «Πολιτικά». Σύμφωνα με τον Surowiecki, τα σοφά πλήθη έχουν έναν αριθμό βασικών χαρακτηριστικών:

- Το πλήθος θα πρέπει να μπορεί να έχει μια ποικιλία απόψεων.

- Η γνώμη ενός ατόμου πρέπει να παραμένει ανεξάρτητη από τους γύρω του (και δεν πρέπει να επηρεάζεται από κανέναν άλλο).
- Οποιοσδήποτε συμμετέχει στο πλήθος θα πρέπει να μπορεί να έχει τη δική του γνώμη με βάση τις ατομικές του γνώσεις.
- Το πλήθος θα πρέπει να έχει μηχανισμούς που να συγκεντρώνουν και να συνθέτουν τις ατομικές απόψεις σε μία συλλογική απόφαση.

Το planning poker είναι μια παραλλαγή της μεθόδου Wideband Delphi (McConnell, 2006). Η τεχνική Delphi έχει τρία βασικά χαρακτηριστικά: (1) ανώνυμη απόκριση των συμμετεχόντων, (2) επανάληψη και ελεγχόμενη ανάδραση αυτών και (3) απόκριση της ομάδας με χρήση στατιστικών μεθόδων. Οι εκτιμήσεις λαμβάνονται με ανώνυμη αλληλεπίδραση ειδικών μέσω πολλών επαναλήψεων, υπό την επίβλεψη ενός συντονιστή έως ότου ληφθεί ένα κατάλληλο σύνολο μεμονωμένων απόψεων. Οι Rowe και Wright (1999) παρείχαν μια συστηματική ανασκόπηση εμπειρικών μελετών που εξετάζουν την αποτελεσματικότητα της τεχνικής Delphi και μια κριτική αυτής της έρευνας. Τα ευρήματά τους υποδηλώνουν ότι οι ομάδες Delphi υπερτερούν των στατιστικών ομάδων και των τυπικών ομάδων αλληλεπίδρασης, αν και η τεχνική δεν έχει δείξει σαφή πλεονεκτήματα σε σχέση με άλλες δομημένες διαδικασίες.

Ο Boehm (1981) πρότεινε μια τροποποίηση της τεχνικής των Δελφών που ονομάζεται Wideband Delphi. Το Wideband Delphi περιλαμβάνει περισσότερη ομαδική αλληλεπίδραση και μειώνει την πολυπλοκότητα της αρχικής μεθόδου. Οι ειδικοί παρέχουν τις εκτιμήσεις τους ανώνυμα, αλλά συναντώνται για να τις συζητήσουν (και ενδεχομένως να τις αναθεωρήσουν) στο πλαίσιο άλλων εκτιμήσεων. Ο κύκλος επαναλαμβάνεται έως ότου η ομάδα αποφασίσει ότι ο μέσος όρος είναι αντιπροσωπευτικός και ικανοποιητικός.

Το planning poker είναι μια ομαδική τεχνική εκτίμησης που βασίζεται στη συναίνεση. Η εκτίμηση γίνεται σε επίπεδο user story, όπου ο ιδιοκτήτης του προϊόντος ή ο πελάτης διαβάζει ένα user story στους εκτιμητές, συνήθως μέλη της ομάδας έργου. Κάθε εκτιμητής κρατά μια τράπουλα planning poker. Οι κάρτες στην τράπουλα έχουν αριθμούς πάνω τους. Μια τυπική τράπουλα έχει κάρτες που δείχνουν την ακολουθία Fibonacci που περιλαμβάνει ένα μηδέν: 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89. Άλλες τράπουλες χρησιμοποιούν παρόμοιες προόδους με μια σταθερή αναλογία μεταξύ κάθε τιμής όπως

- 1, 2, 4, 8, ή
- 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100
- XS, S, M, L, XL, XXL (t-short sizes)
- Κ.λπ.

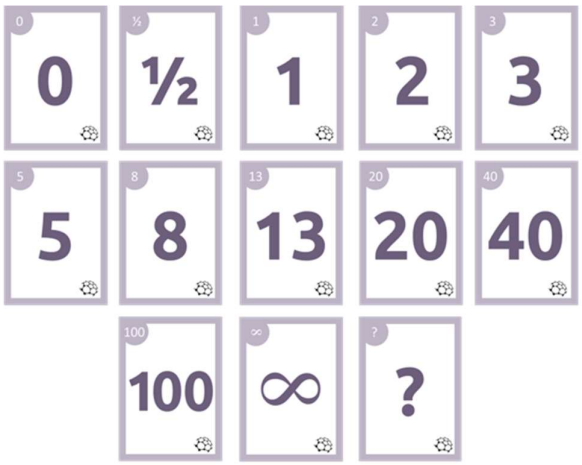
Ο λόγος για τη χρήση της ακολουθίας Fibonacci ή παρόμοιων ακολουθιών αντί του απλού διπλασιασμού κάθε επόμενης τιμής είναι επειδή η εκτίμηση μιας εργασίας ως διπλάσιας δεν είναι συνήθως ακριβής. Μια εργασία που είναι περίπου διπλάσια προσπάθεια από ένα μέγεθος 5, πρέπει να αξιολογηθεί ως λίγο λιγότερο από το διπλάσιο (8) ή λίγο περισσότερο από το διπλάσιο (13).

Στην Εικόνα 7.9 παρουσιάζονται τα φύλλα μιας τέτοιας τράπουλας planning poker. Η κάρτα με το σύμβολο ∞ συμβολίζει ότι το user story είναι πολύ γενικό και δεν μπορεί να γίνει εκτίμηση, ενώ το ? ότι υπάρχουν ακόμα σημεία που χρειάζονται συζήτηση και ανάλυση.

Τα νούμερα στις κάρτες μπορεί να αντιπροσωπεύουν σημεία ιστοριών - story points που όπως είπαμε παραπάνω αποτελούν μια σχετική μονάδα μέτρησης για την εκτίμηση της συνολικής προσπάθειας που θα απαιτηθεί για την πλήρη υλοποίηση μιας ιστορίας χρήστη. Χαρακτηρίσαμε τα story points ως μια σχετική μονάδα μέτρησης, διότι αφενός δεν αντιπροσωπεύουν ένα φυσικό μέγεθος (π.χ. ώρες) και αφετέρου διότι ένα user story που έχει 2 story points χρειάζεται διπλάσια προσπάθεια για να υλοποιηθεί, σε σχέση με ένα user story που έχει 1 story point.

Η διαδικασία που ακολουθούμε για να παίξουμε το planning poker είναι η ακόλουθη:

1. Αρχικά ο ιδιοκτήτης προϊόντος εξηγεί την ιστορία χρήστη, όσο το δυνατό πιο αναλυτικά.
2. Οι εκτιμητές συζητούν για την κάθε ιστορία χρήστη θέτοντας ερωτήσεις στον ιδιοκτήτη προϊόντος. Οι ερωτήσεις μπορεί να είναι:
    a. Σχετικές με το σχεδιασμό - Πρέπει να έχουμε νέες πληροφορίες πριν ξεκινήσουμε τη σχεδίαση σε HTML, python κ.λπ.;
    b. Σχετικές με τον προγραμματισμό του συστήματος - Έχουμε έτοιμη κάποια βιβλιοθήκη κλάσεων ή πρέπει να γραφεί από την αρχή;
    c. Σχετικές με τον έλεγχο- Απαιτείται κάποια συγκεκριμένη ρύθμιση για το μοναδιαίο έλεγχο;
3. Όταν κάθε ιστορία χρήστη έχει συζητηθεί ενδελεχώς, ο κάθε εκτιμητής επιλέγει ένα φύλλο της τράπουλας για να παρουσιάσει την εκτίμησή του, ατομικά στην αρχή.
4. Αφού όλοι οι εκτιμητές έχουν κάνει την εκτίμησή τους, ή μετά την πάροδο συγκεκριμένου χρονικού διαστήματος, όλες οι κάρτες αποκαλύπτονται ταυτόχρονα.
5. Εάν όλοι οι εκτιμητές έχουν επιλέξει την ίδια τιμή, τότε αυτή η τιμή αποτελεί την εκτίμηση για την ιστορία χρήστη.
6. Εάν όχι, οι εκτιμητές συζητούν τις εκτιμήσεις τους αναφέροντας τους λόγους για τους οποίους έκαναν αυτή την επιλογή, ιδιαίτερα αυτοί των οποίων η εκτίμηση ήταν ιδιαίτερα χαμηλή ή υψηλή. Μετά από αυτή τη συζήτηση ο κάθε εκτιμητής επιλέγει ξανά μια κάρτα εκτίμησης και όλες οι κάρτες αποκαλύπτονται και πάλι την ίδια στιγμή.
7. Η διαδικασία του planning poker επαναλαμβάνεται έως ότου επιτευχθεί συναίνεση.



Εικόνα 7.9: *Η τράπουλα με τα φύλλα εκτίμησης της προσπάθειας στο planning poker.*

Η εφαρμογή του πόκερ σχεδιασμού έχει ορισμένα βασικά προβλήματα τα οποία μπορούν να συνοψισθούν ως ακολούθως:

- Εμφάνιση των καρτών πολύ νωρίς: Είναι σύνηθες ότι ορισμένα μέλη της ομάδας δείχνουν την κάρτα πριν από τα υπόλοιπα. Αυτό μπορεί να επηρεάσει την απόφαση των υπολοίπων να επιλέξουν την τιμή. Οι κάρτες πρέπει να εμφανίζονται όλες ταυτόχρονα πάντα. Αυτό σήμερα δεν είναι τόσο συχνό καθώς το πόκερ σχεδιασμού παίζεται με χρήση εφαρμογών.
- Εκτίμηση με βάση το μέσο όρο ή με δημοκρατικές διαδικασίες: Ορισμένα μέλη της ομάδας κουράζονται να επαναλαμβάνουν τη διαδικασία ή πιστεύουν ότι πρέπει να συμφωνήσουν σύντομα με μια εκτίμηση, αυτή που πιστεύει η πλειοψηφία ή απλώς χρησιμοποιούν τον μέσο όρο

των εκτιμήσεων. Αυτό όμως δεν είναι σωστό αφού το ζητούμενο δεν είναι οι δημοκρατικές διαδικασίες αλλά θέλουμε να υπάρχει συναίνεση όλων γιατί όλα τα μέλη της ομάδας θα δεσμευτούν σε αυτή την εκτίμηση και το μέγεθος της ιστορίας.

- Βιασύνη στη παραγωγή εκτιμήσεων: Σχεδόν πάντα, μερικά μέλη της ομάδας έργου βιάζονται να παράγουν μια εκτίμηση. Ο βασικός σκοπός είναι να κατανοήσουμε τα ζητούμενα και πιθανόν να βελτιώσουμε τις ιστορίες χρηστών καθώς και τα κριτήρια αποδοχής.
- Μετατροπή των εκτιμήσεων σε ώρες: Η εκτίμηση είναι πάντα σχετική, εάν η πρώτη ιστορία που εκτιμήθηκε είναι μεγέθους 5, τα μέλη της ομάδας θα συγκρίνουν την πολυπλοκότητα και το μέγεθος των επόμενων ιστοριών χρηστών με αυτήν, ώστε να καθορίσουν, εάν η νέα ιστορία είναι μεγέθους 1, 2, 3, 5, 8 ή άλλου μεγέθους.
- Εξαίρεση κάποιων μελών της ομάδας έργου από το πόκερ σχεδιασμού. Πολλές φορές μέλη της ομάδας έργου που είναι επιφορτισμένα με τον ρόλο διασφάλισης ποιότητας ή/και μη προγραμματιστικό εξαιρούνται από τη διαδικασία εκτίμησης. Αυτό είναι λάθος, καθώς όλα τα μέλη της ομάδας πρέπει να συμμετέχουν στο πόκερ σχεδιασμού. Για παράδειγμα: Εάν κάποιος είναι ειδικός στις διαδικασίες διασφάλισης ποιότητας, ή στη διαχείριση της βάσης δεδομένων ή στη δημιουργία της υποδομής, μπορεί να έχει μια άλλη άποψη σχετικά με την πολυπλοκότητα των ιστοριών χρηστών και υποθέσεων ανάπτυξης.

7.3 Οι ευέλικτες μετρικές

Μέτρηση είναι η διαδικασία με την οποία αριθμοί ή σύμβολα αντιστοιχούνται σε ιδιότητες οντοτήτων του πραγματικού κόσμου, έτσι ώστε να τις περιγράφουν σύμφωνα με καθορισμένους κανόνες (Φιτσιλής et al., 2008). Η διασφάλιση ποιότητας είναι συνυφασμένη με την έννοια των μετρήσεων (measurements). Ο DeMarco τονίζει πως «*είναι αδύνατο να ελέγξεις ότι δεν μπορείς να μετρήσεις*» (DeMarco, 1982).

Η ευέλικτη προσέγγιση απαιτεί ποιοτικές μετρήσεις που θα βοηθήσουν τα μέλη της ομάδας να βελτιώσουν τις διαδικασίες τους. Όπως έχουμε τονίσει, οι ομάδες στην ευέλικτη προσέγγιση είναι αυτοοργανωμένες και συνεπώς η ύπαρξη ουσιαστικών μετρικών οδηγεί σε πραγματική βελτίωση της απόδοσης. Ορισμένοι γενικοί κανόνες είναι οι ακόλουθοι:

- Οποιαδήποτε ευέλικτη μέτρηση δεν θα πρέπει να επιβάλλεται από τη διοίκηση αλλά να ορίζεται από την ομάδα ανάπτυξης με στόχο τη μάθηση και τη βελτίωση.
- Οι μετρικές θα πρέπει να απαντούν σε συγκεκριμένες πρακτικές ερωτήσεις και προβλήματα.
- Οι μετρικές θα πρέπει να συνδυάζονται μεταξύ τους στην εξαγωγή συμπερασμάτων. Ο συνδυασμός αυτών δίνει μια ισορροπημένη εικόνα του έργου αλλά και του αποτελέσματος που παράγεται.
- Μια μέτρηση πρέπει να είναι ένας ουσιαστικός δείκτης για μια πιθανή αλλαγή. Επίσης θα πρέπει να δείχνει τα προβλήματα εγκαίρως ώστε να μπορούν να γίνουν διορθωτικές κινήσεις.

Η επιλογή των μετρικών πρέπει να γίνεται συνολικά και να πληροί για παράδειγμα τα παρακάτω κριτήρια (Kupiainen et al, 2015):

- Να μετρούν την αξία και την ικανοποίηση των αναγκών του πελάτη.
- Να μετρούν την ποιότητα, δηλαδή το προϊόν να είναι χωρίς προβλήματα και ελαττώματα.
- Να έχουν δυνατότητα πρόβλεψης σχετικά με την ικανότητα προγραμματισμού και παράδοσης.
- Να έχουν δυνατότητα μέτρησης της σταθερότητας, δηλαδή της ικανότητας της ομάδας έργου να διατηρήσει ένα ρυθμό ανάπτυξης επ' αόριστον.
- Να μετρούν την παραγωγικότητα ώστε να μπορούμε να αξιοποιούμε τους πόρους της επιχείρησης με τον καλύτερο δυνατό τρόπο.

7.3.1 Sprint Burndown

Μια από τις πιο αποτελεσματικές και δημοφιλείς μετρήσεις παραγωγικότητας στην ευέλικτη προσέγγιση είναι το διάγραμμα burndown.

Τα μέλη της ομάδας έργου στην αρχή του έργου αλλά και καθημερινά υπολογίζουν τον εκτιμώμενο χρόνο που απαιτείται για την ολοκλήρωση κάθε ιστορίας χρήστη. Μετά την ενημέρωση των επιμέρους εργασιών, προκύπτει ο συνολικός χρόνος που απομένει ώστε να ολοκληρωθεί το sprint (βλέπε Εικόνα 7.10).

Εικόνα 7.10: *Το διάγραμμα burndown*

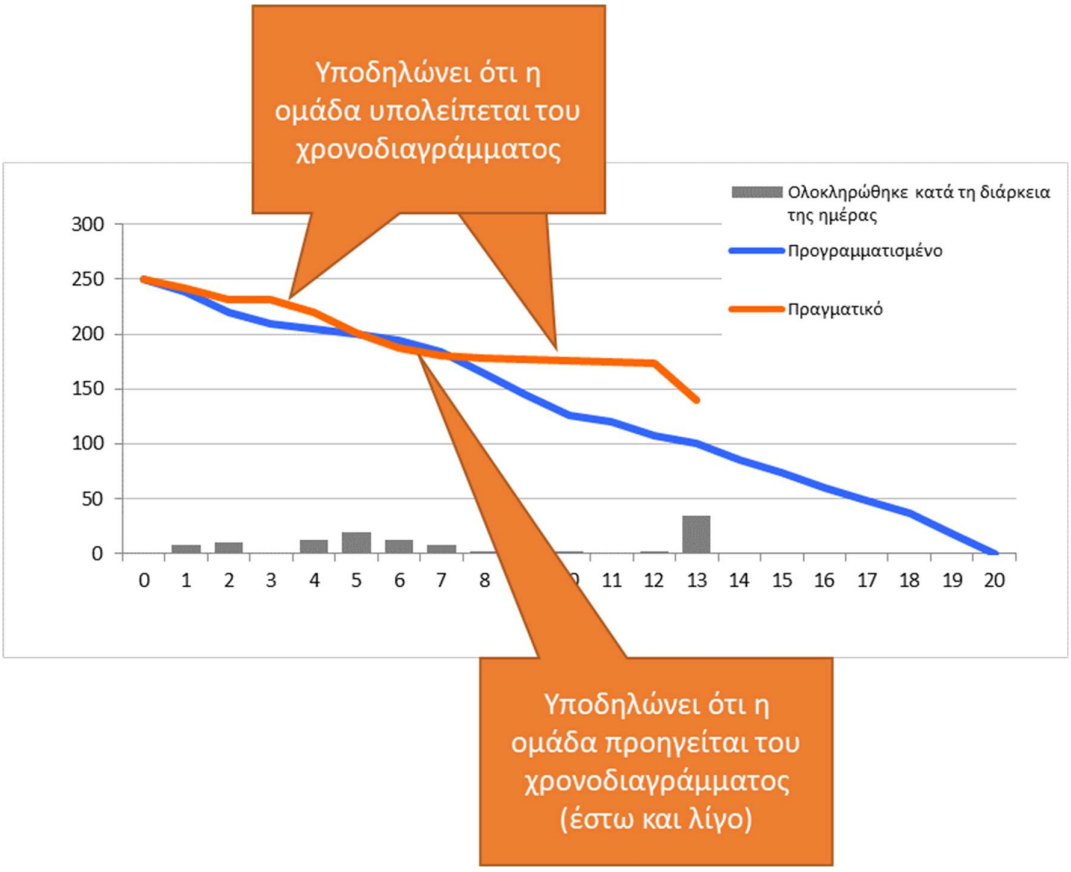
Ο οριζόντιος άξονας αντιπροσωπεύει τον χρόνο που αφιερώνεται για ένα συγκεκριμένο sprint ή για την ολοκλήρωση του έργου.

Ο κάθετος άξονας αντιπροσωπεύει την υπόλοιπη ανεκτέλεστη προσπάθεια που απαιτεί το sprint ή το έργο. Συνεπώς, η επάνω αριστερή γωνία είναι το σημείο όπου ξεκινά το έργο (ή το sprint) και η κάτω δεξιά γωνία είναι το σημείο που τελειώνει.

Οι δύο γραμμές που εμφανίζονται στο διάγραμμα αντιπροσωπεύουν:

- Την ιδανική γραμμή εργασίας (πορτοκαλί γραμμή - διακεκομμένη). Η γραμμή αυτή αντιπροσωπεύει τον τρόπο με τον οποίο η ομάδα θα υλοποιήσει, με τον ιδανικό τρόπο, όλη την εναπομένουσα δουλειά αν όλα συμβούν σύμφωνα με το πλάνο. Είναι η ιδανική εκτίμηση που λειτουργεί ως βάση για όλους τους υπολογισμούς του έργου.
- Την γραμμή που αναπαριστά την πραγματικά υπολειπόμενη εργασία (μπλε γραμμή– συνεχής). Η γραμμή αυτή αναπαριστά την πραγματική εργασία που έχει γίνει καθώς και πόση ποσότητα εργασίας έχει απομείνει. Συνήθως δεν είναι ευθεία γραμμή, καθώς επηρεάζεται από τα γεγονότα της πραγματικής ζωής και τις καθυστερήσεις της ομάδας. Στην ιδανική περίπτωση, θέλουμε η γραμμή που αναπαριστά την πραγματικά υπολειπόμενη εργασία να παραμένει κάτω από την ιδανική γραμμή εργασίας.

Η δημιουργία του διαγράμματος burndown είναι απλή, η πληροφορία που δίνει είναι συνοπτική και επιτρέπει εύκολα να κατανοήσουμε την πρόοδο του έργου ανάλογα με τη σχετική θέση των δύο γραμμών (βλέπε Εικόνα 7.11).



Εικόνα 7.11: *Ανάλυση του διαγράμματος burndown*

Αντίστοιχα με το διάγραμμα burndown το οποίο παράγεται για ένα Sprint υπάρχουν:

- Διάγραμμα burndown για epic
- Διάγραμμα burndown για αποδέσμευση (release)

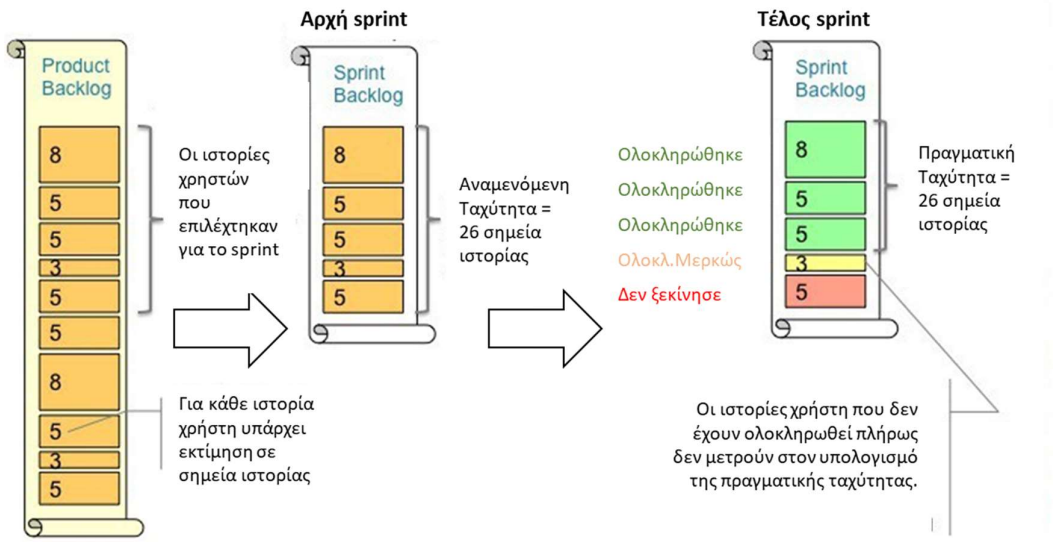
7.3.2 Η ταχύτητα (velocity)

Η μετρική της ταχύτητας (velocity) στην ευέλικτη προσέγγιση είναι ένας απλός υπολογισμός των μονάδων εργασίας που ολοκληρώθηκαν σε ένα συγκεκριμένο χρονικό διάστημα. Οι μονάδες εργασίας μπορούν να μετρηθούν με διάφορους τρόπους, συμπεριλαμβανομένων των ανθρωποωρών, των σημείων ιστορίας, των ιδανικών ημερών, κ.λπ. Το ίδιο ισχύει και για το χρονικό διάστημα για το οποίο γίνεται η μέτρηση που μπορεί να είναι εβδομάδες, sprint, epic, ή αποδεσμεύσεις.

Η ταχύτητα στη προσέγγιση Scrum έχει δύο παραλλαγές, αλλά οι υπολογισμοί και για τις δύο παραλλαγές είναι παρόμοιοι. Η πραγματική ταχύτητα υπολογίζεται διαιρώντας τα συνολικά σημεία ιστορίας (story points) που υλοποίησε η ομάδα με τον αριθμό των sprint. Για παράδειγμα, εάν η ομάδα έργου έχει υλοποιήσει συνολικά 80 σημεία ιστορίας σε 4 sprint, τότε η πραγματική ταχύτητα της ομάδας θα είναι 20 σημεία ιστορίας ανά sprint. Αυτή η έκδοση είναι πιο αποδεκτή και περισσότερο γνωστή από τις ομάδες Scrum και χρησιμοποιείται συνήθως κατά τον υπολογισμό της ταχύτητας.

