ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συστήματα Συνεχούς Ενσωμάτωσης: Βελτίωση Δικτύων 3ης Γενιάς και Τεχνολογία Λογισμικού

Σπυρίδων Δ. Δελβινιώτης

Επιβλέπων: **Λάζαρος Μεράκος,** Καθηγητής

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2016

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συστήματα Συνεχούς Ενσωμάτωσης: Βελτίωση Δικτύων 3ης Γενιάς και Τεχνολογία Λογισμικού

Σπυρίδων Δ. Δελβινιώτης

Α.Μ.: 1115201000037

ΕΠΙΒΛΕΠΩΝ: **Λάζαρος Μεράκος,** Καθηγητής

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια έχει παρατηρηθεί ραγδαία αύξηση των συνδρομητών κινητής τηλεφωνίας, αφού η χρήση των κινητών συσκευών εξαπλώνεται με ταχύτατους ρυθμούς. Πιο συγκεκριμένα, υπάρχει μια «έκρηξη» της κίνησης των δεδομένων από κινητούς χρήστες. Το γεγονός αυτό καθιστά αναγκαία την ανάπτυξη νέων δικτύων αλλά και τη βελτίωση των ήδη υπαρχόντων, προκειμένου να υποστηρίζουν αυξημένες δυνατότητες, ώστε να είναι δυνατή η καλύτερη εξυπηρέτηση των χρηστών.

Η παρούσα πτυχιακή εργασία εστιάζει στην βελτίωση των ήδη υπαρχόντων δικτύων τρίτης γενιάς. Πιο αναλυτικά, έχει ως στόχο την βελτίωση της ποιότητας του λογισμικού, που χρησιμοποιείται από ένα συγκεκριμένο συστατικό στοιχείο των δικτύων τρίτης γενιάς(3G), το SGSN. Ειδικότερα, το προϊόν λογισμικού που αναλύεται στην παρούσα εργασία είναι το “FlexiNS” της NOKIA. Επιπλέον, μελετάται ο τρόπος με τον οποίο η τεχνολογία λογισμικού μπορεί να συμβάλει στην αύξηση της ποιότητας του παραγόμενου λογισμικού καθώς και τη μείωση του κόστους παραγωγής και συντήρησής του. Όλα τα παραπάνω πραγματοποιήθηκαν επιτυχώς με την ένταξη της Συνεχούς Ενσωμάτωσης στα στάδια παραγωγής λογισμικού.

Ως προς την μεθοδολογία που ακολουθήθηκε, αρχικά γίνεται μία αναφορά στα δίκτυα κινητής τηλεφωνίας και στην τεχνολογία λογισμικού. Έπειτα, μελετώνται οι ήδη υπάρχουσες τεχνικές σχετικά με την ανάπτυξη λογισμικού και η θέση του SGSN στην αρχιτεκτονική δικτύων τρίτης γενιάς. Στη συνέχεια, παρουσιάζονται τα εργαλεία για την δημιουργία του CI συστήματος. Εν συνεχεία γίνεται εκτενής ανάλυση της δημιουργίας και λειτουργίας του CI συστήματος που χρησιμοποιήθηκε για τις ανάγκες του FlexiNS της NOKIA. Ύστερα, γίνεται ανάλυση των αποτελεσμάτων και των συμπερασμάτων που παρήχθησαν από την παραπάνω εφαρμογή. Τέλος, παρατίθενται ιδέες για μελλοντική μελέτη.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Δίκτυα κινητών επικοινωνιών, Τεχνολογία Λογισμικού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Συνεχής ενσωμάτωση, Ακραίος προγραμματισμός, Βελτίωση δικτύων τρίτης γενιάς, Κόμβος εξυπηρέτησης υποστήριξης GPRS

ABSTRACT

Recent years have witnessed rapid increase in mobile subscribers, since the use of mobile devices is spreading rapidly. More specifically, there has been an ' explosion ' of data traffic from mobile users. This fact necessitates the development of new networks and the improvement of existing ones, in order to support increased capabilities, so that the users can be served much better.
This thesis focuses on the improvement of already existing third-generation networks. Specifically, aims to improve the quality of software used by a specific component of third-generation networks (3G), the SGSN. More specifically, the software product that is tested is the "Flexi NS" of NOKIA. Furthermore, this thesis studies the way that software technology contributes to the increase of the quality of the software produced and the reduction of production costs and upkeep. Also, the role of the inclusion of Continuous Integration (CI) in the software production stages is examined.

As to the methodology followed, firstly we pinpoint the fundamentals of mobile networks and software engineering. Secondly, a study of the existing software development techniques and the position of the SGSN on third-generation network architecture follows. Then, we present the tools for the creation of a CI system. Subsequently, an extensive analysis of the creation and operation of the CI system used for the needs of NOKIA Flexi NS is presented. In addition, we include of an analysis of the results and conclusions, generated by the above application. Finally, ideas for future study are discussed.