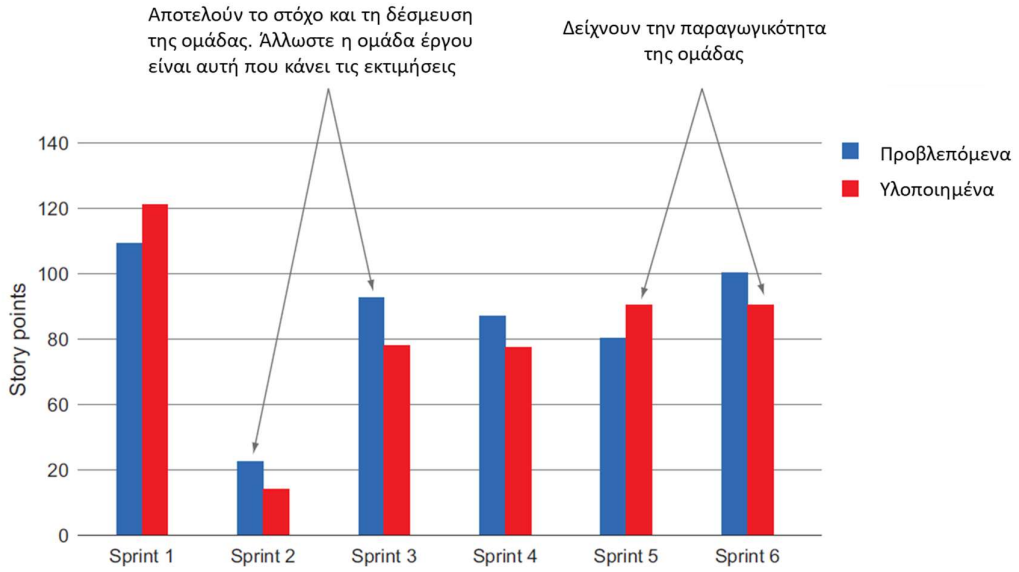
Η δεύτερη παραλλαγή υπολογισμού της ταχύτητας είναι γνωστή ως η αναμενόμενη ταχύτητα (expected velocity), όπου τα συνολικά εκτιμώμενα σημεία ιστορίας διαιρούνται με τον αριθμό των sprints. Εδώ, η ομάδα εκτιμά τα σημεία ιστορίας που μπορεί να ολοκληρώσει σε ένα δεδομένο αριθμό sprint. Για παράδειγμα, εάν η ομάδα εκτιμήσει ότι μπορεί να ολοκληρώσει συνολικά 150 σημεία ιστορίας σε δύο sprint τότε, η αναμενόμενη ταχύτητά της είναι 75 πόντους ανά sprint. Αυτή η παραλλαγή μας επιτρέπει να

συγκρίνουμε την πραγματική ταχύτητα με την αναμενόμενη ταχύτητα, έτσι ώστε δούμε κατά πόσο η απόδοση της ομάδας είναι η αναμενόμενη και εάν η ομάδα ανταποκρίνεται στις δεσμεύσεις της.



Εικόνα 7.12: *Υπολογισμός της αναμενόμενης και πραγματικής ταχύτητας.*

Ένα διάγραμμα που είναι χρήσιμο για την ομάδα έργου και τον ιδιοκτήτη προϊόντος είναι το διάγραμμα που δείχνει συνολικά για όλα τα sprint την αναμενόμενη ταχύτητα σε σχέση με την πραγματική (βλέπε Εικόνα 7.13).



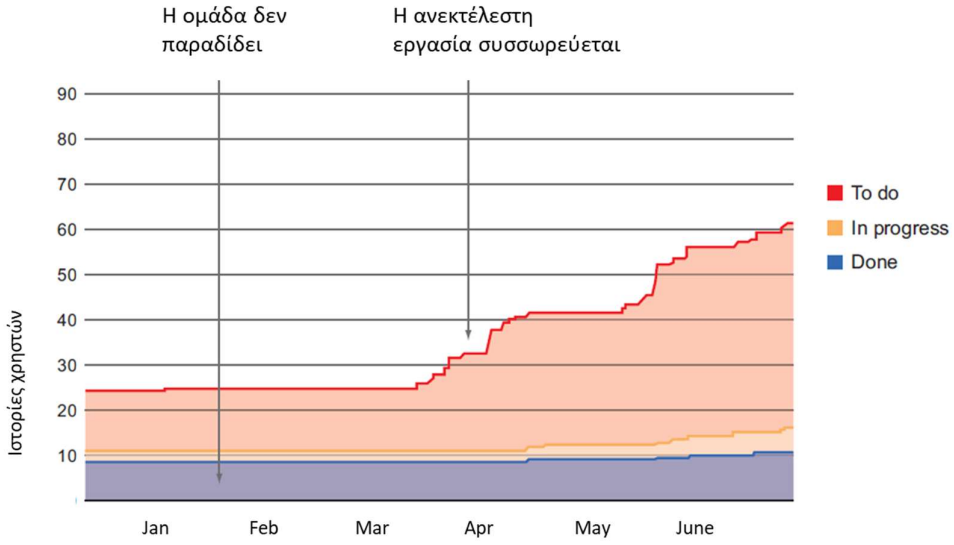
Εικόνα 7.13: *Αναμενόμενη και πραγματική ταχύτητα ανά sprint*

7.3.3 Η αθροιστική ροή εργασιών (cumulative flow)

Η αθροιστική ροή εργασιών παρουσιάζει σε βάθος χρόνου πόση εργασία έχει γίνει, πόση εργασία είναι σε εξέλιξη καθώς και πόση εργασία απομένει. Το διάγραμμα αθροιστικής ροής χρησιμοποιείται συνήθως για τον εντοπισμό σημείων συμφόρησης στη διαδικασία αναπαριστάνοντας οπτικά πότε ένας τύπος εργασίας αυξάνεται ταχύτερα από τους υπόλοιπους. Στην Εικόνα 7.14 παρουσιάζεται ένα παράδειγμα αθροιστικού

διαγράμματος ροής όπου η ομάδα δεν ολοκληρώνει σχεδόν καμία από τις ιστορίες χρηστών και η λίστα των ανεκτέλεστων ιστοριών χρηστών αυξάνεται δυσανάλογα.

Όπως και με τα άλλα διαγράμματα, το διάγραμμα αθροιστικής ροής εργασιών δείχνει μόνο ένα τμήμα της συνολικής εικόνας. Είναι πιθανό ότι οι ιστορίες χρηστών να μην έχουν διασπαστεί σωστά και να είναι απλώς πολύ μεγάλες (Davis, 2015).



Εικόνα 7.14: *Το διάγραμμα αθροιστικής ροής εργασιών*

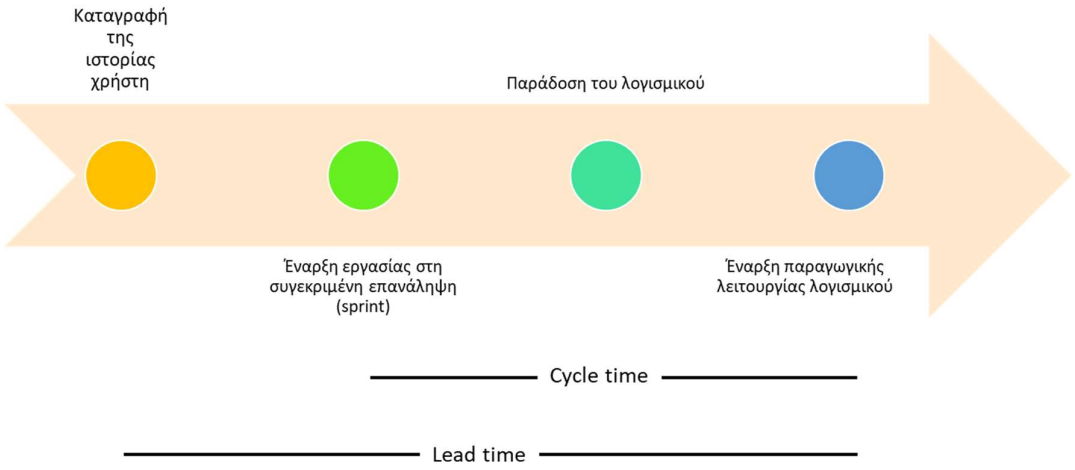
7.3.4 Χρόνος παράδοσης (lead time) και χρόνος υλοποίησης (cycle time)

Ο χρόνος παράδοσης είναι ένας όρος που έχουμε δανειστεί από τα παραγωγικά συστήματα και από τη λιτή προσέγγιση διοίκησης (lean management) και ορίζεται ως ο χρόνος που μεσολαβεί μεταξύ της παραγγελίας ενός πελάτη και της παραλαβής του προϊόντος που παρήγγειλε.

Μεταφέροντας την παραπάνω λογική στη μηχανική λογισμικού, ο χρόνος παράδοσης μπορεί να περιγραφεί πιο αφηρημένα ως ο χρόνος που μεσολάβησε μεταξύ του προσδιορισμού μιας απαίτησης/μιας ιστορίας χρήσης και της χρήσης της «στην παραγωγή», δηλαδή από πραγματικούς χρήστες υπό κανονικές συνθήκες.

Αντίστοιχα ο χρόνος υλοποίησης (ο όρος δίνεται σε ελεύθερη μετάφραση) ή χρόνος κύκλου (Cycle time) είναι ο χρόνος μεταξύ της έναρξης υλοποίησης της ιστορίας χρήσης και της χρήσης της «στην παραγωγή», δηλαδή από πραγματικούς χρήστες υπό κανονικές συνθήκες.

Η γραφική αναπαράσταση των παραπάνω χρόνων επιτρέπει την καλύτερη κατανόησή τους (βλέπε Εικόνα 7.15).



Εικόνα 7.15: *Lead time και cycle time*

Οι ομάδες που επιλέγουν την προσέγγιση kanban επιλέγουν τις μετρικές αυτές σε σχέση με την πιο γνωστή μετρική της ταχύτητας. Συνεπώς, αντί να στοχεύουν στην αύξηση της ταχύτητας, οι πρωτοβουλίες βελτίωσης της ομάδας έργου έχουν ως στόχο να μειώσουν τον χρόνο παράδοσης.

Και οι δύο μετρικές είναι πολύ καλοί δείκτες της ικανότητας της ομάδας να προσφέρει αξία στους πελάτες της. Προφανώς, μια εφάπαξ μέτρηση έχει περιορισμένη αξία καθότι δεν μπορεί να περιγράψει τη λειτουργία της ομάδας στη καθημερινή της δουλειά. Συνεπώς είναι αναγκαίο να κάνουμε συνεχείς μετρήσεις διότι μόνο τότε μπορεί να δει κανείς εάν μια ομάδα βελτιώνεται. Ο απώτερος στόχος είναι να μειωθεί ο χρόνος παράδοσης, καθώς αυτό είναι κάτι που εκτιμούν γενικότερα όλοι οι πελάτες.

Ο χρόνος παράδοσης είναι μια πολύ σημαντική μετρική που συνδέεται και με τη συντήρηση των συστημάτων λογισμικού (maintenability). Στην περίπτωση αυτή το αίτημα του πελάτη δεν αφορά μια ιστορία χρήστη αλλά είναι μια αναφορά προβλήματος και συνδέεται άμεσα με μετρικές όπως MTTR – Mean Time to Repair (μέσος χρόνος επισκευής), τη διαθεσιμότητα του συστήματος, κ.λπ. (Φιτσιλής, 2018).

7.3.5 Αριθμός ελαττωμάτων (Bug Count)

Στον παραδοσιακό τρόπο ανάπτυξης λογισμικού με το μοντέλο καταρράκτη, η διασφάλιση της ποιότητας, το QA (Quality Assurance), και ο έλεγχος λογισμικού είναι ξεχωριστές δραστηριότητες που πολλές φορές εκτελούνταν από ξεχωριστή ομάδα. Το μοντέλο καταρράκτη δεν εμπεριέχει επαναλήψεις και κάθε στάδιο πρέπει να ολοκληρωθεί πριν ξεκινήσει το επόμενο στάδιο. Σε αυτό το μοντέλο, μια ανεξάρτητη ομάδα διασφάλισης ποιότητας ορίζει τις δοκιμές (test cases) που καθορίζουν εάν το λογισμικό πληροί τις αρχικές απαιτήσεις.

Στην ευέλικτη προσέγγιση ο έλεγχος λογισμικού και οι δοκιμές είναι μια καθημερινή εργασία σε κάθε sprint. Θα θυμίσουμε ότι στον ορισμό κάθε ιστορίας χρήστη περιλαμβάνονται κριτήρια αποδοχής (acceptance criteria).

Τα σφάλματα (bugs) ή ελαττώματα (defects) αντιπροσωπεύουν ασυνέπειες (inconsistencies) ή ασυμβατότητες (non compliance) είτε απλά λάθη στην υλοποίηση του συστήματος λογισμικού. Διάφορες ομάδες έχουν διαφορετικούς ορισμούς για την έννοια του σφάλματος. Η παλαιότερη και πιο δημοφιλής ταξινόμηση σφαλμάτων προτάθηκε από την IBM (Chillarege et al., 1992), η οποία εισήγαγε τη λεγόμενη Ορθογώνια Ταξινόμηση Ατελειών (Orthogonal Defect Classification - ODC). Αυτή η ταξινόμηση περιλαμβάνει 13 κατηγορίες και ταξινομεί τα σφάλματα ως προς τη δομή του προγράμματος.

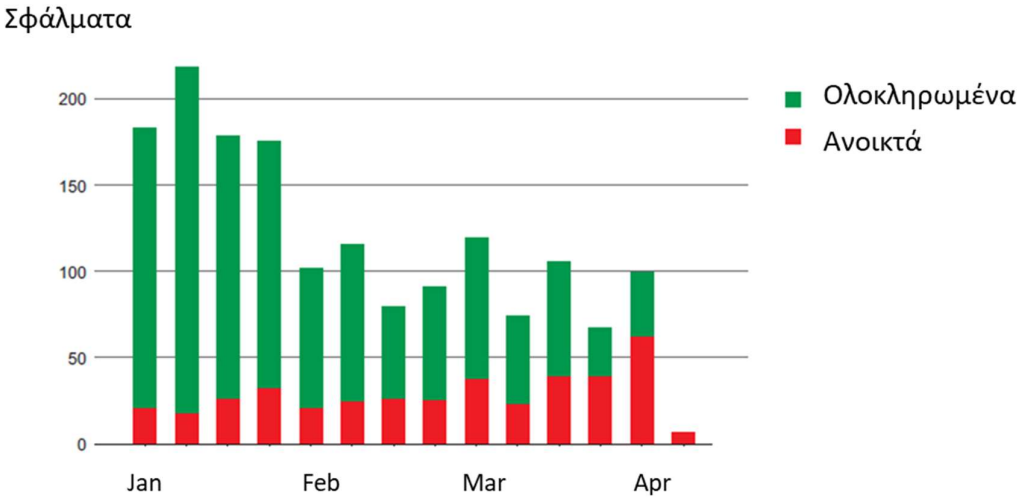
Τα σφάλματα θα εμφανιστούν σε διαφορετικές χρονικές στιγμές, ανάλογα με το πώς εργάζεται η ομάδα έργου. Στα ευέλικτα έργα υπάρχουν πολλαπλοί βρόχοι ανατροφοδότησης και ελέγχου. Στην Εικόνα 7.16, βλέπετε τα διαφορετικά σημεία ελέγχου κατά τη διάρκεια ενός έργου που ακολουθεί την προσέγγιση scrum. Προφανώς, η ομάδα ανάπτυξης μπορεί να ενσωματώσει αμέσως αυτή την ανατροφοδότηση στο προϊόν, αυξάνοντας την ποιότητα του προϊόντος σε καθημερινή βάση.

Εικόνα 7.16: *Σημεία ελέγχου στην ευέλικτη προσέγγιση*

Η Εικόνα 7.17 δείχνει ένα γράφημα που παρακολουθεί σφάλματα με την πάροδο του χρόνου. Επειδή τα σφάλματα αντιπροσωπεύουν μη επιθυμητή λειτουργία του λογισμικού, προφανώς ο μικρός αριθμός σφαλμάτων υποδηλώνει καλή ποιότητα λογισμικού.

Τα σφάλματα μπορούν επίσης να αναλυθούν ανάλογα με τη σοβαρότητά τους. Ένα μη κρίσιμο σφάλμα μπορεί να είναι ότι δεν υπάρχει η σωστή γραμματοσειρά στο υποσέλιδο ενός ιστότοπου.

Ο Πίνακας 7.4 παρουσιάζει μια κατάταξη ανάλογα με τη σοβαρότητα του προβλήματος.



.

Εικόνα 7.17: *Γράφημα σφαλμάτων*

Τύπος	**Περιγραφή**
Τύπου 0	Σοβαρό σφάλμα του συστήματος. Το σύστημα δεν μπορεί να εκτελέσει μία από τις βασικές του λειτουργίες. Αυτού του είδους οι αλλαγές προκύπτουν κατά τη διάρκεια του ελέγχου του συστήματος και υλοποιούνται συνήθως πριν την παράδοση του συστήματος.

Τύπος	**Περιγραφή**
Τύπου 1	Σφάλμα του συστήματος. Πιο συγκεκριμένα, σφάλμα στη λειτουργικότητα του συστήματος, η οποία δεν εμποδίζει τη βασική του λειτουργία ή υπάρχει εναλλακτικός τρόπος λειτουργίας.
Τύπου 2	Βελτίωση του συστήματος. Αλλαγή που προκύπτει από αίτηση για βελτίωση της χρηστικότητας, της απόδοσης κ.λπ. του συστήματος.
Τύπου 3	Αλλαγή απαιτήσεων. Νέες απαιτήσεις χρηστών ή τροποποίηση υπάρχουσας λειτουργικότητας.
Τύπου 4	Άλλου είδους αλλαγές. Πρόκειται για αλλαγές που δεν ανήκουν σε καμία από τις παραπάνω κατηγορίες.

Πίνακας 7.4: *Κατάταξη σφαλμάτων ανάλογα με τη σοβαρότητά τους*

Βιβλιογραφία/Αναφορές

Boehm, B.W., 1981. *Software Engineering Economics*. Prentice-Hall.

Bukhsh, F. A., Bukhsh, Z. A., & Daneva, M. (2020). A systematic literature review on requirement prioritization techniques and their empirical evaluation. *Computer Standards & Interfaces*, *69*, 103389.

Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K., & Wong, M. Y. (1992). Orthogonal defect classification-a concept for in-process measurements. *IEEE Transactions on software Engineering*, *18*(11), 943-956.

Cohn, M. (2004). *User stories applied: For agile software development*. Addison-Wesley Professional.

Cohn, M. Agile Estimation and Planning (2005).

Davis, C. (2015). *Agile Metrics in Action: How to measure and improve team performance*. Simon and Schuster.

DeMarco T. (1982). Controlling Software Projects, Prentice Hall, New-York.

Goldratt, E. M. (1990). *Theory of constraints* (pp. 1-159). Croton-on-Hudson: North River.

IIBA. 2015. *A Guide to the Business Analysis Body of Knowledge*. IIBA, v3.0. Toronto, Ontario, Canada: International Institute of Business Analysis.

Karlsson, J., Wohlin, C., & Regnell, B. (1998). An evaluation of methods for prioritizing software requirements. *Information and software technology*, *39*(14-15), 939-947.

Kravchenko, T., Bogdanova, T., & Shevgunov, T. (2022). Ranking Requirements Using MoSCoW Methodology in Practice. In *Computer Science On-line Conference* (pp. 188-199). Springer, Cham.

Kukhnavets, P. (2022). Agile Metrics. https://hygger.io/guides/agile/project-management/agile-metrics/

Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). Using metrics in Agile and Lean Software Development–A systematic literature review of industrial studies. *Information and software technology*, *62*, 143-163.

Layton, M. C., & Ostermiller, S. J. (2017). Agile project management for dummies, 2nd edition. John Wiley & Sons.

Lawrence, L. & Green, P. (2021). The Humanizing Work Guide to Splitting User Stories. https://www.humanizingwork.com/the-humanizing-work-guide-to-splitting-user-stories

Leffingwell, D. (2010). *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.

Meyer, A. N., Barr, E. T., Bird, C., & Zimmermann, T. (2019). Today was a good day: The daily life of software developers. *IEEE Transactions on Software Engineering*, *47*(5), 863-880.

McConnell, S. (2006). *Software estimation: demystifying the black art*. Microsoft press.

Messenger, S. (2014). *DSDM Agile Project Framework Handbook,* Agile Business Consortium. https://www.agilebusiness.org/page/ProjectFramework_00_welcome

Pohl, K. (2010). *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated.

Saaty, R. W. (1987). The analytic hierarchy process—what it is and how it is used. *Mathematical modelling*, *9*(3-5), 161-176.

Sandeep, R. C. (2020). *Estimation Techniques in Agile Software Development* (Master's thesis).

Surowiecki, J. (2005). *The wisdom of crowds*. Anchor.

Vacanti, D. (2015). Actionable Agile Metrics for Predictability. Leanpub. https://leanpub.com/actionableagilemetrics

Wake, B. (2013). Invest in good stories, and smart tasks. https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

Wiegers, K. (1999). Software Requirements: A Pragmatic Approach.

Wiegers, K. (2005). *More about software requirements: thorny issues and practical advice*. Microsoft Press.

Εξάρχου, Μ. (2014), Τεχνικές ιεράρχησης απαιτήσεων σε έργα πληροφορικής, Πανεπιστήμιο Θεσσαλίας. https://ir.lib.uth.gr/xmlui/bitstream/handle/11615/53090/20852.pdf?sequence=1

Φιτσιλής, Π. (2018). Σύγχρονα πληροφοριακά συστήματα επιχειρήσεων (2η έκδοση). Broken Hill Publishers

Φιτσιλής Π., Σταμέλος Ι. & Ξένος Μ. (2009), Προγραμματισμός Έργων Πληροφορικής – Αντικειμενοστρεφείς Μεθοδολογίες, Ελληνικό Ανοικτό Πανεπιστήμιο.

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Ποιες από τις παρακάτω δεν είναι καλές ιστορίες χρήστη και γιατί;

- **Ο χρήστης μπορεί να τρέξει το σύστημα σε Windows και σε Linux.**
- **Όλα τα γραφήματα και τα διαγράμματα θα υλοποιηθούν χρησιμοποιώντας εξωτερικές βιβλιοθήκες.**
- **Ο χρήστης μπορεί να αναιρέσει (undo) έως και πενήντα εντολές.**
- **Το λογισμικό θα κυκλοφορήσει έως τις 30 Ιουνίου.**
- **Το λογισμικό θα είναι γραμμένο σε Java.**
- **Ο χρήστης μπορεί να επιλέξει τη χώρα του από μια αναπτυσσόμενη λίστα.**
- **Ο χρήστης θα κληθεί να αποθηκεύσει την εργασία του εάν δεν την έχει αποθηκεύσει για 15 λεπτά.**
- **Ο χρήστης μπορεί να επιλέξει τη λειτουργία "Αποθήκευση αρχείου σε PDF ".**

Απάντηση/Λύση

Οι παρακάτω ιστορίες χρήστη δεν είναι κατάλληλες

- Όλα τα γραφήματα και τα διαγράμματα θα υλοποιηθούν χρησιμοποιώντας εξωτερικές βιβλιοθήκες.
    - Ο τελικός χρήστης δεν ενδιαφέρεται για το ποια βιβλιοθήκη θα χρησιμοποιηθεί
- Το λογισμικό θα κυκλοφορήσει έως τις 30 Ιουνίου.
    - Δεν είναι καλή ιστορία χρήστη διότι η πληροφορία αυτή αφορά την αποδέσμευση του λογισμικού.
- Το λογισμικό θα είναι γραμμένο σε Java.
    - Εξαρτάται από το είδος του προϊόντος. Αν το λογισμικό αφορά μια επιχειρηματική εφαρμογή δεν αποτελεί μια καλή ιστορία χρήστη. Αν αφορά μια βιβλιοθήκη τότε είναι αποδεκτή.
- Ο χρήστης μπορεί να επιλέξει τη χώρα του από μια αναπτυσσόμενη λίστα.
    - Είναι μια καλή ιστορία χρήστη αλλά πολύ μικρού μεγέθους

Κριτήριο αξιολόγησης 2

Χρησιμοποιώντας το πρότυπο που δόθηκε στον Πίνακα 7.1. περιγράψτε την ιστορία χρήστη για την ακύρωση κράτησης ενός εισιτηρίου μιας παράστασης.

Απάντηση/Λύση

Τίτλος: Ακύρωση κράτησης	**Προτεραιότητα:** Πολύ μεγάλη	**Εκτίμηση προσπάθειας:** 5
Ως (**As a**) Χρήστης		

Θέλω να (**I want to**) μπορώ να ακυρώσω την κράτησή μου οποιαδήποτε στιγμή **Ώστε** (**so that I can**) να μην χάσω τα χρήματα που έχω πληρώσει σε περίπτωση ενός έκτακτου συμβάντος.
Κριτήρια Αποδοχής (Acceptance criteria) **Δεδομένης της κατάστασης (Given)** να ακυρώσω την κράτηση **Πότε συμβαίνει (When)** σε περίπτωση έκτακτης ανάγκης **Τότε <Then>** Σε περίπτωση ακύρωσης θα πρέπει να • **ΚΑΙ** Να επαληθεύσω ότι οι πελάτες μπορούν να ακυρώσουν τα χρήματα με 10% παρακράτησης με βάση την τιμή πληρωμής • **ΚΑΙ** Να επαληθεύσω ότι έχει σταλεί μήνυμα ακύρωσης στην επιχείρηση διοργάνωσης της παράστασης • **ΚΑΙ** Να επαληθεύσω ότι έχει σταλεί μήνυμα για την επιστροφή των χρημάτων στο υποσύστημα πληρωμών, • **ΚΑΙ** Να επαληθεύσω ότι έχει σταλεί ενημερωτικό μήνυμα στον πελάτη της παράστασης.

Κριτήριο αξιολόγησης 3

Χρησιμοποιώντας το πρότυπο που δόθηκε στον Πίνακα 7.1. περιγράψτε την ιστορία χρήστη για την οργάνωση μιας έκτακτης διάλεξης στο πλαίσιο ενός μαθήματος (με μέτρα COVID-19) σε μια αίθουσα ενός Πανεπιστημίου.