SUBJECT AREA: Mobile communications Networks, Software Technology

KEYWORDS: Continuous integration, Extreme programming, Improvement of third generation of mobile networks, Serving GPRS support node

Η παρούσα πτυχιακή εργασία αφιερώνεται στην οικογένειά μου.

ΕΥΧΑΡΙΣΤΙΕΣ

Η εκπόνηση της εργασίας αυτής ήταν μακρά και περίπλοκη, περιλαμβάνοντας προσωπικές προκλήσεις. Αυτή η εμπειρία πυροδότησε συνεργασίες και φιλίες, με αποτέλεσμα να ευχαριστώ πολλούς ανθρώπους που συνέβαλλαν στην υλοποίηση αυτής της εργασίας.

Θα ήθελα να ευχαριστήσω θερμά τον υπεύθυνο καθηγητή μου κ. Λάζαρο Μεράκο για την υποστήριξή του και τις χρήσιμες συμβουλές που μου έδινε καθ' όλη την διάρκεια της δημιουργίας. Ήταν πολύ προσιτός στο να ακούσει τους προβληματισμούς μου και πάντα πρόθυμος να με συμβουλεύσει.

Τον Manager μου κ. Σπύρο Χρυσανθάκη για την υποστήριξή του και την πολύτιμη βοήθειά του κατά την διάρκεια της πρακτικής μου άσκησης στην ΝΟΚΙΑ αλλά και μετά από αυτήν. Υπήρξε μεγάλη έμπνευση για εμένα καθώς πάντα με ωθούσε στο να είμαι πιο παραγωγικός και αποτελεσματικός στην δουλειά μου.

Τους συνεργάτες μου στην ΝΟΚΙΑ, με τους οποίους μοιράστηκα τους φόβους μου και τους προβληματισμούς μου.

Ένα ιδιαίτερο ευχαριστώ στην οικογένειά μου και στους φίλους μου για την υπομονή τους και την υποστήριξή τους καθ' όλη την διάρκεια των σπουδών μου, καθώς και όλους εκείνους που συνέβαλαν με τον τρόπο τους στην επιτυχή ολοκλήρωση της προσπάθειάς μου.

Σπυρίδων Δελβινιώτης

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΠΡΟΛΟΓΟΣ

Η εργασία αυτή εκπονήθηκε στα πλαίσια της πτυχιακής εργασίας του Εθνικού Καποδιστριακού Πανεπιστημίου Αθηνών για το τμήμα Πληροφορικής και Τηλεπικοινωνιών. Η διεξαγωγή της έρευνας, που περιλαμβάνεται στην παρούσα εργασία, διενεργήθηκε στα πλαίσια της πρακτικής άσκησης στην εταιρεία NOKIA κατά τη χρονική περίοδο Μάϊος 2014 – Μάϊος 2015.

Η εκπόνηση της εργασίας αυτής ήταν μακρά και περίπλοκη, περιλαμβάνοντας προσωπικές προκλήσεις. Αυτή η εμπειρία πυροδότησε συνεργασίες και φιλίες, με αποτέλεσμα να ευχαριστώ πολλούς ανθρώπους που συνέβαλλαν στην υλοποίηση αυτής της εργασίας. Επιβλέποντες ήταν ο καθηγητής κ. Λάζαρος Μεράκος, στον οποίο οφείλω ιδιαίτερες ευχαριστίες.

Η παρούσα εργασία πραγματεύεται την ανάπτυξη και εφαρμογή του CI συστήματος, προκειμένου να διασφαλιστεί η ποιότητα του λογισμικού. Ο πρωταρχικός στόχος αυτής της εργασίας είναι η παρουσίαση, ανάπτυξη και εφαρμογή του συστήματος CI στον κόσμο των δικτύων κινητής τηλεφωνίας. Προκειμένου να επιτευχθεί ο παραπάνω στόχος θα χρησιμοποιηθούν στοιχεία από βιβλιογραφική έρευνα και από πρακτική έρευνα. Οι βασικές μέθοδοι που χρησιμοποιήθηκαν είναι πρακτική εφαρμογή σε συστήματα της NOKIA.

Αρχικά το θέμα θα παρουσιαστεί με βιβλιογραφική έρευνα η οποία θα περιλαμβάνει παρουσίαση των δικτύων κινητής τηλεφωνίας(δεύτερης και τρίτης γενιάς δίκτυα). Συγκεκριμένα, θα παρουσιαστεί το SGSN στα δίκτυα κινητής τηλεφωνία και θα παρουσιαστούν τα μοντέλα ανάπτυξης λογισμικού. Εν συνεχεία στα πλαίσια της πρακτικής έρευνας θα παρουσιαστεί και θα αναλυθεί το SGSN από άποψη υλικού και λογισμικού. Επίσης, θα αναλυθεί και το Continuous Integration (CI) σε γενικό επίπεδο αλλά και στα πλαίσια της NOKIA. Τέλος, θα παρουσιαστούν τα αποτελέσματα και συμπεράσματα της ερευνητικής εφαρμογής του μοντέλου αυτού μέσω γραφικής αναπαράστασης.

1. ΕΙΣΑΓΩΓΗ

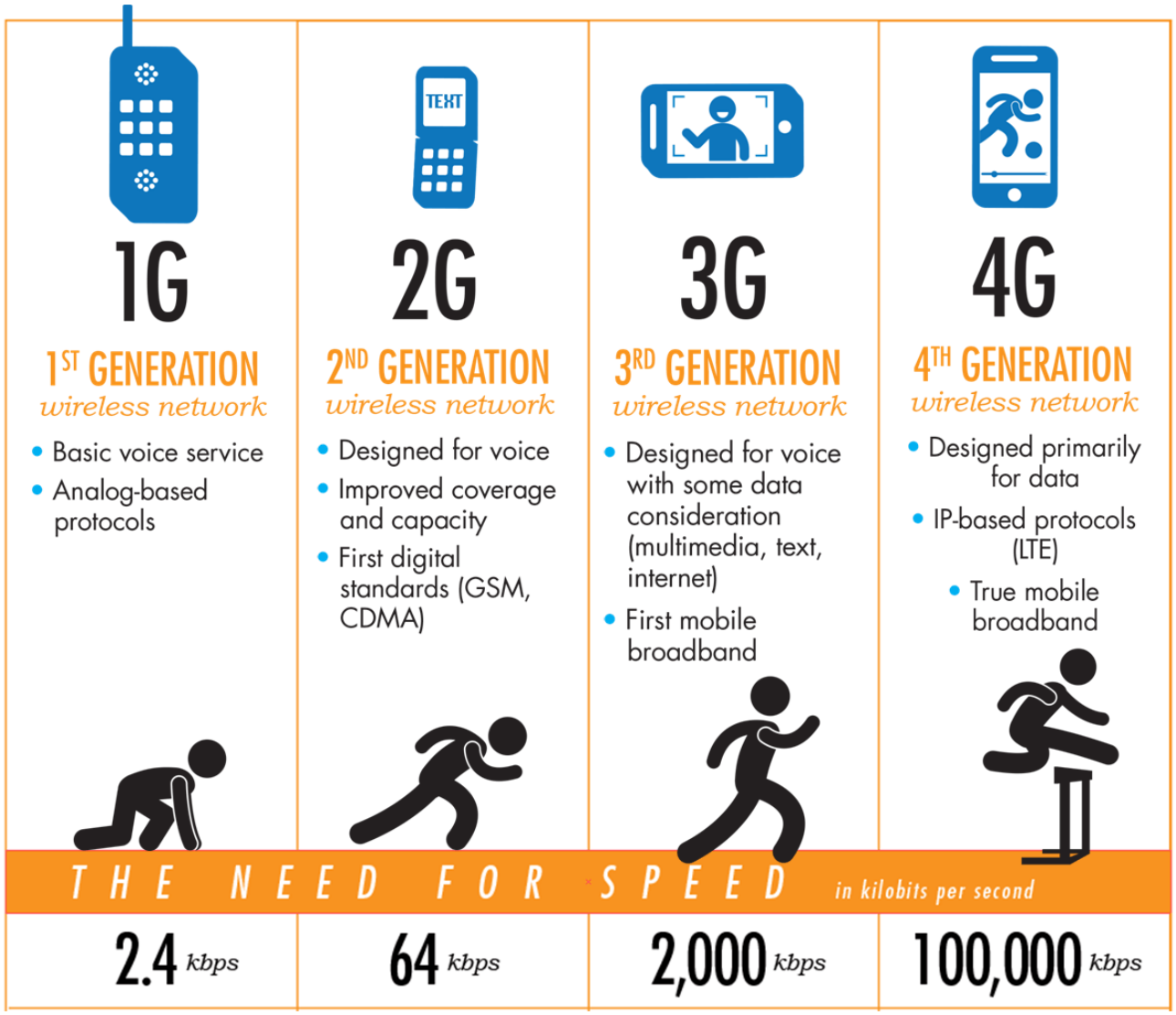
Σε αυτό το κεφάλαιο θα παρουσιαστούν ορισμένες βασικές πληροφορίες, οι οποίες είναι χρήσιμες για την πλήρη κατανόηση και την διασφάλιση της ομαλής ροής των δεδομένων της εργασίας. Αρχικά, θα αναλυθούν τα δίκτυα κινητής τηλεφωνίας δίνοντας έμφαση στην περιγραφή και ανάλυση των δικτύων δεύτερης και τρίτης γενιάς. Στη συνέχεια θα αναλυθεί η τεχνολογία λογισμικού και τα μοντέλα της. Τέλος, θα γίνει αναφορά στη βελτίωση της ποιότητας του λογισμικού, αναφέροντας κάποια βασικά δεδομένα για την βελτίωση των δικτύων κινητών και το Continuous Integration.