Απάντηση/Λύση

Τίτλος: Οργάνωση μαθήματος	**Προτεραιότητα:** Πολύ μεγάλη	**Εκτίμηση προσπάθειας:** 5
Ως (**As a**) Καθηγητής Πανεπιστημίου **Θέλω να** (**I want to**) οργανώσω μια έκτακτη διάλεξη στο πλαίσιο ενός μαθήματος **Ώστε** (**so that I can**) να κάνω αναπλήρωση ενός χαμένου μαθήματος.		
Κριτήρια Αποδοχής (Acceptance criteria) **Δεδομένης της κατάστασης (Given)** εύρεση διαθέσιμης ώρας στο πρόγραμμα μαθημάτων **Πότε συμβαίνει (When)** σε περίπτωση έκτακτου μαθήματος **Τότε <Then>** Σε περίπτωση έκτακτου μαθήματος θα πρέπει: • Να επαληθεύσω ότι το πρόγραμμα μαθημάτων έχει ενημερωθεί • Να αποστείλω μήνυμα στην Γραμματεία του Τμήματος ενημερώνοντας ότι έχω δεσμεύσει την ώρα **Δεδομένης της κατάστασης (Given)** εύρεση διαθέσιμης αίθουσας στο κτήριο		

Πότε συμβαίνει (When) σε περίπτωση έκτακτου μαθήματος

Τότε <Then> Σε περίπτωση έκτακτου μαθήματος θα πρέπει να

- **ΚΑΙ** Να επαληθεύσω ότι το πρόγραμμα διαθεσιμότητας αιθουσών έχει ενημερωθεί
- **ΚΑΙ** Να επαληθεύσω ότι έχω αποστείλει μήνυμα στην Γραμματεία του Τμήματος ενημερώνοντας ότι έχω δεσμεύσει την αίθουσα
- **ΚΑΙ** Να επαληθεύσω ότι έχει αναρτηθεί η ανακοίνωση
- **Ή** Να επαληθεύσω ότι έχει σταλεί email στους φοιτητές

Κριτήριο αξιολόγησης 4

Δίνεται η παρακάτω Εικόνα

Χρησιμοποιώντας το πρότυπο που δόθηκε στον Πίνακα 7.1. περιγράψτε την ιστορία χρήστη.

Απάντηση/Λύση

Τίτλος: CAPTCHA	**Προτεραιότητα:** Μεσαία	**Εκτίμηση προσπάθειας:** 5
Ως (**As a**) Χρήστης **Θέλω να** (**I want to**) μπορώ να αποδείξω ότι είμαι πραγματικός χρήστης **Ώστε** (**so that I can**) να μπορώ να κάνω login ξανά.		
Κριτήρια Αποδοχής (Acceptance criteria) **Δεδομένης της κατάστασης (Given)** ο χρήστης είναι μπλοκαρισμένος **Πότε συμβαίνει (When)** σε περίπτωση λάθους login **Τότε <Then>** Το σύστημα για τη δεδομένη διεύθυνση IP θα δείξει το παράθυρο CAPTCHA		

Κριτήριο αξιολόγησης 5

Περιγράψτε την ιστορία χρήστη για ένα εγγεγραμμένο χρήστη που έχει λησμονήσει το login όνομα ή το password του χρησιμοποιώντας το πρότυπο που δόθηκε στον Πίνακα 7.1.

Απάντηση/Λύση

Τίτλος: Υπενθύμιση login/password	**Προτεραιότητα:** Υψηλή	**Εκτίμηση προσπάθειας:** 5
Ως (**As a**) Εγγεγραμμένος χρήστης που δεν θυμάμαι το username/password **Θέλω να** (**I want to**) ανακτήσω το username/password **Ώστε** (**so that I can**) να μπορώ να κάνω login ξανά.		
Κριτήρια Αποδοχής (Acceptance criteria) **Δεδομένης της κατάστασης (Given)** ο χρήστης είναι μπλοκαρισμένος **Πότε συμβαίνει (When)** σε περίπτωση λάθους login ή όταν έχουμε απωλέσει το username/password **Τότε <Then>** η διεπαφή σύνδεσης πρέπει να εμφανίζει έναν σύνδεσμο με την ετικέτα "ξέχασα το username" ή "ξέχασα τον κωδικό πρόσβασης". • Κατά την πρόσβαση στην επιλογή ξεχασμένο username ή ξεχασμένος κωδικός πρόσβασης, ο χρήστης πρέπει να παράσχει το email του και το σύστημα θα εμφανίσει, "Εάν το email σας βρίσκεται στη βάση δεδομένων μας, θα λάβετε ένα email με τα βήματα για τη δημιουργία νέου κωδικού πρόσβασης". • Εάν το σύστημα βρει έναν χρήστη με ένα τέτοιο email, το σύστημα στέλνει ένα email με το όνομα χρήστη και έναν νέο προσωρινό κωδικό πρόσβασης που θα ισχύει για 10 λεπτά. • Ο χρήστης θα μπορεί να συνδεθεί με τον προσωρινό κωδικό πρόσβασης, αλλά θα πρέπει να τον αλλάξει για να χρησιμοποιήσει το σύστημα.		

Κριτήριο αξιολόγησης 6

Στο έργο του Ουίλιαμ Σαίξπηρ, «Μάκβεθ» η προφητεία που έλαβε ο Μάκβεθ λέει ότι «κανένας άνδρας που γεννήθηκε από γυναίκα δεν θα τον βλάψει». Αυτό τον κάνει να νιώθει άτρωτος και στο τέλος παλεύει με τον Μακντάφ, σίγουρος για τη νίκη. Αλλά ο Μακντάφ κερδίζει τον αγώνα και σκοτώνει τον Μάκβεθ επειδή:

"Despair thy charm;

And let the angel whom thou still hast served

Tell thee, Macduff was from his mother's womb

Untimely ripp'd."

Η ελεύθερη μετάφραση του παραπάνω αποσπάσματος είναι «Παρά την προφητεία, πες στο Σατανά που υπηρετείς ότι ο Μακντάφ δεν γεννήθηκε, αλλά η μήτρα της μάνας του σκίστηκε». Μερικές επεξηγήσεις για τα παραπάνω:

1) **Ο Μακντάφ θεωρούσε το Μάκβεθ πολύ κακό και συνεπώς ο άγγελος είναι ο σατανάς**
2) **Ο Μακντάφ όταν λέει ότι η μήτρα της μητέρας του σκίστηκε εννοεί ότι γεννήθηκε με καισαρική τομή.**

Αν ο Μάκβεθ γνώριζε ιστορίες χρηστών πως θα έγραφε ως ιστορία χρήστη το παρακάτω απόσπασμα;

Πρωτότυπο	Μετάφραση
Be bloody, bold, and resolute; laugh to scorn ***The power of man, for none of woman born*** ***Shall harm Macbeth.***	**Να είσαι αιμοσταγής, τολμηρός και αποφασιστικός, γέλα με περιφρόνηση** **Δεν υπάρχει άντρας, που να γεννήθηκε από γυναίκα** **Που μπορεί να βλάψει τον Μάκβεθ.**

Απάντηση/Λύση

Η ιστορία χρήστη έχει ως ακολούθως (https://www.luxoft-training.com/news/shakespeare-s-macbeth-and-agile-product-management/):

Τίτλος: Η τυραννία του Μάκβεθ	**Προτεραιότητα:**	**Εκτίμηση προσπάθειας:**
Ως (**As a**) τύραννος **Θέλω να** (**I want to**) είμαι άτρωτος στους εχθρούς μου **Ώστε** (**so that I can**) να συνεχίσω να διαπράττω φρικαλεότητες χωρίς φόβο εκδίκησης.		
Κριτήρια Αποδοχής (Acceptance criteria) **Δεδομένης της κατάστασης (Given)** Υπάρχει τύραννος • **ΚΑΙ** ο τύραννος σκοτώνει κάποιον • **ΚΑΙ** κάποιος από την οικογένεια ζητάει εκδίκηση • **ΚΑΙ** έχει γεννηθεί με φυσιολογικό τοκετό **Πότε συμβαίνει (When)** θέλει να κάνει κακό στον τύραννο **Τότε <Then>** Δεν μπορεί να κάνει κακό στον τύραννο		

Κριτήριο αξιολόγησης 7

Α) Χρησιμοποιώντας τη μέθοδο ιεράρχησης Wieger αξιολογήστε τα χαρακτηριστικά του πίνακα για ένα σύστημα διαχείρισης χημικών υλικών. Συμπληρώστε τα κελιά με κίτρινο χρώμα.

Β) Πώς μπορεί να μεταβληθούν τα βάρη που αναφέρονται στο κόστος και στον κίνδυνο. Πώς θα πρέπει να μεταβληθούν τα βάρη για ένα έργο χωρίς ιδιαίτερα σημαντικά τεχνικά προβλήματα.

Το φύλλο εργασίας μπορείτε να το κάνετε download από τη διεύθυνση https://cs.anu.edu.au/courses/comp3120/public_docs/Requirements%20Prioritization%20Worksheet.xls

	A	B	C	D	E	F	G	H	I	J
1	Σχετικά βάρη	2	1			1		1		
2										
3	**Χαρακτηριστικό**	**Σχετικό Όφελος**	**Σχετική Ποινή**	**Συνολική Αξία**	**Αξία %**	**Σχετικό Κόστος**	**Κόστος%**	**Σχετικός κίνδυνος**	**Κίνδυνος %**	**Προτεραιότητα**
4	Εκτυπώστε ένα δελτίο δεδομένων ασφαλείας υλικού			=B4*$B$1+C4*$C$1	=100*D4/$D$14		=100*F4/$F$14		=100*H4/$H$14	=E4/(G4*$F$1+I4*H
5	Κατάσταση παραγγελίας πωλητή			=B5*$B$1+C5*$C$1	=100*D5/$D$14		=100*F5/$F$14		=100*H5/$H$14	=E5/(G5*$F$1+I5*H
6	Δημιουργία μια αναφορά αποθέματος			=B6*$B$1+C6*$C$1	=100*D6/$D$14		=100*F6/$F$14		=100*H6/$H$14	=E6/(G6*$F$1+I6*H
7	Δείτε το ιστορικό ενός συγκεκριμένου δοχείου χημικών			=B7*$B$1+C7*$C$1	=100*D7/$D$14		=100*F7/$F$14		=100*H7/$H$14	=E7/(G7*$F$1+I7*H
8	Αναζητήστε καταλόγους προμηθευτών για μια συγκεκριμένη χημική ουσία			=B8*$B$1+C8*$C$1	=100*D8/$D$14		=100*F8/$F$14		=100*H8/$H$14	=E8/(G8*$F$1+I8*H
9	Διατηρήστε μια λίστα με επικίνδυνες χημικές ουσίες			=B9*$B$1+C9*$C$1	=100*D9/$D$14		=100*F9/$F$14		=100*H9/$H$14	=E9/(G9*$F$1+I9*H
10	Τροποποιήστε ένα εκκρεμές αίτημα για χημικά			=B10*$B$1+C10*C	=100*D10/$D$14		=100*F10/$F$14		=100*H10/$H$14	=E10/(G10*$F$1+I10*
11	Δημιουργήστε μια ατομική αναφορά απογραφής εργαστηρίου			=B11*$B$1+C11*C	=100*D11/$D$14		=100*F11/$F$14		=100*H11/$H$14	=E11/(G11*$F$1+I11*
12	Ελέγξτε τη βάση δεδομένων εκπαίδευσης για αρχείο εκπαίδευσης επικίνδυνων χημικών			=B12*$B$1+C12*C	=100*D12/$D$14		=100*F12/$F$14		=100*H12/$H$14	=E12/(G12*$F$1+I12*
13	Εισαγωγή χημικών δομών από εργαλεία σχεδίασης κατασκευών			=B13*$B$1+C13*C	=100*D13/$D$14		=100*F13/$F$14		=100*H13/$H$14	=E13/(G13*$F$1+I13*
14	**Totals**	**=SUM(B4:B13)**	**=SUM(C4:C13)**	**=SUM(D4:D13)**	**=SUM(E4:E13)**	**=SUM(F4:F13)**	**=SUM(G4:G13)**	**=SUM(H4:H13)**	**=SUM(I4:I13)**	

Απάντηση/Λύση

Η αξιολόγηση γίνεται συμπληρώνοντας τα κελιά του πίνακα με κίτρινο χρώμα. Η βαρύτητα των κριτηρίων σχετικό όφελος και σχετική ποινή αλλάζει αλλάζοντας τις τιμές στα κελιά Β1 και C1. Αντίστοιχα η τιμή των κριτηρίων σχετικό κόστος και σχετικός κίνδυνος αλλάζει με την συμπλήρωση των κελιών Φ1 και Η1. Σε ένα έργο με μικρές τεχνικές δυσκολίες το βάρος του κριτηρίου για τον κίνδυνο μπορεί να είναι μικρότερο της μονάδας, ενώ για ένα δύσκολο τεχνικό το κριτήριο θα μπορούσε να έχει βαρύτητα 2.

	A	B	C	D	E	F	G	H	I	J
1	**Σχετικά βάρη**	2,0	1,0			1,0		1		
2										
3	**Χαρακτηριστικό**	**Σχετικό Όφελος**	**Σχετική Ποινή**	**Συνολική Αξία**	**Αξία %**	**Σχετικό Κόστος**	**Κόστος%**	**Σχετικός κίνδυνος**	**Κίνδυνος %**	**Προτεραιότητα**
4	Εκτυπώστε ένα δελτίο δεδομένων ασφαλείας υλικού	2	4	8	5,2	1	2,7	1	3,0	0,90
5	Κατάσταση παραγγελίας πωλητή	5	3	13	8,4	2	5,4	1	3,0	0,99
6	Δημιουργία μια αναφορά αποθέματος	9	7	25	16,1	5	13,5	3	9,1	0,71
7	Δείτε το ιστορικό ενός συγκεκριμένου δοχείου χημικών	5	5	15	9,7	3	8,1	2	6,1	0,68
8	Αναζητήστε καταλόγους προμηθευτών για μια συγκεκριμένη χημική ουσία	9	8	26	16,8	3	8,1	8	24,2	0,52
9	Διατηρήστε μια λίστα με επικίνδυνες χημικές ουσίες	3	9	15	9,7	3	8,1	4	12,1	0,48
10	Τροποποιήστε ένα εκκρεμές αίτημα για χημικά	4	3	11	7,1	3	8,1	2	6,1	0,50
11	Δημιουργήστε μια ατομική αναφορά απογραφής εργαστηρίου	6	2	14	9,0	4	10,8	3	9,1	0,45
12	Ελέγξτε τη βάση δεδομένων εκπαίδευσης για αρχείο εκπαίδευσης επικίνδυνων χημικών	3	4	10	6,5	4	10,8	2	6,1	0,38
13	Εισαγωγή χημικών δομών από εργαλεία σχεδίασης κατασκευών	7	4	18	11,6	9	24,3	7	21,2	0,26
14	**Totals**	**53**	**49**	**155**	**100,0**	**37**	**100,0**	**33**	**100,0**	

Κριτήριο αξιολόγησης 8

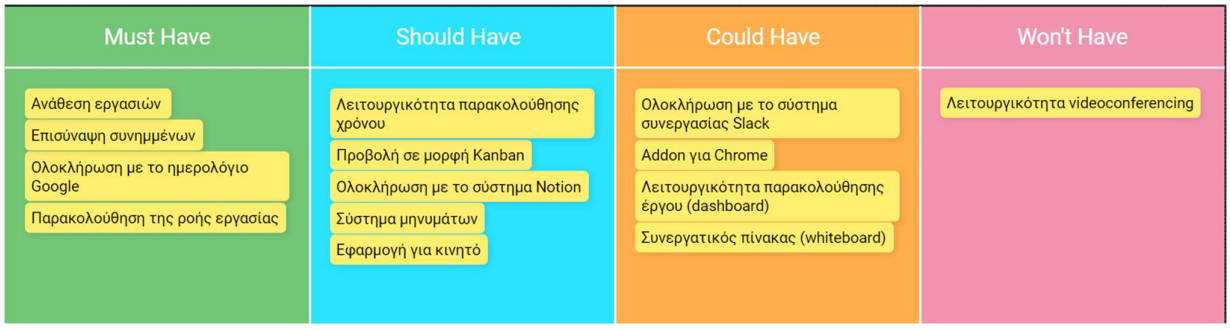
Δίνονται οι παρακάτω απαιτήσεις για ένα σύστημα διαχείρισης έργων:

- **Παρακολούθηση της ροής εργασίας**
- **Λειτουργικότητα παρακολούθησης χρόνου**
- **Προβολή σε μορφή Kanban**
- **Ολοκλήρωση με το σύστημα συνεργασίας Slack**
- **Ολοκλήρωση με το σύστημα Notion**
- **Σύστημα μηνυμάτων**
- **Εφαρμογή για κινητό**
- **Addon για Chrome**
- **Ανάθεση εργασιών**
- **Ολοκλήρωση με το ημερολόγιο Google**
- **Λειτουργικότητα video conferencing**
- **Συνεργατικός πίνακας (whiteboard)**
- **Λειτουργικότητα παρακολούθησης έργου (dashboard)**
- **Επισύναψη συνημμένων**

Με τη χρήση της μεθόδου MoSCoW κατατάξτε τις ανωτέρω απαιτήσεις σε κατηγορίες. Όπου απαιτείται κάντε τις παραδοχές σας.

Απάντηση/Λύση

Προφανώς η προτεινόμενη κατάταξη είναι ενδεικτική. Υπάρχει μεγάλος αριθμός διαγραμματικών εργαλείων που μπορούν να χρησιμοποιηθούν για αυτή την εργασία. Ο παρακάτω πίνακας έγινε με τη χρήση https:/draft.io.



Κριτήριο αξιολόγησης 9

Έστω ένα έργο ανάπτυξης λογισμικού το οποίο θα εκτελεστεί σε τρία sprint.

- **Στο πρώτο sprint, η ομάδα έργου έχει δεσμευτεί ότι θα υλοποιήσει 8 ιστορίες χρηστών όπου κάθε ιστορία έχει μετρηθεί με 3 σημεία ιστορίας. Στην πραγματικότητα όμως η ομάδα ολοκληρώνει μόνο 4 ιστορίες χρηστών.**

- **Στο δεύτερο sprint η ομάδα σας έχει δεσμευτεί να υλοποιήσει 10 ιστορίες χρηστών και κάθε ιστορία ισούται με 5 πόντους ιστορίας. Στην πράξη η ομάδα ολοκλήρωσε μόνο 7 ιστορίες χρηστών.**
- **Στο τρίτο sprint, η ομάδα σας δεσμεύεται για 9 ιστορίες χρηστών και κάθε ιστορία ισούται με 5 πόντους ιστορίας. Η ομάδα σας υλοποιεί στην πραγματικότητα 7 ιστορίες χρηστών.**

Να υπολογίσετε:

Α) την αναμενόμενη ταχύτητα και

Β) την πραγματική ταχύτητα

Απάντηση/Λύση

Η αναμενόμενη ταχύτητα υπολογίζεται με βάση τη δέσμευση της ομάδας για υλοποίηση των ιστοριών χρηστών ανά sprint. Συνεπώς:

$$\boldsymbol{Αναμενόμενη\ ταχύτητα} = \frac{(\mathbf{8} * \mathbf{3}) + (\mathbf{10} * \mathbf{5}) + (\mathbf{9} * \mathbf{5})}{\mathbf{3}} = \frac{\mathbf{24} + \mathbf{50} + \mathbf{45}}{\mathbf{3}} = \frac{\mathbf{119}}{\mathbf{3}} = \mathbf{29,66}$$

Αντίστοιχα η πραγματική ταχύτητα υπολογίζεται με βάση τις ιστορίες χρηστών που υλοποιήθηκαν.

$$\boldsymbol{Πραγματική\ ταχύτητα} = \frac{(\mathbf{4} * \mathbf{3}) + (\mathbf{7} * \mathbf{5}) + (\mathbf{7} * \mathbf{5})}{\mathbf{3}} = \frac{\mathbf{12} + \mathbf{35} + \mathbf{35}}{\mathbf{3}} = \frac{\mathbf{82}}{\mathbf{3}} = \mathbf{27,33}$$

Προφανώς η πραγματική ταχύτητα είναι μικρότερη από την αναμενόμενη και συνεπώς θα πρέπει να αναζητηθούν οι λόγοι.

Η ευελιξία στις επιχειρήσεις

A system must be managed. It will not manage itself.

Left to themselves, components become selfish, competitive,

independent profit centers, and thus destroy the system.

The secret is cooperation between components toward the aim of the organization.

W. Edwards Deming

Σύνοψη

Το όγδοο κεφάλαιο παρουσιάζει πως οι ευέλικτες αρχές εφαρμόζονται γενικότερα στις επιχειρήσεις και σε διάφορους επιχειρηματικούς κλάδους. Η εφαρμογή των ευέλικτων μεθόδων είναι ιδιαίτερα σημαντική σε διαφορετικές λειτουργίες της επιχείρησης και όχι μόνο στην ανάπτυξη λογισμικού, ενώ τα τελευταία χρόνια είναι ιδιαίτερα δημοφιλής. Στις επόμενες παραγράφους θα παρουσιάσουμε συνοπτικά κάποιες περιπτώσεις εφαρμογής της ευέλικτης προσέγγισης σε διαφορετικούς κλάδους της επιχειρηματικής/οικονομικής δραστηριότητας.

8 Η ευελιξία στις επιχειρήσεις

Τα τελευταία 20 χρόνια πολλά έχουν αλλάξει στις επιχειρήσεις. Για παράδειγμα, σε πολλές περιπτώσεις η παροχή υπηρεσιών γίνεται μέσω του διαδικτύου και όχι σε φυσικά καταστήματα. Οι καταναλωτές θεωρούν δεδομένη την πρόσβαση των υπηρεσιών από το διαδίκτυο, την ευκολία χρήσης των υπηρεσιών, τους γρήγορους χρόνους απόκρισης, την 24/7 πρόσβαση στις υπηρεσίες που χρειάζονται, ενώ η τάση αυτή είναι διαρκώς επιταχυνόμενη. Οι προκλήσεις της διάχυτης παροχής υπηρεσιών (ubiquitous service delivery) αυξάνονται καθώς νέες τεχνολογίες γίνονται διαθέσιμες. Οι σημερινές επιχειρήσεις αντιλαμβάνονται ότι δεν ανταγωνίζονται πλέον μόνο τοπικούς, ή ακόμη και εθνικούς παρόχους υπηρεσιών, αλλά παρόχους υπηρεσιών, που προσφέρουν ισοδύναμα προϊόντα και υπηρεσίες από όλο τον κόσμο. Επιπλέον σήμερα οι επιχειρήσεις έχουν αλλάξει οργανωτική δομή και αποτελούνται από ένα δίκτυο ομάδων με ανθρωποκεντρική κουλτούρα που μαθαίνει γρήγορα, λαμβάνει αποφάσεις γρήγορα με τη χρήση τεχνολογίας και καθοδηγούνται από έναν ισχυρό κοινό σκοπό, τη δημιουργία αξίας, για όλα τα ενδιαφερόμενα μέρη. Μόνο με αυτό τον τρόπο, με ταχύτητα και προσαρμοστικότητα μπορούν να δημιουργήσουν ανταγωνιστικό πλεονέκτημα σε ασταθείς, αβέβαιες, περίπλοκες και διφορούμενες επιχειρηματικές συνθήκες (Volatile, Uncertain, Complex, and Ambiguous - VUCA).

Πού οδηγούν αυτές οι θεμελιώδες αλλαγές τη σημερινή επιχείρηση; Εάν οι υπάρχουσες επιχειρηματικές διαδικασίες ήδη δυσκολεύονται στο να εξυπηρετήσουν τους υπάρχοντες πελάτες, πώς η επιχείρηση θα αντιμετωπίσει την αναγκαία βελτίωση στην παροχή υπηρεσιών χωρίς να μειώσει τα κέρδη της; Πώς μπορεί η επιχείρηση να μειώσει τον χρόνο παροχής υπηρεσιών (time to market), ώστε να έχει ανταγωνιστικό πλεονέκτημα και θετική δημόσια εικόνα; Πώς θα μπορέσει να εξασφαλίσει ότι τα προϊόντα και οι υπηρεσίες που παρέχει η επιχείρηση δεν καθίστανται παρωχημένα, επειδή το υπερβολικό κόστος, ο χρόνος ή οι πόροι που απαιτούνται για την αποτελεσματική παράδοση αυξάνονται;

Ακόμη και οι πιο εύρωστες επιχειρήσεις συνειδητοποιούν ότι οι επιχειρηματικές διαδικασίες και πρακτικές στις οποίες βασίστηκαν στο παρελθόν θα πρέπει να αλλάξουν και να γίνουν πιο αποτελεσματικές

ώστε να μπορέσουν να ανταποκριθούν στις νέες συνθήκες. Ένα από τα πιο ενδιαφέροντα χαρακτηριστικά στον κόσμο των επιχειρήσεων είναι ότι οι περισσότερες επιχειρήσεις αντιμετωπίζουν τις ίδιες βασικές προκλήσεις και προβλήματα, όπως για παράδειγμα:

- χαμένες (ή καθυστερημένες) παραδόσεις προϊόντων
- μεγάλες αυξήσεις στα κοστολόγια και στον προϋπολογισμό
- καταπονημένοι και αγχωμένοι εργαζόμενοι
- προβλήματα στη διαχείριση της γνώσης (π.χ. σιλό γνώσης)
- κ.λπ.