1.1 Δίκτυα Κινητής Τηλεφωνίας

Καθώς το κινητό τηλέφωνο έχει απαραίτητο ρόλο στη σύγχρονη καθημερινότητα της ζωής μας τα δίκτυα κινητής τηλεφωνίας εξελίσσονται συνεχώς με νέες τεχνολογίες, έτσι ώστε να προσφέρουν καινούριες υπηρεσίες και να καλύπτουν τις συνεχώς αυξανόμενες ανάγκες των χρηστών.

Η κινητή τηλεφωνία βασίζεται στα λεγόμενα *δίκτυα κυψέλης* (cellular networks), όπου η κυψέλη αποτελεί την βασική υποδιαίρεση ενός συστήματος κινητής τηλεφωνίας. Το μέγεθος κάθε κυψέλης είναι ανάλογο του συστήματος στο οποίο ανήκει. Στα παλαιότερα συστήματα χρησιμοποιούσαν μεγάλο εύρος περιοχής κυψέλης (10 έως 20 χιλιόμετρα) ενώ στα σημερινά συστήματα το εύρος κυψέλης είναι μεταξύ 0.8 και 8 χιλιομέτρων ανάλογα με την πυκνότητα πληθυσμού. Σε μια πυκνοκατοικημένη περιοχή απαιτείται μικρό εύρος ενώ σε μια λιγότερο πυκνοκατοικημένη το εύρος είναι μεγαλύτερο. Βασικός κανόνας είναι το σύνολο συχνοτήτων που χρησιμοποιείται από μια κυψέλη να είναι διαφορετικό από τις γειτονικές της κυψέλες.

Σε κάθε κυψέλη υπάρχει ο σταθμός βάσης (Base station), στον οποίο συνδέονται όλα τα τηλέφωνα της κυψέλης και αποτελείται συνήθως από έναν υπολογιστή και έναν πομποδέκτη συνδεδεμένο με μια κεραία. Για μικρές γεωγραφικές περιοχές υπάρχει η συσκευή Mobile Switching Center (MSC) πού πάνω της συνδέονται όλοι οι σταθμοί βάσης, ενώ για μεγαλύτερες περιοχές οι MSCs οργανώνονται σε πάνω από ένα επίπεδα. Στις παρακάτω ενότητες ακολουθεί η παρουσίαση της εξέλιξης της κινητής τηλεφωνίας από τα δίκτυα πρώτης γενιάς μέχρι και τα σημερινά δίκτυα τέταρτης και πέμπτης γενιάς.



Εικόνα 1: Η εξέλιξη των δικτύων κινητής τηλεφωνίας.