Το αξιοσημείωτο είναι τα παραπάνω προβλήματα ανεξάρτητα αν η επιχείρηση είναι μια μικρή συμβουλευτική εταιρεία ή μια μεγάλη πολυεθνική, είτε δραστηριοποιείται στον ιδιωτικό ή δημόσιο τομέα είναι τα ίδια ή παρόμοια. Κατά την ίδια περίοδο των τελευταίων 20 ετών, οι τεχνολογικές καινοτομίες του διαδικτύου δημιούργησαν ένα περιβάλλον κατάλληλο για παροχή ψηφιακών υπηρεσιών στην παγκόσμια αγορά, ενώ ταυτόχρονα επιχειρήσεις δύο κλάδων (τεχνολογίας πληροφοριών και κατασκευής προϊόντων - manufacturing) έχουν εφαρμόσει ένα σύνολο επιχειρηματικών πρακτικών και τεχνικών (γνωστές ως η λιτή αι ευέλικτη προσέγγιση) που επέτρεψαν τη δημιουργία ενός πιο αποτελεσματικού περιβάλλοντος εργασίας.

Η επιχειρηματική ευελιξία ορίζεται ως η ευελιξία στην κουλτούρα, την ηγεσία, τη στρατηγική και τη διακυβέρνηση ενός οργανισμού που προσθέτει αξία για όλα τα ενδιαφερόμενα μέρη που λειτουργούν σε αβέβαια, περίπλοκα και διφορούμενα περιβάλλοντα (https://www.agilebusiness.org/page/WhatisBusinessAgility).

Η επιτυχία της ευέλικτης προσέγγισης βασίζεται σε ένα πυρήνα 12 αρχών, που έχουμε ήδη παρουσιάζει, αλλά παρουσιάζουμε ξανά από μια διαφορετική οπτική γωνία:

- **Ευέλικτος Σχεδιασμός (Responsive planning)**: περιλαμβάνει την κατανομή των μακροπρόθεσμων στόχων σε συντομότερους κύκλους παράδοσης και στη συνέχεια την προσαρμογή του επόμενου κύκλου εργασίας με βάση τα αποτελέσματα του προηγούμενου κύκλου εργασίας.
- **Εργασία με γνώμονα την επιχειρηματική αξία (Business-value-driven work)**: περιλαμβάνει την ιεράρχηση της εργασίας σύμφωνα με την επιχειρηματική αξία που κάθε δραστηριότητα είναι πιθανό να αποφέρει στον οργανισμό.
- **Πρακτικά επιχειρηματικά αποτελέσματα**: περιλαμβάνει την τακτική επιθεώρηση των αποτελεσμάτων, προκειμένου να καθοριστεί εάν πληρούνται οι επιχειρηματικές απαιτήσεις – και εάν παρέχεται η αναγκαία επιχειρηματική αξία.
- **Δέσμευση συμμετεχόντων (stakeholder engagement)**: περιλαμβάνει ενεργό συμμετοχή εσωτερικών και εξωτερικών πελατών κατά τη διάρκεια μιας επιχειρηματικής διεργασίας με σκοπό να διασφαλιστεί ότι τα παραδοτέα που προκύπτουν ανταποκρίνονται στις προσδοκίες τους.
- **Σταθερά ορόσημα**: ύπαρξη σταθερών ορόσημων με σκοπό να ενθαρρύνουν τα μέλη του προσωπικού να προσφέρουν σταθερά αξία στον οργανισμό.
- **Κινητοποίηση των εργαζομένων (self-motivation):** περιλαμβάνει τη χρήση αυτοοργανούμενων ομάδων για παροχή καλύτερων αποτελεσμάτων πάντα υπό την καθοδήγηση και την επίβλεψη του πελάτη.
- **Επικοινωνία «Just-in-time»**: αντικαθιστά τις παραδοσιακές εταιρικές συναντήσεις με πιο αποτελεσματικές τεχνικές επικοινωνίας και μεταφοράς γνώσης.
- **Διαρκής και άμεση παρακολούθηση κατάστασης**: η επιχείρηση παρέχει εργαλεία που επιτρέπουν στο προσωπικό να διατηρεί συνεχώς ενήμερους άλλους συμμετέχοντες στον οργανισμό για την κατάσταση της εργασίας που εκτελούν.

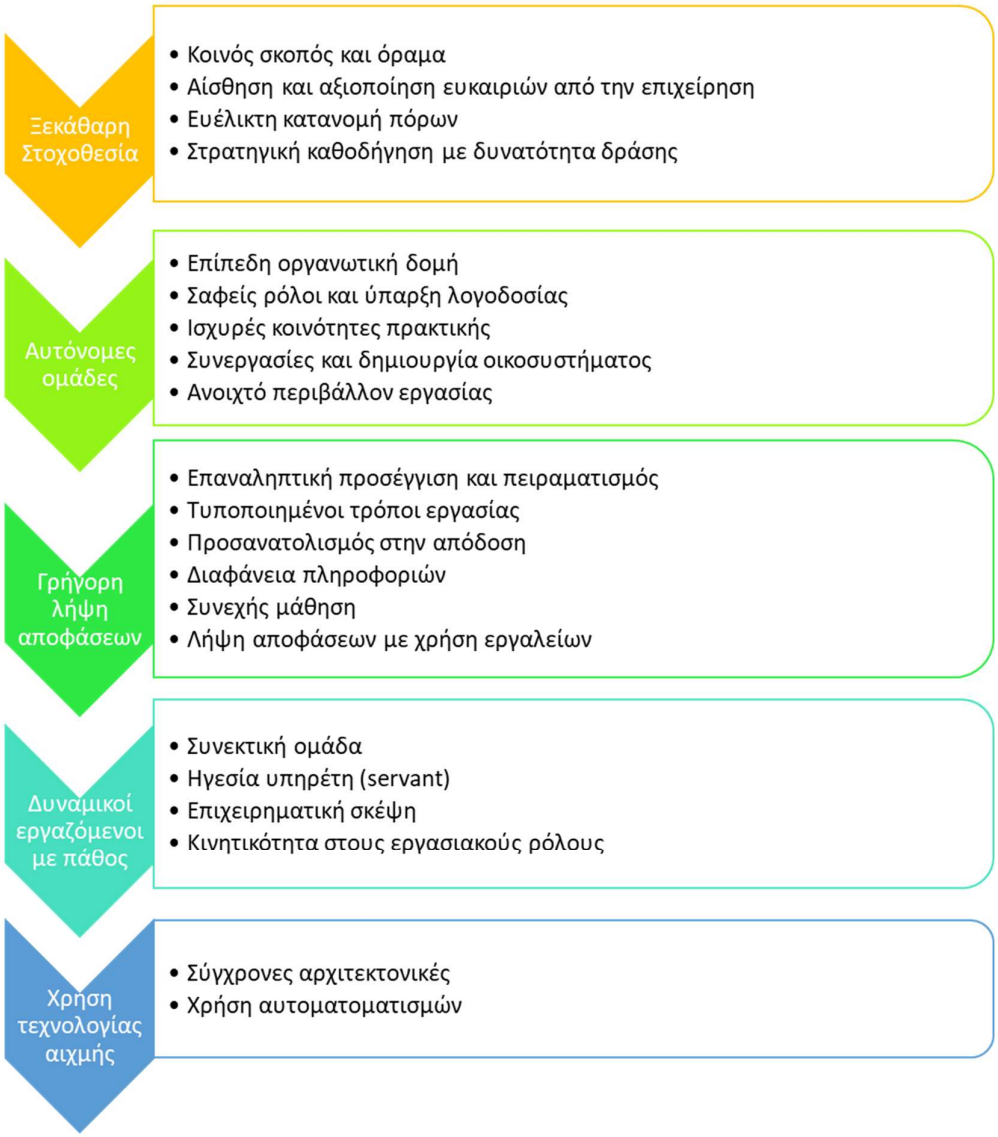
- **Διαχείριση των περιττών (waste)**: περιλαμβάνει τη βέλτιστη χρήση των πόρων του οργανισμού με τη μείωση και, όπου είναι δυνατόν, ή και την εξάλειψη δραστηριοτήτων χαμηλής επιχειρηματικής αξίας.
- **Διαρκώς μετρήσιμη ποιότητα**: περιλαμβάνει τη δημιουργία ενεργών σημείων ελέγχου όπου οι επιχειρήσεις μπορούν να αξιολογήσουν τα αποτελέσματα τόσο σε ποιοτικές όσο και σε ποσοτικές μετρήσεις.
- **Αξιολόγηση της εργασίας σε τακτά χρονικά διαστήματα**: η επιχείρηση παρέχει στο προσωπικό εργαλεία για την τακτική παρακολούθηση και συνεχή βελτίωση της εργασίας του.
- **Συνεχής βελτίωση**: περιλαμβάνει την τακτική αναθεώρηση και προσαρμογή των επιχειρηματικών δραστηριοτήτων ώστε να διασφαλιστεί ότι ο οργανισμός ανταποκρίνεται στις ανάγκες της αγοράς καθώς και των πελατών.

Σύμφωνα με την εταιρεία συμβούλων McKinsey (2017) πέντε είναι τα βασικά χαρακτηριστικά των ευέλικτων επιχειρήσεων/οργανισμών. Αυτά είναι (βλέπε Εικόνα 8.1):

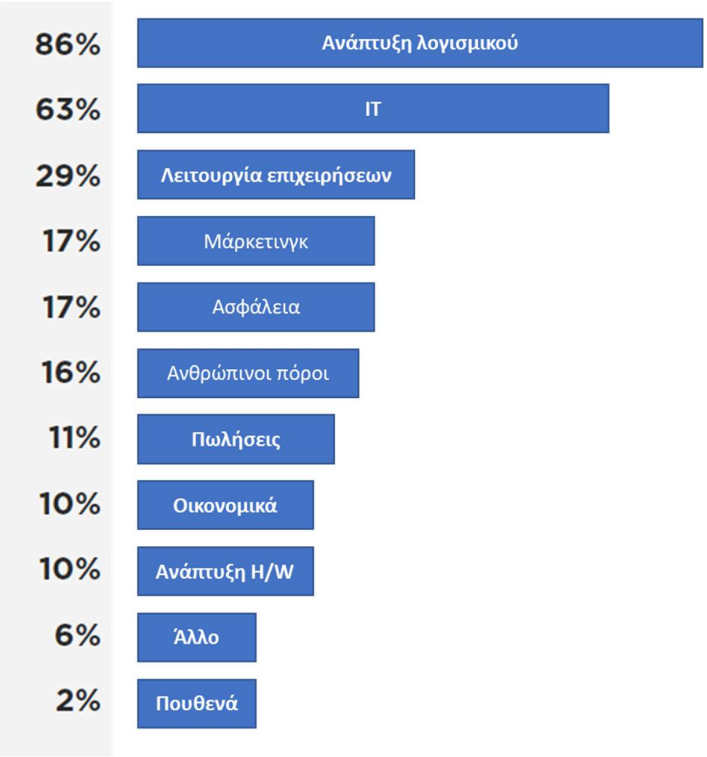
- Κοινός σκοπός και στόχοι ξεκάθαροι αλλά και καλά γνωστοί σε όλους τους συμμετέχοντες. Αυτός ο κοινός σκοπός και όραμα είναι ο «πολικός αστέρας» της επιχείρησης στον επιχειρηματικό ουρανό, που όλοι οι εργαζόμενοι αλλά και οι πελάτες μπορούν να τον αναγνωρίσουν.
- Η επιχείρηση είναι ένα δίκτυο ισχυρών, ανεξάρτητων, αυτοοργανούμενων ομάδων.
- Η επιχείρηση έχει ικανότητα για γρήγορη λήψη αποφάσεων με χρήση ψηφιακών εργαλείων και γρήγορους κύκλους μάθησης.
- Η επιχείρηση έχει δυναμικούς εργαζόμενους που παρακινούνται από το πάθος δημιουργίας και καινοτομίας και όχι αποκλειστικά από οικονομικά κίνητρα.
- Η επιχείρηση χρησιμοποιεί τεχνολογία αιχμής ώστε να έχει ανταγωνιστικό πλεονέκτημα.

Η ευέλικτη προσέγγιση έχει τεράστιες δυνατότητες εφαρμογής στις υπηρεσίες, όπως για παράδειγμα σε επιχειρήσεις/οργανισμούς υγείας, την εκπαίδευση, στις δημόσιες υπηρεσίες, στις υπηρεσίες κοινής ωφέλειας κ.λπ. Σε πολλές περιπτώσεις, η ευέλικτη προσέγγιση χρησιμοποιείται ήδη, ωστόσο, υπάρχει άπλετος χώρος για τη δημιουργία λιτών συστημάτων εξοικονόμησης κόστους και βελτίωσης της ποιότητας.

Συνεπώς, η εφαρμογή των ευέλικτων μεθόδων είναι ιδιαίτερα σημαντική σε διαφορετικούς κλάδους αλλά και διαφορετικές λειτουργίες της επιχείρησης και όχι μόνο στην ανάπτυξη λογισμικού, όπως φαίνεται στην Εικόνα 8.2. (Digital.ai, 2021). Στις επόμενες παραγράφους θα παρουσιάσουμε συνοπτικά κάποιες περιπτώσεις εφαρμογής της ευέλικτης προσέγγισης σε διαφορετικούς κλάδους οικονομικής δραστηριότητας καθώς και σε διαφορετικές λειτουργίες της επιχείρησης. Προφανώς η παρουσία είναι επιλεκτική αφού η ευελιξία χρησιμοποιείται σε πληθώρα περιπτώσεων.



Εικόνα 8.1: *Τα πέντε βασικά χαρακτηριστικά των ευέλικτων οργανισμών*



Εικόνα 8.2: *Η διείσδυση των ευέλικτων μεθόδων σε διαφορετικές λειτουργίες μιας επιχείρησης*

8.1 Η ευέλικτη προσέγγιση στην εκπαίδευση

Τα τελευταία χρόνια υπάρχει παγκοσμίως ένα αυξανόμενο ενδιαφέρον για το πώς οι ευέλικτες μέθοδοι, που αναπτύχθηκαν στη βιομηχανία λογισμικού, μπορούν να εφαρμοστούν στην εκπαίδευση. Ομοίως, έχουμε δει ότι η λιτή σκέψη, που αναπτύχθηκε μέσω της λιτής διοίκησης παραγωγής από την ιαπωνική αυτοκινητοβιομηχανία, είναι επίσης ένας σημαντικός τομέας ενδιαφέροντος για τους εκπαιδευτικούς. Είναι προφανές ότι ερευνητές και επαγγελματίες σε όλο τον κόσμο διερευνούν ολοένα και περισσότερο τον τρόπο με τον οποίο οι ευέλικτες και λιτές τεχνικές μπορούν να χρησιμοποιηθούν στην εκπαίδευση (Parson & MacCallum, 2019). Η εκπαίδευση είναι ένας κλάδος στον οποίο η εφαρμογή των ευέλικτων μεθόδων μπορεί να έχει εφαρμογή με διάφορους τρόπους:

- ως ένα εκπαιδευτικό εργαλείο
- ως ένα εργαλείο ανάπτυξης της συνεργασίας και της ομαδικότητας των εκπαιδευτικών, αλλά και
- ως ένα εργαλείο για την καλύτερη εκτέλεση των έργων της εκπαιδευτικής μονάδας (π.χ. κατασκευή ενός νέου γυμναστηρίου, διοργάνωση μιας εκδήλωσης, μιας σχολικής θεατρικής παράστασης ή τη συμμετοχή σε ένα έργο ERASMUS+).

Η εκπαίδευση των νέων και η διασφάλιση ότι έχουν την ικανότητα να δημιουργούν καινοτόμες λύσεις σε προβλήματα και να λαμβάνουν σωστές αποφάσεις είναι η κορυφαία προτεραιότητα κάθε κοινωνίας και πολιτισμού. Η δημόσια εκπαίδευση όταν δημιουργήθηκε στη σημερινή της μορφή, σχεδόν 70 χρόνια πριν, σχεδιάστηκε για να καλύψει τις ανάγκες των ατόμων σε βασικές γνώσεις. Οι ανάγκες ήταν πολύ πιο απλές, όπως και ο κόσμος της εργασίας. Σήμερα, ωστόσο, η πολυπλοκότητα έχει αυξηθεί εκθετικά, όπως και οι επιλογές των μαθητών για θέσεις εργασίας και διαφορετικούς ρόλους.

Λόγω αυτής της αλλαγής στο εργασιακό τοπίο, η εκπαίδευση πρέπει να προετοιμάσει τα παιδιά να συμμετέχουν αποτελεσματικά σε αυτό. Όμως οι εκπαιδευτικοί εξακολουθούν να εκπαιδεύονται, να προετοιμάζονται και να μεταφέρουν τις γνώσεις σχεδόν με τον ίδιο τρόπο.

Η εκπαίδευση αντιμετωπίζει σήμερα διαφορετικές προκλήσεις σε σχέση με το παρελθόν. Το πρόγραμμα σπουδών διευρύνεται, οι μαθητές με χαμηλή απόδοση απαιτούν περισσότερη προσοχή, το μέγεθος των τάξεων αυξάνεται, κ.λπ.

Οι απαιτήσεις των προγραμμάτων σπουδών για τους εκπαιδευτικούς επεκτείνονται εδώ και αρκετές δεκαετίες. Οι λόγοι της διεύρυνσης του προγράμματος σπουδών προέρχονται από το γεγονός ότι:

- Η σύγχρονη εκπαίδευση είναι πολύπλευρη και πρέπει να καλύψει ποικιλία θεμάτων π.χ. πρόληψη του εκφοβισμού, υγιεινή διατροφή και παρακολούθηση της παχυσαρκίας, ανάπτυξη επιχειρηματικών δεξιοτήτων, κ.λπ.
- Οι μαθητές έχουν πλήθος εμπειριών και γνώσεων κάτι που θα ήταν αδιανόητο πριν από μερικές δεκαετίες. Από τα πρώτα τους χρόνια χρησιμοποιούν ηλεκτρονικούς υπολογιστές ενώ έρχονται σε επαφή με θέματα «ενηλίκων» μέσω της τηλεόρασης, των ταινιών και της μουσικής.
- Το εκτενές πρόγραμμα σπουδών και η πληθώρα γνώσεων αναγκάζει τους εκπαιδευτικούς να προχωρούν σε επόμενες έννοιες χωρίς συχνά οι μαθητές να έχουν εμπεδώσει τα προηγούμενα.
- Το μέγεθος της τάξης έχει αυξηθεί σημαντικά, ειδικά στην ανώτατη εκπαίδευση.

Οι ευέλικτες μέθοδοι μπορούν να δώσουν λύση σε μερικά από τα παραπάνω ζητήματα και έχει αποδειχτεί ότι λειτουργούν αποτελεσματικά και σε εκπαιδευτικά περιβάλλοντα. Για παράδειγμα, η χρήση Scrum στην εκπαίδευση επιτρέπει να θέτουμε στόχους, να έχουμε όραμα, να καθορίζουμε εστιασμένους μαθησιακούς στόχους όπου επιτρέπουμε στους μαθητές να προσαρμόσουν το στυλ μάθησης στην ταχύτητα μάθησης. Εστιάζοντας σε έναν μαθησιακό στόχο κάθε φορά, οι εκπαιδευόμενοι μπορούν να κατακτήσουν το κάθε θέμα και επίπεδο πριν προχωρήσουν σε κάποιο επόμενο. Αντίστοιχα, οι εκπαιδευτικοί μπορούν να δώσουν έμφαση σε μαθησιακά αντικείμενα υψηλής προτεραιότητας. Επιπλέον, το Scrum παραμένει αποτελεσματικό σε μεγάλες τάξεις επειδή μπορούν να δημιουργηθούν μικρότερες ομάδες Scrum για έργα.

Οι ευέλικτες μέθοδοι στην τάξη βοηθούν τους μαθητές να προσαρμοστούν δημιουργικά και ευέλικτα στις αλλαγές, να δώσουν προτεραιότητα σε έργα και ιδέες και να βρουν νέες λύσεις. Η τεχνολογία και τα μέσα ενημέρωσης αλλάζουν τόσο γρήγορα που οι πληροφορίες που λαμβάνουν τα παιδιά είναι πάντα σε ροή. Με τις ευέλικτες μεθόδους, μαθαίνουν οι μαθητές από πολύ νωρίς, πώς να μετατρέπουν την αλλαγή σε πλεονέκτημα.

Για να κατανοήσουμε καλύτερα το πως εφαρμόζεται η ευέλικτη προσέγγιση στην εκπαίδευση παραθέτουμε στον παρακάτω πίνακα την αντιστοίχιση των αξιών που ορίζονται στο Agile Manifesto με αυτές που θα μπορούσαν να χρησιμοποιηθούν σε ένα εκπαιδευτικό περιβάλλον (Stewart et al., 2009).

	Agile Manifesto	**Agile Manifesto στην εκπαίδευση**
1	Άτομα και αλληλεπιδράσεις, αντί για διεργασίες και εργαλεία.	Οι μαθητές είναι πάνω από τις παραδοσιακές εκπαιδευτικές διαδικασίες και εργαλεία.
2	Λογισμικό σε λειτουργία αντί για λεπτομερή τεκμηρίωση	Εργασία μέσω έργων αντί για ολοκληρωμένης τεκμηρίωσης.
3	Συνεργασία με τους πελάτες, αντί για διαπραγμάτευση με βάση τη σύμβαση	Συνεργασία φοιτητών και εκπαιδευτών αντί προσκόλληση σε άκαμπτα προγράμματα σπουδών.
4	Ανταπόκριση στην αλλαγή, αντί για πιστή εφαρμογή του σχεδίου	Ανταπόκριση στην ανατροφοδότηση αντί να ακολουθούμε ένα σχέδιο.

Πίνακας 8.1. *Αντιστοίχιση των ευέλικτων αρχών στο περιβάλλον της τάξης (Stewart et al., 2009).*

Όπως φαίνεται και στον παραπάνω πίνακα, η πρώτη αξία της ευέλικτης προσέγγισης στην εκπαίδευση εστιάζεται στα μαθητοκεντρικά περιβάλλοντα παρουσιάζοντας αυτά ως την πιο αποτελεσματική μέθοδο

μάθησης. Στο ευέλικτο σχολείο η μέθοδος εκπαίδευσης που βασίζεται σε διαλέξεις έχει ξεπεραστεί. Οι μαθητές συμμετέχουν ενεργά στη μαθησιακή διαδικασία μέσω δραστηριοτήτων και ομαδικές συναντήσεις που στοχεύουν στην ενίσχυση της κατανόησης των εννοιών και στην εξερεύνηση αυτών.

Όπως και στην περίπτωση της ανάπτυξης λογισμικού έτσι και στην εκπαίδευση, η δεύτερη αξία στην εκπαίδευση ευνοεί την εργασία μέσω έργων από την αρχή της διδασκαλίας, χωρίς να περιμένει το τέλος ενός μαθήματος. Οι μαθητές εργάζονται με ένα επαναληπτικό τρόπο δίνοντας έμφαση στα παραδοτέα του έργου.

Η τρίτη αξία δίνει έμφαση στην ύπαρξη ενός περιβάλλοντος επικεντρωμένου σε αυτό που κάνουν οι μαθητές και στις παιδαγωγικές μεθόδους που μπορούν να διευκολύνουν τη μάθηση. Στην παραδοσιακή εκπαίδευση, η ύλη αποτελεί μια σύμβαση μεταξύ του μαθητή και του εκπαιδευτή. Εντός του ευέλικτου σχολείου, ο μαθητής και εκπαιδευτικός μπορούν να δημιουργήσουν μια πιο ευέλικτη και συνεργατική σχέση, παρόμοια με τον τρόπο με τον οποίο συνεργάζονται οι προγραμματιστές και οι πελάτες τους.

Τέλος, στην τέταρτη αξία, δίνεται έμφαση στην υιοθέτηση της αλλαγής, αφού μπορούν να υιοθετηθούν διαφορετικές εκπαιδευτικές μέθοδοι, σε περιπτώσεις όπου οι τρέχουσες μέθοδοι δεν παράγουν τα αναμενόμενα αποτελέσματα.

Αντίστοιχα στον Πίνακα 8.2 παρουσιάζεται η αντιστοίχιση των ευέλικτων αρχών στην εκπαίδευση.

	Agile Manifesto	**Agile Manifesto στην εκπαίδευση**
1	Επικέντρωση στον πελάτη	Υψηλή προτεραιότητα στην προετοιμασία του εκπαιδευόμενου ώστε να είναι αυτο-οργανούμενος, παρέχοντας συνεχώς στοιχεία του μαθήματος.
2	Οι μεταβαλλόμενες απαιτήσεις είναι καλοδεχούμενες	Ο εκπαιδευτικός και οι εκπαιδευόμενοι μπορούν να προσαρμοστούν στις αλλαγές ανά πάσα στιγμή ώστε να διευκολύνουν τη μάθηση και να αναπτύξουν αποτελεσματικότερα δεξιότητες.
3	Συχνή παράδοση λογισμικού σε τακτά χρονικά διαστήματα.	Εργασία με παραδοτέα από τους εκπαιδευομένους σε σύντομα χρονικά διαστήματα που επιτρέπουν την έγκαιρη ανατροφοδότηση.
4	Οι πελάτες και οι προγραμματιστές πρέπει να συνεργάζονται καθημερινά.	Επαναληπτική αλληλεπίδραση μεταξύ του εκπαιδευτή και των εκπαιδευομένων (ή των ομάδων εκπαιδευομένων).
5	Δημιουργήστε έργα γύρω από κινητοποιημένα άτομα και υποστηρίξτε τα σε ένα κατάλληλο περιβάλλον.	Παρέχετε στους μαθητές το κατάλληλο περιβάλλον και την υποστήριξη που απαιτείται για να είναι επιτυχημένοι.
6	Προτιμήστε τη συνομιλία πρόσωπο με πρόσωπο.	Επιτρέψτε, την άμεση πρόσωπο με πρόσωπο αλληλεπίδραση με εκπαιδευομένους ή τις ομάδες των εκπαιδευομένων.
7	Το λογισμικό που λειτουργεί είναι το πρωταρχικό μέτρο της προόδου.	Τα παραδοτέα (π.χ. μοντέλα, παραδοτέα έργου, παρουσιάσεις) είναι το πρωταρχικό μέτρο της προόδου των εκπαιδευομένων.
8	Οι χορηγοί, οι προγραμματιστές και οι χρήστες θα πρέπει να είναι σε θέση να διατηρούν σταθερό ρυθμό εργασίας.	Το συνεργατικό μαθησιακό περιβάλλον αποτελεί τη βάση για τη διδασκαλία των δεξιοτήτων που απαιτούνται για τη διά βίου μάθηση.
9	Η συνεχής προσοχή στην τεχνική αρτιότητα και τον καλό σχεδιασμό ενισχύει την ευελιξία.	Η συνεχής προσοχή στην τεχνική αρτιότητα και τον καλό σχεδιασμό ενισχύει τη μάθηση.
10	Η απλότητα είναι απαραίτητη.	Η κατανόηση του προβλήματος και η επίλυσή του απλά και ξεκάθαρα είναι απαραίτητη.
11	Οι καλύτερες αρχιτεκτονικές, απαιτήσεις και σχεδιασμός προκύπτουν από αυτο-οργανωμένες ομάδες.	Οι ομάδες πρέπει να αυτο-οργανώνονται, αλλά όλοι πρέπει να συμμετέχουν ισότιμα στην προσπάθεια.
12	Σε τακτά χρονικά διαστήματα, η ομάδα αξιολογεί πώς να γίνει πιο αποτελεσματική,	Σε τακτά χρονικά διαστήματα, οι εκπαιδευόμενοι και ο εκπαιδευτής σκέφτονται και προσφέρουν ανατροφοδότηση σχετικά με το πώς να είναι πιο αποτελεσματικοί και, στη

	Agile Manifesto	Agile Manifesto στην εκπαίδευση
	στη συνέχεια συντονίζει και προσαρμόζει α-νάλογα τη συμπεριφορά της.	συνέχεια, οι ενδιαφερόμενοι προσαρμόζονται ανάλογα για να είναι πιο αποτελεσματικοί.

Πίνακας 8.2: *Αντιστοίχιση των ευέλικτων αρχών στο περιβάλλον της τάξης (Stewart et al., 2009).*

8.1.1 Το Scrum στην τάξη

Ο τρόπος με τον οποίο ένας εκπαιδευτής προσεγγίζει το scrum ως εκπαιδευτικό εργαλείο (Boti et al., 2021):

- Ορίζουμε ένα έργο μικρής διάρκειας. Το έργο θα πρέπει να μπορεί να χωριστεί σε δύο ή τρία sprint. Ο εκπαιδευτής έχει τον αρχικό κατάλογο απαιτήσεων αλλά οι εκπαιδευόμενοι μπορούν να εμπλουτίσουν το αρχικό backlog μέσω των δικών τους εμπειριών.
- Δημιουργούνται ομάδες μαθητών Scrum και τα μέλη της ομάδας αποφασίζουν ποιος θα αναλάβει τον κάθε ρόλο. Τον ρόλο του ιδιοκτήτη προϊόντος μπορεί να τον αναλάβει ο εκπαιδευτής. Αυτές οι ομάδες έχουν υψηλό βαθμό αυτονομίας.
- Σε κάθε επανάληψη η ομάδα θα πρέπει να αποφασίζει σε συνεργασία με τον product owner για το ποιες απαιτήσεις του product backlog θα συμπεριληφθούν στο sprint.
- Για την ευκολότερη διαχείριση του έργου προτείνεται η χρήση ενός συνεργατικού εργαλείου (π.χ. Trello, basecamp). Για παράδειγμα το Trello είναι ένα Agile-style online εργαλείο διαχείρισης και προγραμματισμού εργασιών. Μπορείτε να βρείτε περισσότερες πληροφορίες και να κάνετε δωρεάν εγγραφή στην ιστοσελίδα του https://trello.com.
- Κατά την έναρξη του έργου θα πρέπει να γίνει αξιολόγηση των απαιτήσεων ώστε να υπολογιστεί η απαιτούμενη προσπάθεια (effort) για κάθε απαίτηση με χρήση της μεθόδου planning poker – (http://www.planningpoker.com/) ή άλλου αντίστοιχου εργαλείου.
- Στη συνέχεια θα πρέπει να γίνει ιεράρχηση, ώστε να αποφασιστεί ποιες απαιτήσεις θα συμπεριληφθούν σε κάθε sprint. Για την ιεράρχηση μπορεί να χρησιμοποιηθεί η μέθοδος priority poker (http://www.uxforthemasses.com/priority-poker/) που αποτελεί μια παραλλαγή της planning poker, όπου οι συμμετέχοντες αντί να επιλέγουν εκτιμήσεις προσπάθειας για κάθε απαίτηση, επιλέγουν προτεραιότητες. Η μέθοδος είναι δυνατό να εκτελεστεί και μέσω του http://www.planningpoker.com/ χρησιμοποιώντας τις κάρτες «T-Shirt Sizes» με την παραδοχή ότι το XXS αντιστοιχεί στην μικρότερη προτεραιότητα και το XXL στη μεγαλύτερη.
- Οι εκπαιδευόμενοι συμμετέχουν καθημερινά σε scrum meeting, ενώ στο τέλος κάθε sprint γίνεται ανασκόπηση με σκοπό την εξαγωγή συμπερασμάτων.

Τα παραδοτέα που μπορούν να ζητηθούν από τους εκπαιδευόμενους είναι:

- **Υπολογισμός της απαιτούμενης προσπάθειας** ανά απαίτηση τεκμηριώνοντας αναλυτικά τον τρόπο υπολογισμού της προσπάθειας.
- Υπολογισμός **των προτεραιοτήτων των απαιτήσεων** τεκμηριώνοντας αναλυτικά τον τρόπο υπολογισμού της προτεραιότητας.
- Το **product backlog,** όπως αυτό διαμορφώθηκε σε κάθε επανάληψη, σε μορφή φύλλου εργασίας.
- Για καθένα από τα user stories του product backlog τα **κριτήρια αποδοχής** (acceptance criteria).
- **Οι αναθέσεις των αρμοδιοτήτων** στα μέλη της ομάδας σε κάθε επανάληψη (sprint).
- Ο **χρόνος υλοποίησης** ανά απαίτηση/ανά επανάληψη (sprint) σε σύγκριση με αυτά που είχαν προϋπολογισθεί, καθώς και το διάγραμμα κατανάλωσης προσπάθειας (burnt down chart).
- Ο υπολογισμός της ταχύτητας παράδοσης (velocity) για κάθε επανάληψη.

8.1.2 Η προσέγγιση eduScrum

Το eduScrum (Wijnands, W. 2020) είναι ένα πλαίσιο εντός του οποίου οι εκπαιδευτές και οι μαθητές αντιμετωπίζουν σύνθετα προβλήματα και επιδιώκουν μαθησιακούς στόχους της υψηλότερης δυνατής αξίας με παραγωγικό και δημιουργικό τρόπο. Το eduScrum είναι μια ελαφριά διεργασία και είναι εύκολο στην κατανόηση γεγονός που καθιστά εύκολη την εφαρμογή του. Από μια άλλη οπτική γωνία η κατανόηση του eduScrum απαιτεί εμπειρία, γιατί δεν προσδιορίζει πως θα εκτελεστούν οι εκπαιδευτικές διεργασίες αφού το eduScrum δεν είναι μια διαδικασία ή τεχνική για την επίβλεψη μαθητών.

Στο eduScrum, οι μαθητές έχουν την ελευθερία να αποφασίζουν, κάτι που απαιτεί εκπαιδευτές με ευέλικτη νοοτροπία. Η έμφαση δίνεται στην οργάνωση και εκτέλεση της εργασίας τους εντός του καθορισμένου χρονικού πλαισίου και εντός καθορισμένων μαθησιακών στόχων. Επιπλέον, το eduScrum εστιάζει στη συνεχή βελτίωση μέσω των συνεχών ανασκοπήσεων. Αυτό παρέχει μια εικόνα για την αποτελεσματικότητα του σχεδιασμού και την επιλεγμένη προσέγγιση των μαθητών, ώστε να μπορούν να βελτιωθούν.

Το πλαίσιο eduScrum, όπως η μέθοδος Scrum, ορίζει ομάδες και τους σχετικούς ρόλους, τελετές, τεχνουργήματα (artifacts) και κανόνες.

Η ομάδα eduScrum αποτελείται από τον ιδιοκτήτη του προϊόντος, τον αρχηγό της ομάδας (πλοίαρχο) eduScrum και την ομάδα. Ο εκπαιδευτικός εκπληρώνει δύο ρόλους, αυτόν του ιδιοκτήτη προϊόντος, και αυτόν του αρχηγού της ομάδας και αποφασίζει για τους μαθησιακούς στόχους καθώς και για τον τρόπο αξιολόγησης. Ο δάσκαλος μοιράζεται το ρόλο του αρχηγού της ομάδας eduScrum μαζί με έναν μαθητή από κάθε ομάδα, με σκοπό να μεταφέρει την υπευθυνότητα αυτή μόλις ο ρόλος γίνει κατανοητός στο μαθητή. Οι ομάδες εκπαιδευομένων αποτελούνται από τέσσερις έως έξι μαθητές, είναι αυτοοργανωμένες και έχουν ποικιλία δεξιοτήτων. Οι ομάδες συγκροτούνται με βάση τις δεξιότητες των μαθητών.

Οι ομάδες μαθητών αναπτύσσουν επαναληπτικά και επαυξητικά τα ζητούμενα του έργου ενώ προσπαθούμε να μεγιστοποιήσουμε τις ευκαιρίες για ανασκόπηση / ανατροφοδότηση και προσαρμογή. Οι σταδιακές παραδόσεις ενός προϊόντος εκμάθησης διασφαλίζουν ότι ένα δυνητικά καλό αποτέλεσμα σε σχέση με τον μαθησιακό στόχο είναι πάντα εφικτό.

Συνοψίζοντας ο εκπαιδευτικός:

- καθορίζει τι θα μάθουν και για ποιο λόγο οι εκπαιδευόμενοι. Ο εκπαιδευτικός εκ φύσεως είναι υπεύθυνος για μετρήσιμα εκπαιδευτικά αποτελέσματα αφού θα πρέπει να διασφαλίσει ότι οι διάφοροι ενδιαφερόμενοι είναι ικανοποιημένοι με τα αποτελέσματα: μαθητές, γονείς, διοικητικό συμβούλιο, επιθεώρηση της εκπαίδευσης. Αυτός είναι ο λόγος για τον οποίο η ευθύνη για τι και γιατί θα διδαχθεί, πρέπει να ανήκει από κοινού μεταξύ του εκπαιδευτικού και των εκπαιδευομένων. Για την παρακολούθηση της ποιότητας των μαθησιακών αποτελεσμάτων, ο εκπαιδευτικός καθορίζει τα κριτήρια εορτασμού (celebration criteria) σε αντιστοιχία με τα κριτήρια αποδοχής των ιστοριών χρηστών. Τα κριτήρια εορτασμού ορίζονται στην έναρξη της εκπαιδευτικής διαδικασίας και μπορεί να περιλαμβάνουν την ελάχιστη βαθμολογία για τις εξετάσεις, τα ορόσημα – ημερομηνίες παράδοσης εργασιών, τις μορφές παρουσίασης, το μέγεθός των εργασιών κ.λπ.
- παρακολουθεί και βελτιώνει την ποιότητα των μαθησιακών αποτελεσμάτων,
- ελέγχει και αξιολογεί τα μαθησιακά αποτελέσματα και παρακολουθεί την προσωπική ανάπτυξη του κάθε εκπαιδευόμενου και
- έχει διαφορετικούς ρόλους.

Επίσης, ο εκπαιδευτικός είναι υπεύθυνος για τη διάδοση της φιλοσοφίας eduScrum. Φροντίζει ώστε το eduScrum να γίνεται κατανοητό και να εκτελείται σωστά και ως εκ τούτου επικεντρώνεται στον τρόπο εργασίας και συνεργασίας όλων των ομάδων σε μια τάξη. Πιο συγκεκριμένα:

- Εξηγεί τι είναι το eduScrum, τη σημασία του και πώς λειτουργεί.

- Εξασφαλίζει την καλή σύνθεση της ομάδας με βάση τα χαρακτηριστικά και τις δεξιότητες του κάθε εκπαιδευομένου.
- Διασφαλίζει ότι η διεργασία eduScrum ακολουθείται από τις ομάδες και ότι οι κανόνες του eduScrum ακολουθούνται.
- Εάν είναι απαραίτητο, αναλαμβάνει την υπευθυνότητα της διεργασίας, δίνοντας επιπλέον εξηγήσεις, κάνοντας επιδείξεις, και σχόλια.
- Δημιουργεί ένα θετικό μαθησιακό περιβάλλον ενθαρρύνοντας τη συμμετοχή.
- Προστατεύει την ομάδα από εξωτερικά προβλήματα.
- Ενθαρρύνει την ομάδα να αντιμετωπίζει δυσκολίες γρήγορα και ανεξάρτητα.

Αντίστοιχα, η ομάδα εκπαιδευομένων αποτελείται από μαθητές που εργάζονται από κοινού ώστε να επιτύχουν τους καθορισμένους μαθησιακούς στόχους στο τέλος του sprint, σύμφωνα με τα κριτήρια εορτασμού. Τα μέλη της ομάδας είναι από κοινού υπεύθυνα για την εκπλήρωση των κριτηρίων εορτασμού. Οι ομάδες εκπαιδευομένων έχουν τα ακόλουθα χαρακτηριστικά:

- Είναι αυτοοργανωμένες. Κανείς (ούτε καν ο εκπαιδευτικός) δεν μπορεί να υπαγορεύσει στην ομάδα πώς να επιτύχει τους μαθησιακούς στόχους.
- Οι ομάδες είναι διεπιστημονικές, δηλαδή υπάρχουν μέλη με όλες τις απαραίτητες δεξιότητες ώστε να είναι σε θέση να επιτύχουν τους μαθησιακούς στόχους ως ομάδα και ταυτόχρονα να είναι σε θέση να αναπτυχθούν προσωπικά.
- Τα μέλη της ομάδας μπορεί να έχουν συγκεκριμένες δεξιότητες ή τομείς εστίασης, αλλά η ευθύνη ανήκει στην ομάδα συνολικά.
- Τα μέλη της ομάδας μπορούν να αποφασίσουν ατομικά εάν θέλουν να χρησιμοποιήσουν τις ικανότητες και δεξιότητές τους ή να αναπτύξουν νέες.
- Η ομάδα παρακολουθεί την πρόοδο της ποιότητα που επιτεύχθηκε, με τη χρήση των κριτηρίων εορτασμού και τις απαιτήσεις της εργασίας.

Το βέλτιστο μέγεθος της ομάδας πρέπει να είναι αρκετά μικρό ώστε η ομάδα να είναι λειτουργική και ταυτόχρονα αρκετά μεγάλο ώστε η ομάδα να μπορεί να υλοποιήσει μια σημαντική εργασία. Ο βασικός κανόνας είναι η δημιουργία ομάδων με τέσσερα ή πέντε μέλη. Λιγότερο από τρία μέλη σημαίνει ότι η αλληλεπίδραση μειώνεται και οι αναγκαίες δεξιότητες δεν υπάρχουν, επαρκώς. Όταν υπάρχουν περισσότερα από έξι μέλη τότε απαιτείται υπερβολικός συντονισμός. Οι μεγάλες ομάδες δημιουργούν υπερβολική πολυπλοκότητα ώστε να ελεγχθούν από μια εμπειρική διαδικασία.

Μέσα στην ομάδα, ένα από τα μέλη παίζει τον ρόλο του αρχηγού της ομάδας (team captain). Ο αρχηγός της ομάδας δεν είναι προϊστάμενος αλλά ένα εργαζόμενο μέλος της ομάδας, ένα ισότιμο μέλος της ομάδας. Αυτός ή αυτή διασφαλίζουν ότι η ομάδα μπορεί να αποδώσει βέλτιστα.

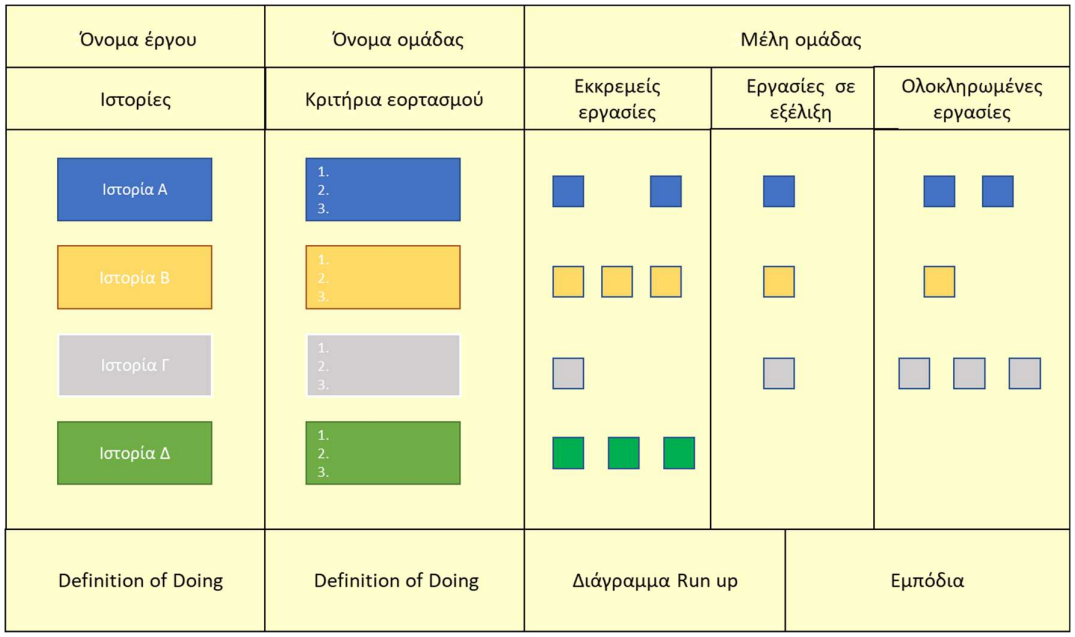
Στο eduScrum, ο αρχηγός της ομάδας είναι ένας ρόλος πιο περιορισμένος από τον ρόλο του Scrum Master στο Scrum. Αυτό συμβαίνει επειδή διαφορετικά καθήκοντα και ευθύνες που θα αποτελούσαν μέρος του ρόλου Scrum Master ανατίθενται στον εκπαιδευτικό. Ο λόγος είναι διότι οι αρχηγοί της ομάδας είναι εκπαιδευόμενοι-μαθητές, και συνεπώς έχουν μικρότερη εμπειρία. Καθώς το έργο προχωρά, μπορεί να αναλάβουν περισσότερες ευθύνες από τον εκπαιδευτικό, πράγμα που σημαίνει ότι οι συνολικές ευθύνες του εκπαιδευτικού μειώνονται σταδιακά. Ο ρόλος του αρχηγού της ομάδας ανατίθεται κατά τη διάρκεια του σχηματισμού της ομάδας στην αρχή του πρώτου σπριντ. Ο αρχηγός της ομάδας ορίζεται από τον εκπαιδευτικό ή επιλέγεται από τους εκπαιδευόμενους. Στη συνέχεια, ανάλογα με τη διαδικασία που χρησιμοποιείται για τη δημιουργία της ομάδας, οι αρχηγοί της ομάδας επιλέγουν τα μέλη της ομάδας τους με βάση συμπληρωματικές ικανότητες-δεξιότητες.

Μέσα στην ομάδα, ο αρχηγός της ομάδας είναι υπεύθυνος για το "Flap". Το "Flap" είναι ο οπτικός πίνακας που κάνει ορατή την εργασία και τις δεσμεύσεις της ομάδας. Ο αρχηγός της ομάδας διασφαλίζει ότι το "Flap" είναι διαθέσιμο όταν χρειάζεται και ότι έχει ενημερωθεί. Ωστόσο, η τελική εργασία είναι ευθύνη ολόκληρης της ομάδας (Wijnands & Stolze, 2020).

Το "Flap" είναι μια χρονολογική αναπαράσταση του έργου κατά τη διάρκεια ενός sprint. Η επάνω γραμμή του "Flap" καθορίζει το έργο, σε ποια ομάδα ανήκει το "Flap" και από ποια μέλη αποτελείται. Οι εργασίες του "Flap" μπορεί να είναι "εκκρεμείς εργασίες", "σε εξέλιξη" καθώς και "ολοκληρωμένες". Το "Flap" πρέπει να ενημερώνεται τακτικά, ώστε πάντα να δείχνει τις τρέχουσες εργασίες της ομάδας της ομάδας.

Επιπλέον των παραπάνω πεδίων, το "Flap" περιέχει και τα ακόλουθα:

- Ιστορίες Χρηστών.
- Κριτήρια Εορτασμού.
- Ορισμός ολοκλήρωσης δραστηριοτήτων - Definition of Doing. Είναι ο ορισμός του πότε μια εργασία έχει ολοκληρωθεί.
- Ορισμός της επικοινωνίας - Definition of communication. Το πως η ομάδα επικοινωνεί.
- Ορισμός της διασκέδασης - Definition of Fun. Η διασκέδαση/ευχαρίστηση είναι ένα σημαντικό κίνητρο για τους εκπαιδευόμενους και ως εκ τούτου είναι απαραίτητη για την επίτευξη καλύτερων μαθησιακών αποτελεσμάτων. Συνεπώς, οι εκπαιδευόμενοι θα πρέπει επίσης να υποδείξουν τι χρειάζονται για να εξασφαλίσουν ευχάριστη εργασία.
- Το διάγραμμα Run-Up απεικονίζει τη μαθησιακή πρόοδο κάθε ομάδας, δείχνει στην ομάδα την πρόοδο και αν πρέπει να δουλέψει πιο γρήγορα ή όχι. Κάθε εργασία έχει έναν αριθμό σημείων σύμφωνα με την εκτίμηση προσπάθειας που γίνεται χρησιμοποιώντας μια μέθοδο όπως το Planning Poker.
- Τα εμπόδια της ομάδας.



Εικόνα 8.3: *Το "flap" στο eduScrum*

8.2 Η ευέλικτη προσέγγιση στο δημόσιο τομέα

Στο μέρος αυτό θα εξετάσουμε τις ιδιαίτερες συνθήκες που ισχύουν στο δημόσιο τομέα ως προς την υιοθέτηση ευέλικτων μεθοδολογιών για την ανάπτυξη και διαχείριση έργων πληροφορικής και θα διερευνήσουμε τους παράγοντες που την επηρεάζουν. Θα αναφερθούμε στα πεδία στα οποία διαφοροποιούνται οι δημόσιοι οργανισμοί από τους ιδιωτικούς καθώς και τον τρόπο που οι διαφορές αυτές επηρεάζουν την υιοθέτηση ευέλικτων μεθόδων. Θα αναγνωριστούν και θα ομαδοποιηθούν οι παράγοντες που θεωρούνται σημαντικοί (και, πιθανώς, κρίσιμοι) για την επιτυχία της υιοθέτησης (Βουτσάς, 2015).

8.2.1 Οι διαφορές δημόσιου και ιδιωτικού τομέα στη διαχείριση των ψηφιακών υπηρεσιών

Οι δημόσιοι οργανισμοί παρότι έχουν παρόμοιες ανάγκες με τους ιδιωτικούς φορείς, από τη φύση τους διαφέρουν σημαντικά στους σκοπούς, το νομικό καθεστώς, την οργάνωση, τις κυρίαρχες αξίες και σε μία σειρά σημείων που επηρεάζουν τη διαχείριση των ψηφιακών υπηρεσιών σε αυτούς.

Ο δημόσιος τομέας έχει πολλαπλούς στόχους, συχνά άυλους και όχι εύκολα μετρήσιμους ενώ εξυπηρετεί μία πλειάδα ενδιαφερομένων, συχνά με αντιτιθέμενα συμφέροντα. Οι κυβερνητικές προτεραιότητες και κατευθύνσεις καθορίζουν και ελέγχουν την επίτευξη των στόχων αυτών. Από την άλλη, ο ιδιωτικός τομέας καθοδηγείται από την επιδίωξη κέρδους και μετράει την αποτελεσματικότητά του κυρίως στη βάση της οικονομικής αποδοτικότητας. Αυτό κάνει τον δημόσιο τομέα λιγότερο ευάλωτο στους οικονομικούς κύκλους αλλά περισσότερο εξαρτημένο από τις πολιτικές αλλαγές. Η αυξημένη πολιτική παρέμβαση δημιουργεί περιοδική αποσταθεροποίηση των δημόσιων οργανισμών, με τις αλλαγές διοικήσεων ή κυβερνήσεων, καθώς σημαντικές ανακατατάξεις προτεραιοτήτων μπορούν να επέλθουν από τις αλλαγές αυτές που συχνά οδηγούν σε διοικητικές ασυνέχειες.