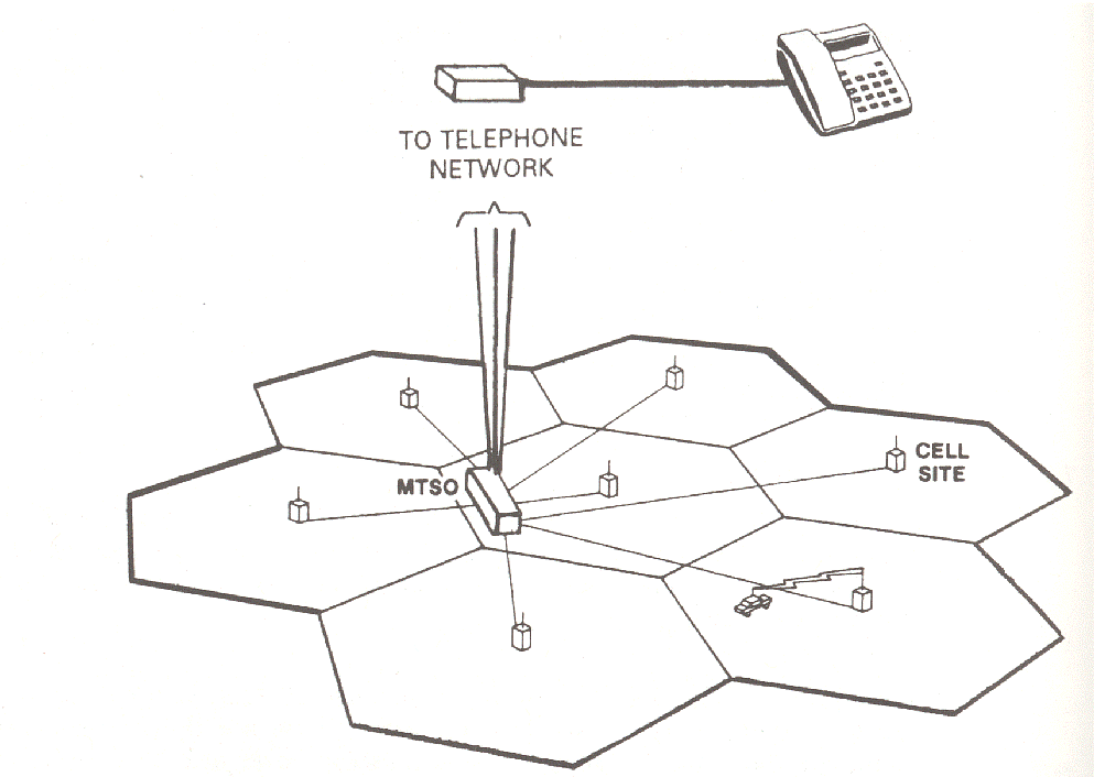
1.1.1 Ιστορική Αναδρομή

1.1.1.1 Πρώτης Γενιάς Δίκτυα (1G)

Τα δίκτυα κινητής τηλεφωνίας βασισμένα σε αναλογικά και όχι ψηφιακά ηλεκτρονικά είναι τα δίκτυα πρώτης γενιάς τα οποία επέτρεπαν μόνο τηλεφωνικές κλήσεις. Τέτοια δίκτυα ήταν τα AMPS, C-450, NMT και TACS. Σήμερα δεν υπάρχουν τέτοια δίκτυα καθώς δεν επέτρεπαν μετάδοση κειμένου ή δεδομένων αλλά έχουν αντικατασταθεί από τα δίκτυα δεύτερης γενιάς. Δίκτυα πρώτης γενιάς δεν υπήρξαν ποτέ στην χώρα μας.

Από τα αναλογικά συστήματα το Advanced Mobile Phone System (AMPS) χρησιμοποιήθηκε στην Αμερική αλλά και σε άλλες χώρες. Στις ΗΠΑ ήταν ευρέως διαδεδομένο (στο 80% των συνδρομητών κινητής) καλύπτοντας ολόκληρη την επικράτεια των ΗΠΑ. Με τροποποίηση του AMPS προέκυψε το Total Access Communication System (TACS), το οποίο

χρησιμοποιήθηκε στο Ηνωμένο Βασίλειο και λειτούργησε βασισμένο στο διαθέσιμο φάσμα συχνοτήτων των 800MHz και στην τεχνολογία πολυπλεξίας συχνότητας (Frequency Division Multiple Access – FMDA). Στα δίκτυα πρώτης γενιάς δεν ήταν δυνατή η κρυπτογράφηση συνομιλιών εξαιτίας της αναλογικότητας του σήματος πληροφορίας, το οποίο μπορούσε εύκολα να ανιχνευθεί με εργαλεία σάρωσης συχνοτήτων και να υποκλαπεί.[1]



Εικόνα 2: Το κυψελωτό δίκτυο AMPS.

1.1.1.2 Δεύτερης Γενιάς Δίκτυα (2G)

Σ' αυτά τα δίκτυα κινητής τηλεφωνίας παρέχεται εκτός από φωνητικές υπηρεσίες και αποστολή δεδομένων και φαξ, κυρίως λόγω του ότι βασίζονται στην ψηφιακή τεχνολογία του Global System for Mobile Communication (GSM) παρέχουν και πολλές άλλες υπηρεσίες. Ο σχεδιασμός του GSM βασίζεται στην χρήση της ζώνης συχνοτήτων των 900MHz που παραχωρήθηκε από την Διεθνή ένωση Τηλεπικοινωνιών (International Telecommunication Union - ITU). Στην Ελλάδα στα πρώτα δίκτυα κινητής τηλεφωνίας χρησιμοποιήθηκε αυτή η τεχνολογία GSM στις αρχές της δεκαετίας του 90. Διάφορες τεχνολογίες δικτύων δεύτερης γενιάς εφαρμόσθηκαν σε διάφορες χώρες με κύριο όμως χαρακτηριστικό τους χαμηλούς ρυθμούς μετάδοσης δεδομένων, περιορίζοντας έτσι το εύρος των προσφερόμενων υπηρεσιών. Για παράδειγμα στις ΗΠΑ ήταν βασισμένες στο TDMA, σε Ευρώπη, Αφρική και Ασία, χρησιμοποιήθηκε το GSM ενώ στην Ιαπωνία εφαρμόσθηκε το PDC. Στα συστήματα αυτά θα γίνει εκτενής αναφορά στη συνέχεια. Τα δίκτυα δεύτερης γενιάς περιορίζουν πάρα πολύ το πρόβλημα της υποκλοπής λόγω της κρυπτογράφησης των συνομιλιών, πράγμα

που δεν γινόταν στα δίκτυα πρώτης γενιάς. Ακόμη, στα δίκτυα 2G εμφανίζονται νέες υπηρεσίες, όπως αυτή της αποστολής σύντομων μηνυμάτων SMS (Short Message Service).