Λόγω της διαφοράς στους βασικούς σκοπούς, οι δημόσιοι οργανισμοί έχουν λιγότερα κίνητρα και περιορισμένη πίεση για βελτίωση της παραγωγικότητας και της οικονομικής απόδοσης, αλλά, από την άλλη, έχουν περισσότερους νομικούς και διοικητικούς περιορισμούς, αυστηρότερες και λιγότερο ελαστικές ιεραρχικές δομές και περισσότερη γραφειοκρατία. Η περισσότερη γραφειοκρατία καθιστά τη λήψη αποφάσεων και την πραγματοποίηση επενδύσεων δυσχερέστερη με την ύπαρξη περισσότερων περιορισμών πχ. στη διαδικασία προμηθειών προϊόντων ή υπηρεσιών. Σημαντική διαφοροποίηση φέρνει και στη δυνατότητα καθορισμού αρμοδιοτήτων, ευθύνης, εξουσιών και ρόλων στο πλαίσιο της οργάνωσης μίας δραστηριότητας. Ακόμα, λόγω των περιορισμένων δυνατοτήτων στη διαμόρφωση των αμοιβών, τα κίνητρα για ένα ανθρώπινο δυναμικό υψηλής κατάρτισης είναι λιγότερα, με αποτέλεσμα οι δημόσιοι οργανισμοί να ανταποκρίνονται στην έλλειψη ικανοτήτων, ιδίως στους τεχνικούς τομείς, με την εξωτερική ανάθεση έργων και υπηρεσιών.

Οι δημόσιοι οργανισμοί είναι λιγότερο αναμενόμενο να αναλαμβάνουν κινδύνους (risk) για την προώθηση της καινοτομίας και της αλλαγής, ενώ οι ιδιωτικοί περισσότερο, στη βάση της επιδίωξης ανταγωνιστικού πλεονεκτήματος που φέρνει η καινοτομία. Συνεπώς, η οργανωσιακή κουλτούρα των ιδιωτικών οργανισμών είναι πιο δεκτική στην καινοτομία και την αλλαγή, ενώ των δημόσιων προσανατολισμένη στις αξίες που εξασφαλίζουν συμμόρφωση με την τήρηση διαδικασιών.

Ο δημόσιος τομέας έχει όμως και ένα σημαντικό συγκριτικό πλεονέκτημα, τουλάχιστον σε ό,τι αφορά τις ψηφιακές υπηρεσίες: Τη δυνατότητα να διαμοιράζεται IT πόρους, εφαρμογές, δεδομένα και τεχνικές ανάμεσα στους δημόσιους οργανισμούς, ενώ η δυνατότητα αυτή είναι σαφώς περιορισμένη στον ιδιωτικό τομέα, λόγω των ανταγωνιστικών σχέσεων. Ο Πίνακας 8.3 συνοψίζει τις διαφορές ανάμεσα στον ιδιωτικό και το δημόσιο τομέα, στα σημεία που επηρεάζουν τη διαχείριση των ψηφιακών υπηρεσιών (IT Governance).

Σε κάθε περίπτωση, η ανάπτυξη ενός δημόσιου έργου πληροφορικής είναι μία σύνθετη (και συχνά περίπλοκη) διαδικασία, με πολλούς συμμετέχοντες, με το τελικό αποτέλεσμα να επηρεάζεται από ένα πλήθος παραγόντων.

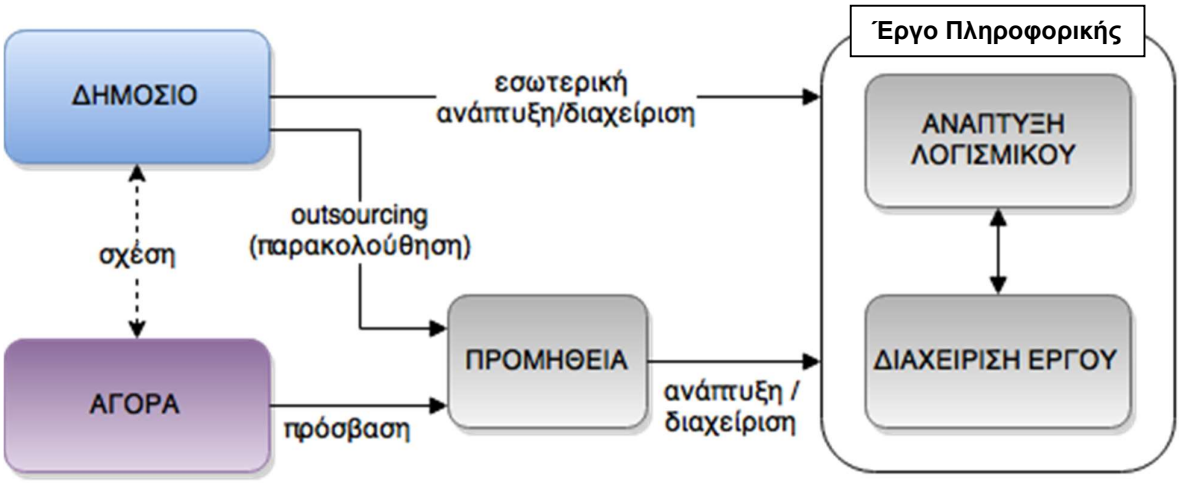
Χαρακτηριστικό	Τομέας	
	Δημόσιος	**Ιδιωτικός**
Στόχοι	Πολλαπλοί και άυλοι	Συγκεκριμένοι και απτοί
Προϊόν	Παροχή δημοσίων αγαθών και υπηρεσιών	Κέρδος
Μέτρο αποτελεσματικότητας	Πολιτική αποτελεσματικότητα και επίτευξη πολιτικών στόχων	Κερδοφορία και οικονομική αποδοτικότητα
Ενδιαφερόμενοι (stakeholders)	Περισσότεροι και με αντιτιθέμενα συμφέροντα	Λιγότεροι, με περισσότερο κοινά συμφέροντα
Περιβάλλον	Λιγότερα κίνητρα για παραγωγικότητα	Περισσότερα κίνητρα
	Περισσότεροι νομικοί και διοικητικοί περιορισμοί - γραφειοκρατία	Λιγότερη γραφειοκρατία
	Πολιτικές επιρροές	Επιρροές από την αγορά
	Κουλτούρα περισσότερο προσανατολισμένη στην τυποποίηση και τήρηση διαδικασιών	Κουλτούρα περισσότερο προσανατολισμένη στην καινοτομία και την αλλαγή
Πολιτική IT	Διαμοιρασμός IT πόρων, εφαρμογών, δεδομένων και τεχνικής βοήθειας	Οι πόροι IT αντιμετωπίζονται ως ιδιωτικό μέσο για την επίτευξη ανταγωνιστικού πλεονεκτήματος.

Πίνακας 8.3: *Κατάταξη σφαλμάτων ανάλογα με τη σοβαρότητά τους*

8.2.2 Η ανάπτυξη έργων πληροφορικής στο δημόσιο τομέα.

Η διαδικασία ανάπτυξης ενός έργου Πληροφορικής στο δημόσιο τομέα, από τη σκοπιά που μας ενδιαφέρει, θα μπορούσε να αναπαρασταθεί με το σχήμα της Εικόνα 8.4. Οι **δημόσιοι φορείς** σε ότι αφορά τα έργα Πληροφορικής, είτε αναπτύσσουν **εσωτερικά** ένα έργο (in-house development) είτε το **αναθέτουν** σε εξωτερικούς συνεργάτες (outsourcing). Στη δεύτερη περίπτωση, που είναι και η συνηθέστερη, μπορούν να αναθέσουν την **ανάπτυξη** ή τη **διαχείριση** ή και τα δύο στον ίδιο ή σε διαφορετικούς προμηθευτές. Ακόμα και στην περίπτωση της εσωτερικής ανάπτυξης ή/και διαχείρισης του έργου, θα χρειαστεί πιθανότητα η εξωτερική ανάθεση κάποιων εργασιών.

Αυτό δημιουργεί την ανάγκη για την εφαρμογή μίας διαδικασίας **προμήθειας** και για την διαχείριση και παρακολούθηση της προμήθειας αυτής. Το πώς θα διαμορφωθεί η προμήθεια αυτή θα ορίσει το πώς θα αναπτυχθεί το έργο και πώς θα γίνει η διαχείρισή του. Οι φορείς της **αγοράς**, μέσα από την **πρόσβαση** που έχουν στη διαδικασία προμηθειών, θα εργαστούν από κοινού με τον δημόσιο οργανισμό, στο πλαίσιο των όρων του συμβολαίου που θα συναφθεί, για την ανάπτυξη του έργου.



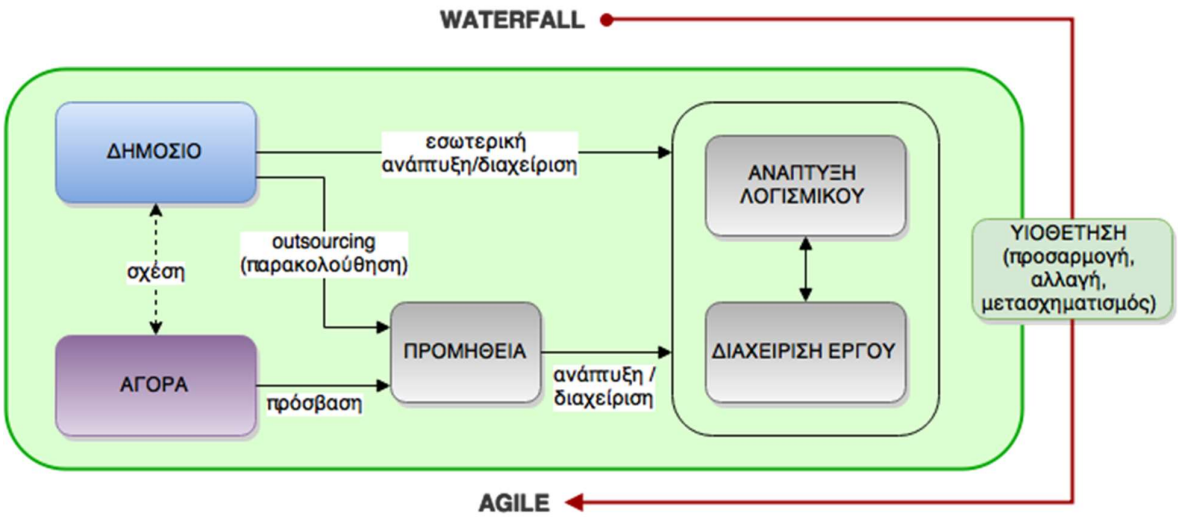
Εικόνα 8.4: *Η διαδικασία ανάπτυξης ενός έργου πληροφορικής στο δημόσιο τομέα*

Το ανωτέρω σχήμα έχει σκοπό να οριοθετήσει τα ζητήματα που σχετίζονται με την υιοθέτηση ευέλικτων μεθόδων από δημόσιους οργανισμούς στο πλαίσιο της συγκεκριμένης εργασίας και να καταδείξει ορισμένα βασικά πεδία όπου αναμένεται να παρουσιαστούν **προβλήματα** / **προκλήσεις** για τη μετάβαση αυτή, επομένως πεδία που χρήζουν διερεύνησης.

8.2.3 Από τις παραδοσιακές σε ευέλικτες μεθόδους: Η διαδικασία υιοθέτησης

Η υιοθέτηση ευέλικτων μεθόδων είναι μία διαδικασία που αφορά κυρίως στην ανάπτυξη λογισμικού, αλλά θέτει προϋποθέσεις σε όλες τις δραστηριότητες πριν από αυτή, προκειμένου να είναι αποτελεσματική. Διατρέχοντας προς τα πίσω στην Εικόνα 8.5, διαπιστώνουμε ότι η αποτελεσματικότητα στην ευέλικτη ανάπτυξη εξαρτάται, περισσότερο ή λιγότερο, τόσο από όλες τις διαδικασίες που οδηγούν σε αυτή όσο και από τα χαρακτηριστικά των οργανισμών που τις εκτελούν. Τα στοιχεία του πεδίου (στο παρόν πλαίσιο, οι κόμβοι και τα βέλη του διαγράμματος) πρέπει να είναι συμβατά με τις απαιτήσεις των ευέλικτων μεθόδων για να αυξήσουν τις πιθανότητες της επιτυχημένης μετάβασης σε αυτές (Stankovic et al., 2013) Ανάλογα με το βαθμό ετοιμότητας των οργανισμών και ένα πλήθος παραγόντων του περιβάλλοντος, η μετάβαση σε ένα ευέλικτο μοντέλο μπορεί να προϋποθέτει μία απλή προσαρμογή των υφιστάμενων στοιχείων, μία βαθύτερη αλλαγή ή ακόμα κι έναν ριζικό μετασχηματισμό.

Τόσο το δημόσιο ως πελάτης, όσο και η αγορά, ως προμηθευτής, θα πρέπει να έχουν κάποιο βαθμό ετοιμότητας (ή ωριμότητας) για την υιοθέτηση ευέλικτων μεθόδων. Από την άλλη, τρόπος για την απόκτηση ευελιξίας είναι η ίδια η εφαρμογή μεθόδων με την προσδοκία ότι, αν βρουν πρόσφορο έδαφος, θα βοηθήσουν σταδιακά σε βαθύτερες αλλαγές.



Εικόνα 8.5: *Η υιοθέτηση ευέλικτης προσέγγισης απαιτεί αλλαγές σε όλη την έκταση της διαδικασίας ανάπτυξης και διαχείρισης έργων IT*

8.2.4 Παράγοντες που επηρεάζουν την υιοθέτηση των ευέλικτων πρακτικών στο δημόσιο τομέα

Παρά την ύπαρξη αρκετών πηγών που αναφέρονται σε παράγοντες επιτυχούς υιοθέτησης μίας περισσότερο ευέλικτης προσέγγισης σε κάθε είδους οργανισμούς και επιχειρήσεις, δεν υπάρχουν ανάλογες που να επικεντρώνονται ειδικά στους δημόσιους οργανισμούς και να αφορούν την ευέλικτη ανάπτυξη. Οι περισσότερες πηγές σχετικά με τους δημόσιους οργανισμούς περιορίζονται σε εκθέσεις κυβερνητικών οργανισμών και σε αναλύσεις συμβουλευτικών εταιρειών ή ιδρυμάτων (National Audit Office, 2012; U.S. Government Accountability Office, 2012; UK National Audit Office, 2012). Από όλες τις παραπάνω πηγές, ως σημαντικοί παράγοντες για τη μετάβαση σε ευέλικτες μεθόδους στο δημόσιο τομέα έχουν θεωρηθεί:

Α. Η **οργανωσιακή κουλτούρα** θεωρείται από τους σημαντικότερους, αν όχι ο σημαντικότερος παράγοντας για την επιτυχή μετάβαση στην ευελιξία. Η συνήθεια μίας δομημένης, γραμμικής διαδικασίας που χαρακτηρίζεται από αργούς ρυθμούς στη λήψη αποφάσεων και τον έλεγχο σε στάδια μπορεί να εντυπωθεί βαθιά στην κουλτούρα των οργανισμών. Η κουλτούρα που αναπτύσσεται σε ιεραρχικές δομές συχνά είναι εσωστρεφής και προσανατολισμένη στη σταθερότητα, όχι στην αλλαγή. Δίνοντας μεγάλη σημασία στην τήρηση διαδικασιών και στον ιεραρχικό έλεγχο, υπονομεύει το δημιουργικό πνεύμα που χρειάζονται οι ευέλικτες μέθοδοι. Η κουλτούρα στις οργανώσεις είναι ένα χαρακτηριστικό που δεν αλλάζει εύκολα, διότι είναι εδραιωμένη στη συνείδηση των εργαζομένων αλλά και των πελατών. Στην τελευταία έρευνα "State of Agile" αναφέρεται ότι στους τρεις κυριότερους λόγους που εμποδίζουν την υιοθέτηση ευέλικτων πρακτικών είναι η ασυνέπεια των διεργασιών και πρακτικών μεταξύ των ομάδων (46% των απαντήσεων), η αδυναμία αλλαγής της οργανωσιακής κουλτούρας (43%) και η αντίσταση της οργάνωσης στην αλλαγή (42%) (Digital.ai, 2021).

Προβλήματα υιοθέτησης ευέλικτων μεθόδων	
Ασυνεπείς διεργασίες και πρακτικές μεταξύ των ομάδων	46%
Η οργανωτική κουλτούρα σε αντίθεση με τις ευέλικτες αξίες	43%
Αντίσταση οργάνωσης στην αλλαγή	42%
Έλλειψη δεξιοτήτων/εμπειρίας με ευέλικτες μεθόδους	42%
Μη δέσμευση της ηγεσίας	41%
Ανεπαρκής διοικητική και οικονομική υποστήριξη	40%

Προβλήματα υιοθέτησης ευέλικτων μεθόδων	
Ανεπαρκής κατάρτιση και εκπαίδευση	35%
Κυριαρχία των παραδοσιακών μεθόδων ανάπτυξης	35%
Έλλειψη ολοκληρωμένων εργαλείων, δεδομένων/μετρήσεων σχετικά με το έργο	30%
Απροθυμία των ατόμων να παραδεχτούν τα λάθη και να μάθουν από τις αποτυχίες παράδοσης	22%
Ελάχιστη συνεργασία μεταξύ των μελών της ομάδας και διάχυσης της γνώσης	17%
Κανονιστική συμμόρφωση με την ευέλικτη προσέγγιση	13%
Δεν ξέρω	7%
Άλλα	5%

Πίνακας 8.4: *Προβλήματα υιοθέτησης ευέλικτων μεθόδων*

Η **εισαγωγή μίας νέας μεθοδολογίας** σε έναν οργανισμό έχει τον χαρακτήρα της εισαγωγής καινοτομίας και, συνακόλουθα, όλη την προβληματική σχετικά με την εισαγωγή καινοτομιών στους οργανισμούς. Οι καινοτομίες, παρότι απαραίτητες για την πρόοδο, συναντούν σε μικρότερο ή μεγαλύτερο βαθμό δυσκολίες στην αφομοίωση, ανάλογα με το βαθμό που «απειλούν» κατεστημένες νοοτροπίες, συμφέροντα και συνήθειες. Προτάσσοντας την ενδυνάμωση ατόμων και ομάδων, το ομαδικό πνεύμα, τον σεβασμό στον εργαζόμενο και την αυτο-οργάνωση, οι ευέλικτες μέθοδοι προσφέρουν ένα πιο δημιουργικό, συνεργατικό και ελεύθερο περιβάλλον εργασίας που βοηθάει τόσο στην αποδοχή των εργαζομένων όσο και στη διάδοση της ευελιξίας. Έχει αναφερθεί ότι μετά από την εισαγωγή ευέλικτων τεχνικών διαχείρισης από ένα τμήμα (συνήθως το τμήμα IT), τα συνεργαζόμενα τμήματα αναζητούν από μόνα τους τρόπους εφαρμογής ευέλικτων τεχνικών για τις δικές τους εργασίες, έχοντας γνωρίσει τις ωφέλειές τους (National Audit Office, 2012).

Η **ανατρεπτική φύση των ευέλικτων μεθοδολογιών** που δεν μπορούν να αντιμετωπιστούν εργαλειακά, ότι θα κάνουν μία δουλειά και μετά θα ενεργοποιηθούν όταν τις χρειαστούμε πάλι, όπως ένα οποιαδήποτε εργαλείο. Η ευέλικτη προσέγγιση αποτελεί μία «φιλοσοφία» ανάπτυξης που πρέπει κάποιος να την ενστερνιστεί και όχι απλά μία τεχνική για να ακολουθήσει. Είναι τέτοια η φύση τους που οι προϋποθέσεις για την επιτυχή τους εφαρμογή οδηγούν σε ανατροπή της κατεστημένης ιεραρχίας αμφισβητώντας την άκαμπτη, καθοδηγητική ηγεσία. Αναθεωρούν τον τρόπο λήψης αποφάσεων δίνοντας έμφαση στις συλλογικές αποφάσεις της ομάδας. Αμφισβητούν την οργάνωση και τους ρόλους που έχουν οριστεί από τη διοίκηση με το να προωθούν την εναλλαγή ρόλων μεταξύ των μελών μίας ομάδας και την ανάπτυξη γνώσης για το αντικείμενο της εργασίας από όλα τα μέλη της. Επίσης, όπως έχουμε επανειλημμένα αναφέρει προωθούν την αυτο-οργάνωση των ομάδων. Προκρίνουν την άμεση, διαπροσωπική επικοινωνία παρά την ανταλλαγή αναφορών και εγγράφων. Προωθούν την πρωτοβουλία και τη διαρκή συνεργασία σε ανοιχτούς χώρους εργασίας και σε ομαδικά εργασιακά περιβάλλοντα. Είναι φανερό ότι τέτοιας έκτασης αλλαγές στις καθημερινές πρακτικές της εργασίας, δεν μπορούν να συμβούν χωρίς να επηρεάσουν την ευρύτερη λειτουργία ενός οργανισμού και δεν είναι εύκολο να εφαρμοστούν αν δεν υπάρχει η σχετική κουλτούρα σε αυτόν.

Ένα άλλο πεδίο όπου εντοπίζονται περισσότερα προβλήματα είναι αυτό της συμμόρφωσης με κάθε είδους **κανονιστικές ρυθμίσεις** που αφορούν στη διαδικασία έγκρισης, χρηματοδότησης και παρακολούθησης της πορείας των έργων, που απαιτούν εκτενή τεκμηρίωση και συγκεκριμένες φάσεις. Σε συνδυασμό με τις εσωτερικές διαδικασίες ελέγχου και εγκρίσεων στους δημόσιους οργανισμούς, μπορεί να δημιουργήσουν ένα αδιαπέραστο πλέγμα γραφειοκρατικών ρυθμίσεων, εχθρικό στην ευέλικτη προσέγγιση. Έχουμε δε αναφερθεί ήδη στα προβλήματα που δημιουργούνται από ανάλογους περιορισμούς στις διαδικασίες των προμηθειών.

Η κουλτούρα μίας ιεραρχικής και γραφειοκρατικής οργάνωσης μπορεί να καταστήσει τους υπαλλήλους λιγότερο αποφασιστικούς και πρόθυμους να αναλάβουν ευθύνες και πρωτοβουλίες και τη διοίκηση να είναι περισσότερο επιφυλακτική. Οι γραφειοκρατικές δομές ευνοούν την απόδοση ευθυνών και κατηγοριών (blame game) στο άτομο, ενώ η ευέλικτη προσέγγιση προτάσσει τις ευθύνες της ομάδας αλλά και όλων των συμμετεχόντων. Επίσης, η συμμετοχή σε ένα ευέλικτο έργο απαιτεί περισσότερο χρόνο και

αφοσίωση από το προσωπικό των δημόσιων φορέων, γεγονός που επηρεάζεται από τη δυσκολία συνδυασμού της συμμετοχής με τα συνήθη καθήκοντα της θέσης εργασίας, χωρίς συχνά να υπάρχει υποστήριξη από τη διοίκηση ή κάποιο κίνητρο γι' αυτό. Ακόμα, έχουν αναφερθεί δυσκολίες στην προσαρμογή ατόμων και ομάδων σε πρακτικές όπως η αυτο-οργάνωση και η συνεργατική ανάπτυξη.

8.2.5 Προϋποθέσεις επιτυχούς εφαρμογής

Οι ευέλικτες μέθοδοι, επειδή είναι ανθρωποκεντρικές και προσανατολισμένες στην αξία, προϋποθέτουν την ύπαρξη μίας σειράς χαρακτηριστικών από το περιβάλλον στο οποίο θα εφαρμοστούν. Ο Πίνακας 8.5 συνοψίζει τα στοιχεία που αναλύθηκαν στις προηγούμενες παραγράφους ως προς τις προϋποθέσεις που θέτουν στους δημόσιους οργανισμούς. Ως συνέπεια των παραπάνω, ορισμένα από τα χαρακτηριστικά που χρειάζεται να διαθέτουν οι δημόσιοι και οι ιδιωτικοί φορείς, αντίστοιχα, για την επιτυχημένη εφαρμογή μίας ευέλικτης μεθοδολογίας σε ένα δημόσιο έργο είναι:

- **Δημόσιες υπηρεσίες**: Γνώση των μεθοδολογιών, πρόσβαση στους πραγματικούς τελικούς χρήστες, διαθεσιμότητα στελεχών για συμμετοχή σε ευέλικτα έργα, ύπαρξη και διατήρηση τεχνικών ικανοτήτων, εξουσία και αρμοδιότητα για ταχείες αποφάσεις, στήριξη από την ανώτερη διοικητική ιεραρχία, στήριξη από την πολιτική ηγεσία, προσήλωση στην αξία για τον πολίτη κ.α.
- **Συνεργαζόμενες εταιρείες**: Να ενστερνίζονται τις αξίες της ευελιξίας, εμπειρία στην εφαρμογή των μεθόδων, τεχνική ικανότητα, επικοινωνιακή ικανότητα, προσήλωση στην αξία για τον πελάτη κ.α.

Επειδή σπάνια καλύπτονται όλες οι προϋποθέσεις (δηλ. οι φορείς - δημόσιοι και ιδιωτικοί - συχνά δεν έχουν κάποια από τα παραπάνω χαρακτηριστικά), θα πρέπει να γίνεται εκτίμηση του κατά πόσον αυτές ισχύουν σε ένα περιβάλλον. Αν κάποια πρακτική ή προσέγγιση δεν είναι εφικτή, λόγω των περιορισμών που υπάρχουν, η ενσωμάτωση στοιχείων από πιο παραδοσιακές μεθόδους μπορεί να βοηθήσει, αρκεί να μην ακυρώνονται τελείως τα πλεονεκτήματα της ευελιξίας.

Παράγοντες του περιβάλλοντος	**Προϋπόθεση**
1. Επιλογή εξωτερικής ανάθεσης	Ικανότητα συνεργατών Ικανότητα στο ρόλο του δημόσιου υπαλλήλου να αναλάβει το ρόλο του ιδιοκτήτη προϊόντος (Product Owner)
2. Η διαδικασία και το κανονιστικό πλαίσιο των προμηθειών	Πλαίσιο προμηθειών φιλικό στην ευέλικτη προσέγγιση
3. Σχέσεις δημοσίου-αγοράς	Κλίμα εμπιστοσύνης Δυνατότητα ευέλικτης συνεργασίας
4. Υποχρεώσεις συμμόρφωσης	Πλαίσιο κανονισμών έγκρισης και παρακολούθησης έργων φιλικό στην ευέλικτη προσέγγιση
Παράγοντες των οργανισμών	
1. Οργανωσιακή κουλτούρα	Υποστήριξη διοίκησης Αποδοχή των εννοιών της ευελιξίας από εργαζόμενους Προσανατολισμός στην παραγωγή αξίας Έμφαση στο αποτέλεσμα, όχι στις διεργασίες Συνεργατική ηγεσία
2. Συμμόρφωση με εσωτερικές διαδικασίες.	Συνεργατική κουλτούρα στις σχέσεις μεταξύ των τμημάτων. Δυνατότητα αλλαγής εσωτερικών διαδικασιών.
3. Ανθρώπινο δυναμικό και διοίκηση	Ικανότητες ανθρώπινου δυναμικού. Νοοτροπία θετική ως προς τις αλλαγές
4. Επιλογή εσωτερικής ανάπτυξης	Θετική εκτίμηση της ικανότητας του οργανισμού σε ευέλικτη ανάπτυξη ή / και διαχείριση έργων.

Πίνακας 8.5: *Προϋποθέσεις της ευέλικτης προσέγγισης για δημόσιους οργανισμούς*

Όλα τα παραπάνω συντείνουν πως η μετάβαση σε μία πραγματικά ευέλικτη προσέγγιση δεν είναι εύκολη υπόθεση και πως οι περισσότεροι οργανισμοί συναντούν δυσκολίες στην υιοθέτηση ευέλικτων μεθόδων.

8.2.6 Σύνοψη ως προς τους σημαντικούς παράγοντες της διαδικασίας υιοθέτησης.

Με βάση όλα τα προαναφερθέντα και μετά από μελέτη της σχετικής θεωρίας, θα μπορούσαμε να ομαδοποιήσουμε τους παράγοντες που επηρεάζουν την υιοθέτηση ευέλικτων μεθόδων ανάπτυξης σε τέσσερις κατηγορίες ως εξής:

- Παράγοντες που σχετίζονται με τους ανθρώπους. Οι άνθρωποι, οι ικανότητές τους στην ανάπτυξη λογισμικού και στη διαχείριση έργων, οι επιδιώξεις και προσδοκίες τους, οι τρόποι επικοινωνίας, οι διαπροσωπικές σχέσεις, οι συγκρούσεις συμφερόντων, ο τρόπος και η δυνατότητα στελέχωσης, η εκπαίδευση και κατάρτιση κ.α.
- Παράγοντες που σχετίζονται με την οργάνωση και τη διοίκηση. Στους παράγοντες αυτούς συγκαταλέγουμε την κουλτούρα της οργάνωσης, το περιβάλλον ανάπτυξης, το οργανωτικό / ιεραρχικό σχήμα, το στυλ της διοίκησης, την υποστήριξη από την ηγεσία, το σύστημα διοίκησης και ελέγχου αλλά και κάθε παράγοντα του εξωτερικού περιβάλλοντος (πχ. το οικονομικό και πολιτικό σύστημα) που επιδρά στον οργανισμό.
- Παράγοντες που σχετίζονται με τις διαδικασίες. Εδώ συγκαταλέγουμε όλους τους διαδικαστικούς παράγοντες. Διαδικασίες και τεχνικές ανάπτυξης λογισμικού και project management πρωτίστως αλλά και διαδικασίες για συνεργασίες, για προμήθειες, διαδικασίες παρακολούθησης, αποτίμησης και αποδοχής ενός έργου, διαδικασίες συμμόρφωσης με νομοθετικούς περιορισμούς και με εσωτερικούς κανονισμούς κλπ.
- Παράγοντες σχετικοί με το έργο. Στους παράγοντες αυτούς περιλαμβάνονται το είδος των έργων, η πολυπλοκότητά τους, το μέγεθος, η κρισιμότητα της λειτουργίας που εξυπηρετούν, η σαφήνεια των απαιτήσεων, το εύρος, η τεχνολογική πολυπλοκότητα, το πλήθος και το είδος των ενδιαφερομένων κ.λπ.

Ο πίνακας 8.6, που αφορά πρακτικές δημόσιων οργανισμών για επιμέρους προβλήματα, υποδηλώνει ότι υπάρχει μία πληθώρα παραγόντων που επηρεάζουν την ευέλικτη ανάπτυξη στο δημόσιο τομέα, οι οποίοι μπορεί να θεωρηθούν **σημαντικοί** για την επιτυχία των σχετικών εγχειρημάτων και που, υπό τις κατάλληλες συνθήκες, μπορεί να καταστούν **κρίσιμοι** (Βουτσάς και Φιτσιλής, 2015)

Κατηγορία	**Πεδίο**	**Τι μπορούν να κάνουν οι κυβερνήσεις.**	**Τι μπορούν να κάνουν οι δημόσιοι οργανισμοί.**
 **ΟΡΓΑΝΩΣΗ & ΔΙΟΙΚΗΣΗ**	Οργανωτικό πλαίσιο	Παροχή οδηγιών και κατευθύνσεων για ρόλους και αρμοδιότητες Προώθηση διοικητικών διευκολύνσεων για ευέλικτη οργάνωση	Ελαχιστοποίηση θέσεων ευθύνης σε ομάδες Οργάνωση σε ομάδες μικρού μεγέθους Αποφυγή οργανωτικών σχημάτων τύπου πίνακα Ελευθερία και αυτοοργάνωση ομάδων Σαφής καθορισμός ευέλικτων ρόλων και αρμοδιοτήτων
	Υποστήριξη διοίκησης	Παροχή πληροφόρησης στις διοικήσεις	

Κατηγορία	Πεδίο	Τι μπορούν να κάνουν οι κυβερνήσεις.	Τι μπορούν να κάνουν οι δημόσιοι οργανισμοί.
		Ρητή ενσωμάτωση ευελιξίας στη δημόσια στρατηγική Υποχρέωση συμμόρφωσης με σχετικά πρότυπα ή με διεργασίες.	
	Κουλτούρα	Διάδοση της στρατηγικής Εκπόνηση συμβουλευτικών και καθοδηγητικών κείμενων (Οδηγοί εφαρμογής) Διάδοση μελετών – αναφορών Πλατφόρμα επικοινωνίας και ανταλλαγής εμπειριών	Εισαγωγή σε μικρά βήματα Πιλοτικά προγράμματα Προώθηση πρωτοβουλιών των εργαζομένων Δημιουργία κλίματος εμπιστοσύνης ανάμεσα στη διοίκηση και την ομάδα.
ΑΝΘΡΩΠΟΙ	Ικανότητες στελεχών	Υποστηρικτικοί οργανισμοί για παροχή συμβουλευτικών υπηρεσιών προς δημοσίους υπαλλήλους Οδηγοί εφαρμογής, παροχή εκπαιδευτικών πηγών. Διαμόρφωση ευέλικτου περιβάλλοντος για συμβουλευτικές υπηρεσίες.	Πρόσληψη στελεχών της αγοράς Συνεργασία με agile coaches Ενσωμάτωση ειδικών για εκπαίδευση ομάδων Διευκόλυνση εσωτερικής εκπαίδευσης στο πλαίσιο ομάδας. Παροχή κινήτρων για εκπαίδευση και βελτίωση ικανοτήτων.
	Επικοινωνία / Συνεργασία	Οδηγοί για κοινή αντίληψη και γλώσσα. Δημιουργία δικτύων Δημιουργία πλατφόρμας επικοινωνίας (πχ. wiki, blog) και ανταλλαγής εμπειριών	Διαμόρφωση χώρων Συνεργατικό στυλ ηγεσίας Ανοιχτές διαδικασίες, διαφάνεια Επιδίωξη εσωτερικής επικοινωνίας μεταξύ εμπλεκομένων Ενεργητική αναζήτηση feedback των χρηστών Εμπλοκή των τελικών χρηστών, αν είναι πολίτες, μέσω crowdsourcing και δημιουργία ενδιάμεσων εκδόσεων.
ΔΙΕΡΓΑΣΙΕΣ	Προμήθειες	Προσαρμογή νομικού πλαισίου στις ευέλικτες προμήθειες. Θεσμοθέτηση νέων τρόπων προμηθειών Δημιουργία υποστηρικτικών υπηρεσιών (πχ. όπως το digital marketplace)	Αναζήτηση του κατάλληλου μέσου για την ευέλικτη προμήθεια υπηρεσιών. Χρήση framework contracts και υποστηρικτικών υπηρεσιών για ταχύτερες προμήθειες. Συμβόλαια μικρής διάρκειας

Κατηγορία	Πεδίο	Τι μπορούν να κάνουν οι κυβερνήσεις.	Τι μπορούν να κάνουν οι δημόσιοι οργανισμοί.
	Απαιτήσεις	Παροχή οδηγιών (πχ. για τη συγγραφή user stories)	Παρατήρηση στο πεδίο Χρήση ευέλικτων τεχνικών (user stories, use cases…) Διαρκής αναζήτηση feedback από τελικούς χρήστες
	Ανάπτυξη	Δημιουργία πρότυπης διαδικασίας για ευέλικτη ανάπτυξη Δημιουργία υποστηρικτικού φορέα - Παροχή εξειδικευμένων συμβούλων	Έλεγχος για πραγματική ευελιξία στις διαδικασίες ανάπτυξης Έμφαση στην ποιότητα της συνεργασίας με εξωτερικούς συμβούλους Συχνή παράδοση λειτουργικών εκδοχών του συστήματος Test-driven development
 ΕΡΓΑ	Κρισιμότητα	Οδηγοί για έργα κρίσιμης λειτουργίας. Παροχή εξειδικευμένων συμβούλων	Αποδεσμεύσεις λογισμικού σε μικρά, προσεκτικά σχεδιασμένα βήματα Εκτενής, αυτοματοποιημένος λειτουργικός έλεγχος. Προσεκτική σχεδίαση της αρχιτεκτονικής και των βασικών απαιτήσεων εκ των προτέρων.

Πίνακας 8.6: *Πρακτικές ανά κατηγορία σε επίπεδο δημόσιων οργανισμών και κυβερνήσεων*

8.3 Ευελιξία και διοίκηση ανθρώπινου δυναμικού

Οι σημερινές επιχειρήσεις δίνουν ιδιαίτερη έμφαση στη διοίκηση ανθρώπινου δυναμικού καθώς η καινοτομία, η ελκυστική σχεδίαση προϊόντων, η προσαρμοστικότητα, κ.λπ. αποτελούν χαρακτηριστικά που προέρχονται από την ύπαρξη ικανών στελεχών και εργαζομένων.

Πιο συγκεκριμένα, οι βασικές περιοχές λειτουργικότητας σε ένα σύστημα διαχείρισης ανθρώπινου δυναμικού είναι (Staff, 2015):

- Διαχείριση της οργάνωσης της επιχείρησης, που επιτρέπει τον ορισμό των οργανωτικών δομών της επιχείρησης.
- Διοίκηση προσωπικού, που επιτρέπει τη δημιουργία και συντήρηση του βασικού αρχείου των εργαζόμενων της επιχείρησης.
- Ανάπτυξη προσωπικού (personnel development), που περιλαμβάνει τις διαδικασίες πρόσληψης προσωπικού, την εκπαίδευση του προσωπικού, καθώς και την αξιολόγηση της απόδοσης του προσωπικού.
- Διαχείριση παροχών προσωπικού (benefit management), που επιτρέπει τη διαχείριση των παροχών της επιχείρησης προς τους εργαζόμενους. Είναι παροχές σε χρήμα ή είδος που καταβάλλει ο εργοδότης σε εργαζόμενους χωρίς να εκκαθαρίζονται μέσω της μηνιαίας μισθοδοσίας. Ενδεικτικά, είναι η παροχή εταιρικού αυτοκινήτου, ιδιωτικής ασφάλισης, ένταξης σε συνταξιοδοτικό πρόγραμμα, κάλυψη ενοικίου, εξόδων κίνησης κ.λπ. Συνήθως οι ανωτέρω παροχές δίνονται σε στελέχη της εταιρείας ανάλογα με το επίπεδο της θέσης τους στην οργανωτική δομή της επιχείρησης.

- Διαχείριση χρόνου προσωπικού (time management), που επιτρέπει την καταγραφή και παρακολούθηση του χρόνου των εργαζόμενων.
- Μισθοδοσία προσωπικού (payroll).

Υπάρχει ένα αυξανόμενο ενδιαφέρον στον επιχειρηματικό κόσμο, και πολλοί οργανισμοί έχουν στραφεί στην εφαρμογή ευέλικτων πρακτικών στη διαχείριση ανθρώπινων πόρων (Human Resource Management – HRM). Οι εργαζόμενοι σήμερα επιθυμούν μεγαλύτερη ευελιξία και αυτονομία όταν εργάζονται και θέλουν η εργασία τους και η προσπάθειά τους να αναγνωρίζεται από την επιχείρηση. Υπάρχουν δύο οπτικές γωνίες για να δούμε την έννοια της ευελιξίας σε σχέση με τη διοίκηση ανθρώπινων πόρων:

- HRM για την ευέλικτη διαχείριση έργων, που στοχεύει στον σχεδιασμό και την εφαρμογή HRM συστημάτων που διευκολύνουν την εφαρμογή των ευέλικτων μεθόδων στην επιχείρηση.
- Η ευελιξία για το HRM, που αναφέρεται στο πώς οι ευέλικτες αρχές και πρακτικές εφαρμόζονται στη λειτουργία του HRM.

Αν θέλαμε να ορίσουμε την ευέλικτη διοίκηση ανθρώπινου δυναμικού θα λέγαμε ότι επιδιώκει να ελαχιστοποιήσει τη σπατάλη και να βελτιστοποιήσει τη ροή της αξίας προς τους πελάτες του, αναθέτοντας τη λειτουργία της διοίκησης ανθρωπίνων πόρων σε διεπιστημονικές, εξουσιοδοτημένες ομάδες, που ευθυγραμμίζονται συνεχώς με τις μεταβαλλόμενες επιχειρηματικές ανάγκες, οι οποίες ανιχνεύονται μέσω της ανοιχτής επικοινωνίας με το περιβάλλον, ενώ οι ομάδες λειτουργούν σε σύντομους χρονικά κύκλους. Οι αρχές της ευελιξίας αντικατοπτρίζονται σε όλες τις πτυχές της λειτουργίας του HRM, συμπεριλαμβανομένων των δομών, των ρόλων, των διεργασιών και των εργαλείων, καθώς και στις δεξιότητες και τις συμπεριφορές του προσωπικού της διοίκησης ανθρώπινου δυναμικού (McMackin, & Heffernan, 2021).

Σε πρακτικούς όρους η ευέλικτη προσέγγιση για το HR μπορεί να εφαρμοστεί στην προσέλκυση εργαζομένων, στις διεργασίες μάθησης και ανάπτυξης προσωπικού, στη διαχείριση των κινήτρων εργαζομένων, στην αξιολόγηση της απόδοσης, κ.λπ. (Thoren, 2017). Ο Πίνακας 8.7 παρουσιάζει τις διαφορές μεταξύ του παραδοσιακού HRM και του ευέλικτου HRM (Ranasinghe & Sangaradeniya, 2021; Maršíková, & Revutska, 2021; Denning, 2018).

Διάσταση	**Παραδοσιακή Διοίκηση Ανθρώπινου Δυναμικού**	**Ευέλικτη Διοίκηση Ανθρώπινου Δυναμικού**
Λήψη αποφάσεων	Οι αποφάσεις λαμβάνονται ιεραρχικά από τη διοίκηση και δεν εκχωρούνται στα τμήματα του οργανισμού.	Οι αποφάσεις λαμβάνονται από αυτόνομες ομάδες
Είδος διεργασιών	Υπάρχουν τυποποιημένες διεργασίες που εφαρμόζονται σε ανάδραση σε εξωτερικά γεγονότα. Εστίαση στις διεργασίες.	Υπάρχουν διεργασίες που εφαρμόζονται προληπτικά και με βάση τις ανάγκες. Εστίαση στους ανθρώπους.
Η λειτουργία της διοίκησης προσωπικού	Υπάρχουν εργαζόμενοι που ειδικεύονται στη διαχείριση του ανθρώπινου δυναμικού.	Οι εργαζόμενοι έχουν ικανότητες τόσο εξειδικευμένες όσο και οριζόντιες (T-shaped) και οι λειτουργίες της διαχείρισης ανθρώπινου δυναμικού αναλαμβάνονται από αυτούς.
Σημείο εστίασης	Επικεντρώνεται κυρίως στις λειτουργίες διαχείρισης ανθρώπινου δυναμικού.	Επικεντρώνεται κυρίως στις λειτουργίες διαχείρισης ανθρώπινου δυναμικού που παράγουν αξία.
Προσέγγιση HR	Η εικόνα για το ανθρώπινο δυναμικό είναι συνήθως αρνητική. Ακολουθεί τη θεωρία X (McGregor, 1960).	Η εικόνα για το ανθρώπινο δυναμικό είναι συνήθως αρνητική. Ακολουθεί τη θεωρία Y.
Κινητοποίηση εργαζομένων	Η απόδοση των εργαζομένων βασίζεται σε εξωγενή κίνητρα.	Η απόδοση των εργαζομένων βασίζεται σε εσωτερικά κίνητρα.
Ο ρόλος του HR	Ο ρόλος της διοίκησης του ανθρώπινου δυναμικού είναι να εφαρμόζει κανόνες και να ελέγχει.	Ο ρόλος της διοίκησης του ανθρώπινου δυναμικού είναι να υποστηρίζει και να καθοδηγεί την οργανωτική ευελιξία

Διάσταση	Παραδοσιακή Διοίκηση Ανθρώπινου Δυναμικού	Ευέλικτη Διοίκηση Ανθρώπινου Δυναμικού
Στρατηγική ανατροφοδότησης	Σπάνια υπάρχουν ανατροφοδοτήσεις για την απόδοση των εργαζομένων. Όταν υπάρχουν είναι περιοδικές (εξαμηνίες ή ετήσιες).	Υπάρχει συχνή ανατροφοδότηση για την απόδοση των εργαζομένων.
Επικοινωνία	Τυπική	Άτυπη

Πίνακας 8.7: *Οι διαφορές παραδοσιακού HPM και ευέλικτου HRM*

Συνοψίζοντας μπορούμε να πούμε ότι τρεις είναι οι βασικές αλλαγές που απαιτούνται για την υιοθέτηση της ευέλικτης προσέγγισης στη διοίκηση ανθρώπινου δυναμικού (Iancu, 2022). Αυτές είναι:

- Διαχείριση ανθρώπινου δυναμικού σε συνεργασία με όλα τα στελέχη της επιχείρησης.
- Ταχύτητα στην αλλαγή των διεργασιών/πρακτικών. Μία από τις πιο ισχυρές πτυχές μιας ευέλικτης προσέγγισης του HR είναι η ταχύτητα με την οποία επιτρέπει την ανάπτυξη των διεργασιών και των πρακτικών που θα χρησιμοποιούνται καθώς οι αποτυχίες και τα λάθη αξιολογούνται άμεσα.
- Συνεχή αξιολόγηση του προσωπικού και ανατροφοδότηση. Σε πολλούς οργανισμούς ακολουθείται ένας ετήσιος κύκλος, στον οποίο οι διάφορες διαδικασίες μπαίνουν σε ένα ετήσιο ημερολόγιο. Έτσι, οι αμοιβές, η διαχείριση της απόδοσης, η διαχείριση ταλέντων και άλλες διαδικασίες ανθρώπινου δυναμικού αναθεωρούνται συνήθως σε ετήσια βάση. Ωστόσο, στην ευέλικτη προσέγγιση οι αξιολόγηση είναι καθημερινή ενώ για τα προβλήματα που προκύπτουν βρίσκουμε λύσεις άμεσα.

Σε έναν ευέλικτο οργανισμό, οι λειτουργίες που απαιτούνται με αυτές που παρέχονται σε ένα παραδοσιακό οργανισμό είναι κοινές, δηλαδή: πρόσληψη προσωπικού, επαγγελματική ανάπτυξη προσωπικού, διαχείριση απόδοσης εργαζομένων αλλά με ένα τρόπο που να ανταποκρίνεται στις συνεχείς αλλαγές, στην κουλτούρα και το στυλ ευέλικτης εργασίας του οργανισμού.

8.3.1 Ευέλικτες προσλήψεις προσωπικού

Η σωστή διαδικασία στελέχωσης προσωπικού είναι πρωταρχικής σημασίας, αποτελεί σημαντική παράμετρο της διαχείρισης του ανθρώπινου δυναμικού και αποτελεί έναν από τους πιο σημαντικούς παράγοντες για την επιβίωση των οργανισμών καθώς και για την απόκτηση ανταγωνιστικού πλεονεκτήματος. Με τον όρο στελέχωση εννοούμε τις λειτουργίες που διασφαλίζουν ότι ο κάθε οργανισμός έχει στο παρόν και θα έχει στο μέλλον το απαιτούμενο και κατάλληλο σε γνώσεις και ικανότητες ανθρώπινο δυναμικό. (Parry & Tyson, 2011)

Η στελέχωση αποτελεί μια δύσκολη διαδικασία που ξεκινά με τον προγραμματισμό του ανθρώπινου δυναμικού, την περιγραφή της κάθε θέσης εργασίας, το προφίλ του κατάλληλου υποψηφίου, την προσέλκυση, την τελική επιλογή, την τοποθέτηση, υποδοχή, εγκατάσταση και πρωταρχική εκπαίδευση του προσωπικού μέσα στον οργανισμό. Η βασική δυσκολία της σωστής στελέχωσης έγκειται στο γεγονός ότι σχετίζεται άμεσα με τον ανθρώπινο παράγοντα - δηλαδή άνθρωποι επιλέγουν ανθρώπους.

Σήμερα, τα επιχειρηματικά μοντέλα αλλάζουν γρήγορα, δημιουργώντας αστάθεια στον όγκο προσλήψεων καθώς αλλάζουν οι ανάγκες των εταιρειών. Επίσης, οι ίδιοι οι ρόλοι εξελίσσονται γρήγορα, δημιουργώντας αβεβαιότητα για το προφίλ των εργαζομένων που πρέπει να προσληφθούν. Ταυτόχρονα, στις ειδικότητες αιχμής, οι προσλήψεις είναι ακόμη πιο δύσκολες, τόσο σε κόστος όσο και σε χρόνο, με δεδομένο ότι σε αυτές τις ειδικότητες υπάρχει έλλειψη προσωπικού (Wiles, 2019).

Οι παραπάνω ανάγκες δημιουργούν ένα κενό στις σύγχρονες επιχειρήσεις που τις αναγκάζει να υιοθετήσουν ένα πιο ευέλικτο μοντέλο στην πρόσληψη προσωπικού. Αυτή η αλλαγή χαρακτηρίζεται από:

- Απομάκρυνση από τον σταθερό σχεδιασμό και την ανάθεση εργασιών σε εργαζομένους (resourcing). Με βάση τις προβλέψεις αναγκών προσωπικού από κάτω προς τα πάνω (bottom-up) προσαρμόζουμε τις παρεχόμενες υπηρεσίες ή ανακατανέμουμε το προσωπικό με βάση τις ανάγκες (top-down).
- Επικέντρωση στις μακροπρόθεσμες τάσεις της αγοράς, και όχι μόνο στις τρέχουσες επιχειρηματικές ανάγκες. Η πρόσληψη προσωπικού σε ειδικότητες αιχμής είναι δύσκολη και συνεπώς η επιχείρηση πρέπει να προσβλέπει στην απόκτηση τεχνογνωσίας σε συγκεκριμένες περιοχές και όχι αποκλειστικά στην ικανοποίηση των τρεχουσών αναγκών.
- Ανάλυση επιχειρηματικών και κλαδικών δεδομένων για την πρόβλεψη των αναγκών

8.3.2 Ευέλικτη αξιολόγηση της απόδοσης προσωπικού

Η αξιολόγηση της απόδοσης προσωπικού (performance management) έχει αλλάξει σημαντικά τα τελευταία χρόνια. Η πιο σημαντική αλλαγή ήταν η εισαγωγή του συστήματος αξιολόγησης των 360 μοιρών.