Το σύστημα GSM

Στο τέλος της δεκαετίας του 80 υπήρξε μια πανευρωπαϊκή προσπάθεια για την δημιουργία ενός προτύπου συστήματος κινητής τηλεφωνίας. Ανάλογα με τον τρόπο πραγματοποίησής του χρησιμοποιεί ζώνες συχνοτήτων στα 900, 1800 και 1900MHz. Κάθε ζώνη συχνοτήτων υποδιαιρείται σε δύο περιοχές συχνοτήτων για αποστολή και λήψη δεδομένων, ενώ κάθε περιοχή συχνοτήτων υποδιαιρείται επίσης σε κανάλια εύρους 200KHz τα οποία μέσω πολυπλεξίας συχνότητας (Frequency Division Multiple Access - FDMA) μπορούν να εξυπηρετούν μεγάλο αριθμό συνδρομητών. Κάθε κανάλι είναι διαθέσιμο στον χρήστη μέσα σε αυστηρά καθορισμένη χρονική περίοδο με διακεκομμένη εκπομπή και λήψη δεδομένων μη αντιληπτή από τον χρήστη. Είναι δυνατή η χρήση από πολλούς χρήστες του ίδιου καναλιού ταυτόχρονα (μέχρι και 8 χρήστες) μέσω της πολυπλεξίας χρόνου (Time Division Multiple Access - TDMA). Το GSM μειονεκτεί στον ρυθμό μετάδοσης δεδομένων και επομένως δεν μπορεί να μεταδώσει μεγάλο όγκο πληροφορίας.

Τα συστήματα PDC (Personal Digital Cellular) και NADC (North American Digital Cellular)

Από τα δύο αυτά συστήματα το πρώτο χρησιμοποιείται αποκλειστικά στην Ιαπωνία ενώ το δεύτερο κυρίως στην Βόρεια Αμερική. Τα δύο συστήματα χρησιμοποιούν την ίδια διαφορική π/4 διαμόρφωση φάσης, ενώ αποτελούν επεκτάσεις του αναλογικού δικτύου που υπάρχει σε κάθε χώρα. Η δομή τους είναι κυψελώδης και οι τεχνικές πολυπλεξίας που εφαρμόζουν μοιάζουν με αυτή του GSM. Οι τηλεφωνικές συσκευές που συνεργάζονται με αυτά τα συστήματα είναι συμβατές και με τα αναλογικά δίκτυα για τους εξής λόγους: α) Το PDC είναι συμβατό με αναλογικά δίκτυα αλλά αποτελεί αυτόνομο δίκτυο και, β) το NADC περιέχει αρκετά μέρη των αναλογικών δικτύων 1ης γενιάς AMPS.

Το σύστημα CDMA (Code Division Multiple Access)

Αυτό το σύστημα εισάχθηκε σαν πρότυπο το 1993 και χρησιμοποιείται στις ΗΠΑ με μάλλον μικρό αριθμό συνδρομητών κινητής τηλεφωνίας παγκοσμίως. Διαφέρει από τις προηγούμενες τεχνολογίες δεύτερης γενιάς κατά το ότι βασίζεται σε τεχνική πολυπλεξίας διαίρεσης κώδικα, ενώ το κανάλι που χρησιμοποιούν οι χρήστες είναι ενιαίο με εύρος 1.23MHz. Το σύστημα CDMA παρόλο που παρέχει τις ίδιες υπηρεσίες περίπου με το GSM δεν μπορεί να το ανταγωνιστεί. Διάφορες τεχνολογίες που βελτιώνουν την απόδοσή του (CDMA-2000) το καθιστούν ικανό να παρέχει και υπηρεσίες τρίτης γενιάς.

1.1.1.3 Μεταβατικά Δίκτυα (2.5G)

Κατά την διαδικασία μετάβασης στην τρίτη γενιά δημιουργούνται μεταβατικές τεχνολογίες, οι οποίες βοηθούν στο να γίνει ομαλή αυτή η μετάβαση μέσω της απόκτησης κατάλληλης τεχνογνωσίας. Αυτές οι τεχνολογίες βασίζονται στο GSM και ονομάζονται 2.5G ή 2G++. Κάποιες από αυτές είναι οι GPRS και HSCSD, όπου το GSM γίνεται αποδοτικότερο στη μεταφορά δεδομένων, ενώ έχουν δυνατότητες παροχής νέων υπηρεσιών προς τους χρήστες.