Ωστόσο πρόσφατα, οι επιχειρήσεις άρχισαν να υιοθετούν ευέλικτες διαδικασίες διαχείρισης αξιολόγησης προσωπικού, μια προσέγγιση που ουσιαστικά αντικατοπτρίζει το σύγχρονο εργασιακό περιβάλλον. Οι χώροι εργασίας σήμερα, ιδιαίτερα στην επιχειρήσεις τεχνολογίας, έχουν εξελιχθεί από μια οργάνωση με ιεραρχική μορφή, σε μια ανοιχτή, συνεργατική οργάνωση, όπου πολλές ομάδες ενώνουν τις δυνάμεις τους για να επιτύχουν έναν κοινό σκοπό. Οι εργαζόμενοι απαιτούν να ορίζουν τους ατομικούς τους στόχους και τον τρόπο για την επίτευξη αυτών, αντί για την παραδοσιακή ανάθεση εργασιών σε αυτούς από τους προϊσταμένους. Αυτή η νοοτροπία προκύπτει εν μέρει από γεγονός ότι ο εργασιακός χώρος είναι ένας διασυνδεδεμένος χώρος, όπου τα μηνύματα και η πληροφορία ρέει άμεσα, ώστε οι εργαζόμενοι και οι προϊστάμενοι να επικοινωνούν σε πραγματικό χρόνο. Συνεπώς, η ανατροφοδότηση των εργαζομένων είναι άμεση και δεν χρειάζεται να περιμένουμε για μια ετήσια αξιολόγηση (Ganesh, 2019).

Σύμφωνα με τη μελέτη της McKinsey&Company (2019), οι ευέλικτοι οργανισμοί θα πρέπει, να εφαρμόσουν τρεις βασικές πρακτικές διαχείρισης απόδοσης ώστε να προσαρμοστούν στην ευέλικτη προσέγγιση. Πιο συγκεκριμένα θα πρέπει:

- Σύνδεση στόχων των εργαζομένων με επιχειρηματικές προτεραιότητες. Το ακρώνυμο που χρησιμοποιείται για να υποδηλώσει την ιδέα αυτή είναι το OKR (Objectives and Key Results). Το 1999, ο John Doerr (2018) παρουσίασε την ιδέα των OKR (στόχοι και βασικά αποτελέσματα) στην Google. Συνεπώς, οι στόχοι πρέπει να υποστηρίζουν το όραμα ή τις κορυφαίες προτεραιότητες της επιχείρησης και πρέπει να είναι ποιοτικοί και όχι μετρήσιμοι. Θα πρέπει να είναι φιλόδοξοι, χρονικά ορατοί και να οδηγούν σε ενέργειες. Αντίθετα, τα βασικά αποτελέσματα πρέπει να είναι ποσοτικά. Στην ιδανική περίπτωση, κάθε στόχος θα υποστηρίζεται από έως τέσσερα βασικά αποτελέσματα. Αυτό βοηθά να κρατήσει τους εργαζόμενους προσηλωμένους στο στόχο. Επιπλέον, κάθε βασικό αποτέλεσμα πρέπει να βασίζεται στα αποτελέσματα, όχι στα καθήκοντα, καθώς και θα πρέπει να επικεντρώνεται στο πού βρίσκεται ο εργαζόμενος σήμερα σε σχέση με τη μελλοντική του πορεία. Η εισαγωγή στόχων μπορεί να γίνει και για την ομάδα και όχι μόνο για μεμονωμένα άτομα. Οι στόχοι της ομάδας θα πρέπει να συζητούνται αναλυτικά όπως επίσης και τα αποτελέσματα. Κατά τη διάρκεια των ανασκοπήσεων θα πρέπει να αποφασίζονται και οι κατάλληλες αλλαγές. Τέλος θα πρέπει να υπάρχει απόλυτη διαφάνεια τόσο των στόχων όσο και του αποτελέσματος – απόδοσης.
- Εκπαίδευση της διοίκησης με «coaching skills». Οι προπονητικές ικανότητες είναι ιδιαίτερα σημαντικές για τη δημιουργία και ανάπτυξη ομάδων. Σε αυτή την κατηγορία ικανοτήτων περιλαμβάνονται οι ικανότητες της ενσυναίσθησης, της επιμονής, της καινοτομικότητας, της καλής επικοινωνίας, της καθοδήγησης, της ειλικρίνειας, κ.λπ. Για να αναπτύξουμε αυτές τις ικανότητες εκτός των άλλων θα πρέπει να αποσαφηνίσουμε τον ρόλο του εργαζόμενου – ηγέτη μέσα στην επιχείρηση, να αναπτύξουμε μια κουλτούρα συζήτησης των προβλημάτων και συνεχούς ανατροφοδότησης.

- Διαφοροποίηση της αξιολόγησης απόδοσης των εργαζομένων. Αν και οι ευέλικτες ομάδες εργάζονται ομαδικά και υπάρχει συνυπευθυνότητα ως προς το τελικό αποτέλεσμα, θα πρέπει να διαφοροποιήσουμε την αξιολόγηση του κάθε εργαζομένου, ανάλογα με την ατομική του συνεισφορά στην απόδοση της ομάδας. Επίσης θα πρέπει να δώσουμε έμφαση και σε μη χρηματικά κίνητρα και ανταμοιβές.

Ο Πίνακας 8.8 παρουσιάζει τις διαφορές στην αξιολόγηση προσωπικού μεταξύ του παραδοσιακού και του ευέλικτου τρόπου.

Παραδοσιακή Διοίκηση Ανθρώπινου Δυναμικού	**Ευέλικτη Διοίκηση Ανθρώπινου Δυναμικού**
Οι αξιολογήσεις του προσωπικού γίνονται είναι ετήσιες.	Συνεχής αξιολόγηση, συζήτηση και ανατροφοδότηση του εργαζομένου.
Η αξιολόγηση είναι ποσοτική και πολλές φορές υπάρχει βαθμολογία.	Δεν υπάρχει ποσοτική αξιολόγηση
Η αξιολόγηση γίνεται από τους προϊσταμένους.	Η αξιολόγηση είναι 360 μοιρών.
Η αναγνώριση των ικανοτήτων του εργαζομένου γίνεται από τους προϊσταμένους.	Η αναγνώριση των ικανοτήτων του εργαζομένου γίνεται από την ομάδα, τους πελάτες, κ.λπ.
Υπάρχει ένα πλάνο για την εξέλιξη του εργαζομένου (career path)	Η εξέλιξη δεν είναι προκαθορισμένη αλλά υπάρχει μεγάλη κινητικότητα.
Η αξιολόγηση είναι ατομική και υπάρχει εμπιστευτικότητα.	Υπάρχει διαφάνεια καθώς οι στόχοι, το αποτέλεσμα και η ανταμοιβή είναι γνωστή σε όλους.

Πίνακας 8.8: *Οι διαφορές μεταξύ της παραδοσιακής και της ευέλικτης αξιολόγησης προσωπικού*

.

Βιβλιογραφία/Αναφορές

Agile Business Consortium. What is business Agility https://www.agilebusiness.org/page/WhatisBusinessAgility

Boti, E., Damasiotis, V., & Fitsilis, P. (2021, June). Skills Development Through Agile Capstone Projects. In *International Conference on Frontiers in Software Engineering* (pp. 97-112). Springer, Cham.

Cooke, J. L. (2014). *Agile Productivity Unleashed: Proven approaches for achieving real productivity gains in any organization*. IT Governance Publishing.

Darino, L., Sieberer, M., Vos, A., & Williams, O. (2019). Performance management in agile organizations. *mckinsey. com*.

Denning, S. (2018). The emergence of Agile people management. *Strategy & Leadership*.

Delhij, A., van Solingen, R., & Wijnands, W. (2015). The eduScrum guide. *The rules of the Game*.

Digital.ai (2021). 15th State of Agile Report. https://info.digital.ai/rs/981-LQX-968/images/SOA15.pdf

Doerr, J. (2018). *Measure what matters: OKRs: The simple idea that drives 10x growth*. Penguin UK.

Ferreira, E. P., & Martins, A. (2016, June). EduScrum–The Empowerment of Students in Engineering Education?. In *Proceedings of the 12th International CDIO Conference* (pp. 596-605).

Ganesh, S. (2019). An Agile Performance Management Process, People Matters. https://www.myadrenalin.com/peoplematters/an-agile-performance-management.pdf

Iancu, S. (2022). Agile HR: All You Need to Know to Get Started, Academy to Innovate HR. https://www.aihr.com/blog/agile-hr/

Kupi, M., & McBride, K. (2021). Agile Development for Digital Government Services: Challenges and Success Factors. In *International Conference on Electronic Participation* (pp. 139-150). Springer, Cham.

Layton, M. C., & Ostermiller, S. J. (2017). Agile project management for dummies, 2nd edition. John Wiley & Sons.

Layton, M. C. & Morrow, D. (2015). *Scrum for dummies*. 2nd edition. John Wiley & Sons.

McKinsey & Company (2017). The 5 Trademarks of Agile Organizations.

Darino, I., Sieberer, M., Vos, A. & Williams, O. (2019). Performance management in agile organizations. McKinsey & Company. https://www.mckinsey.com/business-functions/people-and-organizational-performance/our-insights/performance-management-in-agile-organizations.

McGregor, D. (1960). Theory X and theory Y. *Organization theory*, *358*(374), 5.

McMackin, J., & Heffernan, M. (2021). Agile for HR: Fine in practice, but will it work in theory?. *Human Resource Management Review*, *31*(4), 100791.

National Audit Office (2012). A snapshot of the use of Agile delivery in central government. Review.

Parry, E., & Tyson, S. (2011). Desired goals and actual outcomes of e-HRM. *Human resource management journal*, *21*(3), 335-354.

Parsons, D., & MacCallum, K. (2019). Agile and lean concepts for teaching and learning. *Agile and Lean Concepts for Teaching and Learning. Springer Singapore. https://doi. org/10.1007/978-981-13-2751-3*.

Ranasinghe, V. R., & Sangaradeniya, Y. M. S. W. V. (2021). Agile human resource management. *Human Resource Management in Challenging Environment*, *2021*, 23-31.

Maršíková, K., & Revutska, O. (2021). Agile Approach in Human Resource Management: Focus on Generation Y. *EM Ekonomie a Management*.

Stankovic, D., Nikolic, V., Djordjevic, M., & Cao, D. B. (2013). A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *Journal of Systems and Software*, *86*(6), 1663-1678.

Salza, P., Musmarra, P., & Ferrucci, F. (2019). Agile methodologies in education: A review. *Agile and lean concepts for teaching and learning*, 25-45.

Stewart, J. C., DeCusatis, C. S., Kidder, K., Massi, J. R., & Anne, K. M. (2009). Evaluating agile principles in active and cooperative learning. *Proceedings of Student-Faculty Research Day, CSIS, Pace University*, B3.

Thoren, P. (2017). *Agile people*. Austin, TX: Lioncrest Publishing.

U.S. Government Accountability Office (2012), Effective Practices and Federal Challenges in Applying Agile Methods.

UK National Audit Office (2012). Governance for Agile delivery.

Wijnands, W. (2020). The EduScrum Guide. https://eduscrum.org

Wiles, J. (2019). Why You Need a More Agile Recruiting Approach. https://www.gartner.com/smarterwithgartner/why-you-need-a-more-agile-recruiting-approach.

Βουτσάς, Η., Φιτσιλής, Π. (2015). Υιοθέτηση ευέλικτων μεθόδων ανάπτυξης λογισμικού από τη δημόσια διοίκηση. Οικονομικό Πανεπιστήμιο Αθηνών.

Φιτσιλής, Π. (2018). Σύγχρονα πληροφοριακά συστήματα επιχειρήσεων (2η έκδοση). Broken Hill Publishers

Κριτήρια αξιολόγησης

Κριτήριο αξιολόγησης 1

Ποια είναι τα πέντε βασικά χαρακτηριστικά των ευέλικτων επιχειρήσεων/οργανισμών.

Απάντηση/Λύση

Σύμφωνα με τη μελέτη της εταιρείας McKinsey (2017) πέντε είναι τα βασικά χαρακτηριστικά των ευέλικτων επιχειρήσεων/οργανισμών είναι:

- Κοινός σκοπός και ξεκάθαροι σε όλους τους συμμετέχοντες στόχοι.
- Η επιχείρηση είναι ένα δίκτυο αυτο-οργανούμενων ομάδων.
- Η επιχείρηση λαμβάνει αποφάσεις γρήγορα, με γρήγορους κύκλους μάθησης.
- Η επιχείρηση έχει δυναμικούς εργαζόμενους που παρακινούνται από το πάθος δημιουργίας και καινοτομίας και όχι αποκλειστικά από οικονομικά κίνητρα.
- Η επιχείρηση χρησιμοποιεί τεχνολογίες αιχμής.

Κριτήριο αξιολόγησης 2

Ποιος είναι ο ρόλος του εκπαιδευτικού στη διεργασία eduScrum;

Απάντηση/Λύση

Ο εκπαιδευτικός στη διεργασία eduScrum είναι υπεύθυνος για τα ακόλουθα:

- καθορίζει τι θα μάθουν και για ποιο λόγο οι εκπαιδευόμενοι.
- παρακολουθεί και βελτιώνει την ποιότητα των μαθησιακών αποτελεσμάτων,
- ελέγχει και αξιολογεί τα μαθησιακά αποτελέσματα και παρακολουθεί την προσωπική ανάπτυξη του κάθε εκπαιδευόμενου και
- έχει ένα διττό ρόλο, ως ιδιοκτήτης προϊόντος και υπεύθυνος της ομάδας

Επίσης, ο εκπαιδευτικός είναι υπεύθυνος για τη διάδοση της φιλοσοφίας eduScrum.

Κριτήριο αξιολόγησης 3

Ποιά είναι τα χαρακτηριστικά της ομάδας στη διεργασία eduScrum;

Απάντηση/Λύση

Οι ομάδες εκπαιδευομένων στη διεργασία eduScrum έχουν τα ακόλουθα χαρακτηριστικά:

- Είναι ατοοργανωμένες. Κανείς (ούτε καν ο εκπαιδευτικός) δεν μπορεί να υπαγορεύσει στην ομάδα πώς να επιτύχει τους μαθησιακούς στόχους.
- Οι ομάδες είναι διεπιστημονικές, δηλαδή υπάρχουν μέλη με όλες τις απαραίτητες δεξιότητες ώστε να είναι σε θέση να επιτύχουν τους μαθησιακούς στόχους ως ομάδα και ταυτόχρονα να είναι σε θέση να αναπτυχθούν προσωπικά.

- Τα μέλη της φοιτητικής ομάδας μπορεί να έχουν συγκεκριμένες δεξιότητες ή τομείς εστίασης, αλλά η ευθύνη ανήκει στην ομάδα συνολικά.
- Τα μέλη της ομάδας μπορούν να αποφασίσουν ατομικά εάν θέλουν να χρησιμοποιήσουν τις ικανότητες και δεξιότητές τους ή να αναπτύξουν νέες.
- Η ομάδα παρακολουθεί την πρόοδο της ποιότητας που επιτεύχθηκε, με τη χρήση των κριτηρίων εορτασμού και τις απαιτήσεις της εργασίας.

Κριτήριο αξιολόγησης 4

Ποιο είναι το βέλτιστο μέγεθος ομάδας στη διεργασία eduScrum και γιατί;

Απάντηση/Λύση

Το βέλτιστο μέγεθος της ομάδας πρέπει να είναι αρκετά μικρό ώστε να είναι η ομάδα λειτουργική και ταυτόχρονα αρκετά μεγάλο ώστε η ομάδα να μπορεί να υλοποιήσει μια σημαντική εργασία. Ο βασικός κανόνας είναι η δημιουργία ομάδων με τέσσερα ή πέντε μέλη. Λιγότερο από τρία μέλη σημαίνει ότι η αλληλεπίδραση μειώνεται και οι αναγκαίες δεξιότητες δεν υπάρχουν, επαρκώς. Όταν υπάρχουν περισσότερα από έξι μέλη τότε απαιτείται υπερβολικός συντονισμός. Οι μεγάλες ομάδες δημιουργούν υπερβολική πολυπλοκότητα ώστε να ελεγχθούν από μια εμπειρική διαδικασία.

Κριτήριο αξιολόγησης 5

Ποια είναι τα στοιχεία που περιλαμβάνονται σε ένα «flap” στη διεργασία eduScrum;

Απάντηση/Λύση

Το "Flap" είναι μια χρονολογική αναπαράσταση του έργου κατά τη διάρκεια ενός sprint. Η επάνω γραμμή του "Flap" καθορίζει το έργο, σε ποια ομάδα ανήκει το "Flap" και από ποια μέλη αποτελείται. Οι εργασίες του "Flap" μπορεί να είναι "εκκρεμείς εργασίες", "σε εξέλιξη" καθώς και "ολοκληρωμένες". Το "Flap" πρέπει να ενημερώνεται τακτικά, ώστε πάντα να δείχνει τις τρέχουσες εργασίες της ομάδας της ομάδας.

Επιπλέον των παραπάνω πεδίων, το "Flap" περιέχει και τα ακόλουθα:

- Ιστορίες Χρηστών
- Κριτήρια Εορτασμού
- Ορισμός ολοκλήρωσης δραστηριοτήτων - Definition of Doing. Είναι ο ορισμός του πότε μια εργασία έχει ολοκληρωθεί.
- Ορισμός της επικοινωνίας - Definition of communication. Το πώς η ομάδα επικοινωνεί.
- Ορισμός της διασκέδασης - Definition of Fun. Η διασκέδαση/ευχαρίστηση είναι ένα σημαντικό κίνητρο για τους εκπαιδευόμενους και ως εκ τούτου είναι απαραίτητη για την επίτευξη καλύτερων μαθησιακών αποτελεσμάτων. Συνεπώς, οι εκπαιδευόμενοι θα πρέπει επίσης να υποδείξουν τι χρειάζονται για να εξασφαλίσουν ευχάριστη εργασία.
- Εμπόδια της ομάδας

Κριτήριο αξιολόγησης 6

Δώστε ένα παράδειγμα κριτηρίου εορτασμού, και ορισμού της διασκέδασης;

Απάντηση/Λύση

Κριτήριο Εορτασμού (Celebration Criteria).

Τα κριτήρια εορτασμού είναι κριτήρια με τα οποία αποφασίζουμε κατά πόσο μια εργασία – ιστορία χρήστη έχει ολοκληρωθεί. Για παράδειγμα αν η ιστορία χρήστη είναι:

- Ως ομάδα ΑΛΦΑ θέλουμε να κατανοήσουμε τη διαδικασία διαπραγμάτευσης για τη διοργάνωση μιας σχολικής εκδρομής ώστε να μπορούμε να έχουμε την καλύτερη δυνατή τιμή.
- Τότε τα κριτήρια εορτασμού θα μπορούσε να είναι:
- Η ομάδα δημιούργησε ένα πόστερ με τη διαδικασία διαπραγμάτευσης
- Η ομάδα δημιούργησε μια κάρτα αναφοράς στη οποία αναφέρονται τα βασικά σημεία που κάποιος θα πρέπει να λάβει υπόψη του
- Ορισμός της διασκέδασης - Definition of Fun.

Ο ορισμός της διασκέδασης είναι γενικός και ασαφής και συνεπώς θα μπορούσε να είναι οτιδήποτε ευχαριστεί ή βελτιώνει την ομάδα. Για παράδειγμα:

- Η ενημέρωση της ομάδας για τα καλά νέα της ημέρας είτε μέσα στην ομάδα είτε μέσα στο σχολείο
- Η ενημέρωση της ομάδας για την επίλυση ενός δύσκολου προβλήματος
- Η ανακήρυξη του καλύτερου μέλους της ομάδας σε καθημερινή βάση
- Διοργάνωση κοινωνικής εκδήλωσης σε συνδυασμό με την επίτευξη ενός ορόσημου

Κριτήριο αξιολόγησης 7

Ποιοι είναι οι βασικοί παράγοντες που επηρεάζουν την εισαγωγή των ευέλικτων μεθόδων στο δημόσιο τομέα;

Απάντηση/Λύση

Με βάση όλα τα προαναφερθέντα και μετά από μελέτη της σχετικής βιβλιογραφίας, θα μπορούσαμε να ομαδοποιήσουμε τους παράγοντες που επηρεάζουν την υιοθέτηση ευέλικτων μεθόδων ανάπτυξης ως εξής:

- Οργανωτικοί παράγοντες. Οι οργανωτικές προκλήσεις είναι αυτές που σχετίζονταν με τις δομές εντός του οργανισμού, καθώς η εφαρμογή μεθόδων ευέλικτης ανάπτυξης απαιτεί δομικές αλλαγές, ειδικά όσον αφορά τις διεργασίες που αφορούν την εργασία και την ομάδα.
- Μεθοδολογικοί παράγοντες. Οι προκλήσεις αυτές σχετίζονται με την ευέλικτη προσέγγιση. Αυτές οι προκλήσεις σχετίζονται κυρίως με οργανισμούς που προσπαθούν να εφαρμόσουν την ευέλικτη προσέγγιση για πρώτη φορά, έχοντας μια ελλιπή ή μερική κατανόηση των μεθόδων.
- Σχετικοί με τον τελικό χρήστη. Η τρίτη κατηγορία προκλήσεων είναι αυτές που σχετίζονται με τον τελικό χρήστη, τη συμμετοχή και την άμεση λήψη αποφάσεων σχετικά με το υπό ανάπτυξη προϊόν. Είναι πολύ δύσκολο για ένα δημόσιο υπάλληλο να αποφασίσει με άμεσο και γρήγορο τρόπο για το ποια μορφή πρέπει να έχει η κάθε απαίτηση, πότε η έκδοση είναι ολοκληρωμένη, πότε η ψηφιακή υπηρεσία είναι αποδεκτή, κ.λπ.
- Ρυθμιστικοί παράγοντες. Συχνά, η νομοθεσία και οι κανονισμοί δεν ευνοούν τη χρήση ευέλικτων μεθόδων ανάπτυξης. Σε πολλές περιπτώσεις, ένα μη υποστηρικτικό νομοθετικό πλαίσιο είναι ένα περιβάλλον που πρέπει να ξεπεραστεί. Το βασικότερο πρόβλημα είναι η νομοθεσία/κανονισμοί προμηθειών. Η ευέλικτη ανάπτυξη προϋποθέτει ότι η εργασία στα έργα είναι ευμετάβλητη και το τελικό αποτέλεσμα δεν είναι απαραίτητα γνωστό. Ωστόσο, η νομοθεσία, ειδικά αυτή που

σχετίζεται με τις προμήθειες, απαιτεί σαφή παραδοτέα και στόχους καθώς και ένα σαφώς καθορισμένο τελικό προϊόν. Συνεπώς, υπάρχει ανάγκη για «νομική» καινοτομία ή αλλαγή των πρακτικών που χρησιμοποιούνται στις προμήθειες.

Κριτήριο αξιολόγησης 8

Δώστε ένα παράδειγμα στοχοθεσίας OKR (Objectives and key results). Δώστε τον στόχο καθώς και τα βασικά αποτελέσματα.

Απάντηση/Λύση

Ας δώσουμε ένα παράδειγμα με ένα απλό στόχο. Έστω ότι ο στόχος είναι «να αποκτήσουμε καλή φυσική κατάσταση μέσα σε τρεις μήνες». Ο στόχος αυτός συνδέεται εύκολα με βασικά αποτελέσματα τα οποία μπορεί να είναι,

- Να μπορούμε να τρέξουμε 3 χλμ σε 20 λεπτά
- Να μπορούμε να κάνουμε 50 κάμψεις
- Να μπορούμε να κάνουμε 200 κοιλιακούς

Μόλις καθοριστούν τα OKR, η διαδικασία αξιολόγησης θα πρέπει να επαναλαμβάνεται κάθε τρίμηνο. Κατά την αξιολόγηση, κάθε βασικό αποτέλεσμα βαθμολογείται σε κλίμακα από το ένα έως το εκατό και στη συνέχεια υπολογίζεται ο μέσος όρος.

Εάν χρησιμοποιήσουμε το προηγούμενο παράδειγμα μας για ένα OKR, η διαδικασία αξιολόγησης θα μπορούσε να είναι:

- Να μπορούμε να τρέξουμε 3 χλμ σε 20 λεπτά. Επιτεύχθηκε το 75% του στόχου.
- Να μπορούμε να κάνουμε 50 κάμψεις. Επιτεύχθηκε το 50% του στόχου
- Να μπορούμε να κάνουμε 200 κοιλιακούς. Επιτεύχθηκε το 50% του στόχου

Συνεπώς, ο στόχος «να αποκτήσουμε καλή φυσική κατάσταση μέσα σε τρεις μήνες» έχει επιτευχθεί κατά 68%.

Προφανώς, ο σκοπός μας δεν είναι η επίτευξη ενός τέλειου σκορ 100%. Εάν η βαθμολογία που επιτεύχθηκε είναι πάνω από 90%, πιθανόν να σημαίνει ότι ο στόχος ήταν πολύ εύκολος να επιτευχθεί και ότι δεν υπάρχουν αρκετά περιθώριο βελτίωσης. Εάν η βαθμολογία είναι χαμηλότερη από 40%, τότε ίσως ο στόχος να ήταν πολύ δύσκολος. Επίσης, οι χαμηλές βαθμολογίες δεν είναι αποτυχίες αλλά ευκαιρίες μάθησης.