Το HSCSD (High-Speed Circuit-Switched Data) σύστημα

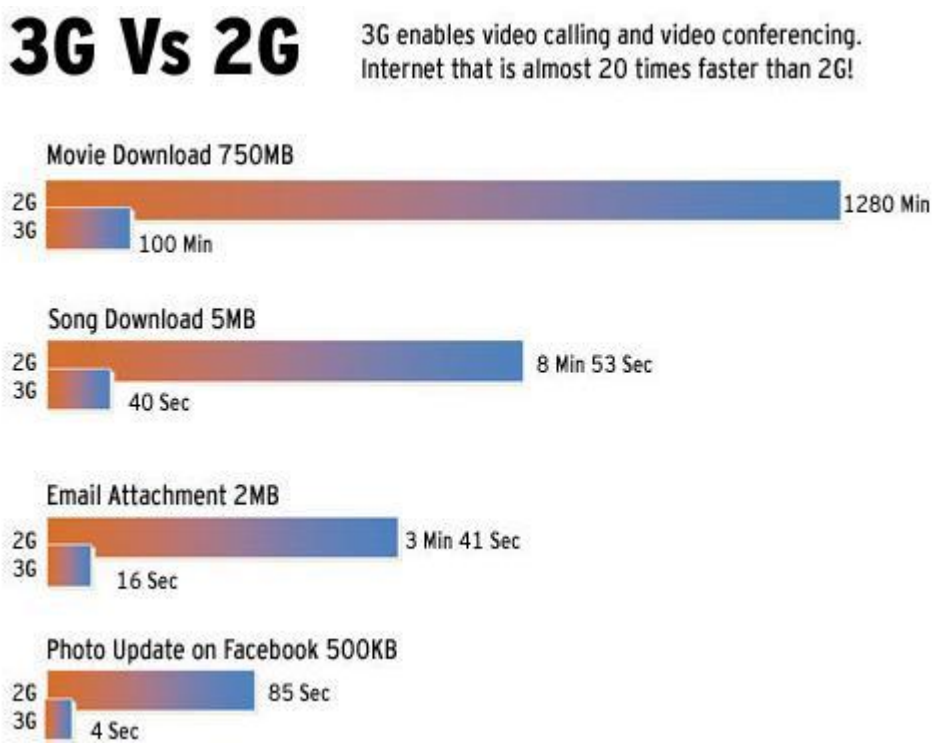
Το HSCSD βελτιώνει το GSM καθώς αυξάνει την ταχύτητα μεταφοράς δεδομένων παρεμβαίνοντας στον χρόνο που διαθέτει στον χρήστη. Πιο συγκεκριμένα, ενώ ένα αναλογικό μόντεμ αποδίδει 44-48Kbps ροή δεδομένων κατά μέγιστο και μια γραμμή ISDN 64-128Kbps, οι ρυθμοί μεταφοράς δεδομένων στο GSM περιορίζονται στα 9.6Kbps. Ο λόγος είναι ότι εξαιτίας της τεχνικής πολυπλεξίας στον χρόνο και της πολυπλεξίας στη συχνότητα που εφαρμόζονται η ροή δεδομένων γίνεται κατά ορισμένα χρονικά διαστήματα που λέγονται χρονοθυρίδες. Μέσω του συστήματος HSCSD κάθε χρήστης μπορεί να καταλάβει τρείς χρονοθυρίδες αντί μία, δηλαδή, περισσότερο χρόνο στην χρήση του καναλιού. Επομένως, η ταχύτητα αυξάνεται σε 3x9.6=28.8Kbps με αποτέλεσμα την βελτίωση της ποιότητας των υπηρεσιών με κόστος την μείωση της διαθεσιμότητας των καναλιών. Κατ' αυτόν τον τρόπο, το κόστος για τους συνδρομητές είναι υψηλό, γι’ αυτό απευθύνεται κυρίως σε επαγγελματίες, ενώ δεν παρέχει μόνιμη σύνδεση με το δίκτυο.

Το GPRS (General Packet Radio Service) σύστημα

Βασικό χαρακτηριστικό αυτού του συστήματος μετάδοσης δεδομένων αποτελεί η μεταγωγή πακέτων δεδομένων μόνο κατά την αποστολή ή λήψη δεδομένων από τον χρήστη, με χρέωση του όγκου πληροφορίας που διακινήθηκε και εξασφαλίζοντας την μόνιμη σύνδεση των κινητών συσκευών με το δίκτυο. Ειδικότερα, κατά τη χρήση του κινητού τηλεφώνου π.χ. για το διάβασμα μιας σελίδας στο διαδίκτυο, κατά τον κλασσικό τρόπο μεταγωγής απασχολείται το δίκτυο τηλεφωνίας συνεχώς όσο είμαστε συνδεδεμένοι, κάνοντας αποκλειστική χρήση του καναλιού με την μεταγωγή του κατάλληλου κυκλώματος, Σε αυτή τη περίπτωση η χρέωση βασίζεται στο συνολικό χρόνο που απασχολείται το δίκτυο τηλεφωνίας. Χρησιμοποιώντας την μεταγωγή πακέτων πληροφορίας μόνο κατά τη λήψη και αποστολή δεδομένων, το κανάλι επικοινωνίας αποδεσμεύεται και μπορεί να δοθεί σε άλλον χρήστη κάνοντας έτσι πιο ορθολογική τη χρήση των πόρων και αλλάζοντας την χρέωση ανάλογα με την ποσότητα πληροφορίας. Το GPRS προσφέρει μεγαλύτερες ταχύτητες στην επικοινωνία και αυξάνει τη δυνατότητα παροχής νέων υπηρεσιών. Λόγω των υψηλών ταχυτήτων μετάδοσης αρχίζει να εφαρμόζεται και το νέο πρωτόκολλο WAP (Wireless Application Protocol) ενώ επιτρέπεται ο εμπλουτισμός των πληροφοριών και βελτιώνεται η εμπειρία χρήσης. Το GPRS είναι ιδανικό για εφαρμογές διαδικτύου, αποστολή - λήψη ηλεκτρονικού ταχυδρομείου, πλοήγηση στο διαδίκτυο και εφαρμογές ομάδων συζητήσεων (Chat).

1.1.1.4 Τρίτης Γενιάς Δίκτυα (3G)

Συστήματα όπως τα WCDMA, UMTS, CDMA2000 και EDGE ή, όπως συνήθως λέγεται 2.75G ανήκουν στην τεχνολογία δικτύων τρίτης γενιάς, τα οποία έχοντας αυξημένες ταχύτητες μεταφοράς δεδομένων προσφέρουν νέες υπηρεσίες και αυξημένες δυνατότητες επικοινωνίας. Μία πασίγνωστη εφαρμογή των δικτύων 3G είναι το λεγόμενο mobile internet το οποίο αποτελεί σημαντικό παράδειγμα σύγκλισης ανάμεσα σε υπολογιστές και τηλεπικοινωνίες. Δια μέσου των δικτύων 3G είναι δυνατή η εκτέλεση εργασιών και η πρόσβαση σε δεδομένα από οποιαδήποτε γεωγραφική θέση που καλύπτεται από το δίκτυο 3G.



Εικόνα 3: Σύγκριση δικτύων δεύτερης και τρίτης γενιάς.[2]

Οι υπηρεσίες επικοινωνίας με βίντεο κλήσεις ή οι τηλεδιασκέψεις γίνονται εύκολα μέσα από τα δίκτυα τρίτης γενιάς, ενώ ταυτόχρονα μπορεί να σταλούν μηνύματα κειμένου με φωτογραφίες, βίντεο και ήχο. Παράλληλα, η χρονική διάρκεια αποστολής τους είναι κατά πολύ μικρότερη από ότι στα δίκτυα 2G. Λόγω του αυξημένου εύρους ζώνης μπορεί να γίνει μετάδοση βίντεο σε πραγματικό χρόνο ώστε να είναι δυνατή η παρακολούθηση ζωντανών τηλεοπτικών προγραμμάτων. Μπορούν ακόμα οι χρήστες να παίζουν σε πραγματικό χρόνο παιχνίδια ανεξάρτητα από την γεωγραφική τοποθεσία, αρκεί να καλύπτεται από το δίκτυο.

Εν συνεχεία, θα παρουσιαστούν δύο διαφορετικής τεχνολογίας συστήματα ώστε να γίνει αντιληπτή η εξέλιξη των δικτύων σε τρίτης γενιάς δίκτυα: το WCDMA και το EDGE. Το πρώτο αφορά μια τεχνολογία ευρυζωνικής μετάδοσης δεδομένων (Wideband Code Division Multiple Access - WCDMA) τέτοια ώστε να υποστηρίζει υπηρεσίες Internet, Multimedia, Video και γενικά απαιτητικές υπηρεσίες σε όγκο ροής πληροφορίας. Το WCDMA εγκρίθηκε από το European Telecommunications Standards Institute (ETSI), το 1998 και μπορεί να έχει ρυθμούς μεταφοράς μέχρι και 384Kbits/sec για όλη την περιοχή κάλυψης, ενώ τοπικά μπορεί να φτάσει και μέχρι τα 2Mbits/sic. Με άλλα λόγια το WCDMA είναι ταχύτερο από το GSM περίπου 40 φορές.

Η τεχνολογία πολυπλεξίας στο GSM βασίζεται στην διαίρεση κώδικα διαφορετικά από την πολυπλεξία διαίρεσης χρόνου και συχνότητας στην οποία βασίζεται το WCDMA. Όταν ένα κινητό εκπέμψει στο GSM τότε αποκαθίσταται ένα είδος επικοινωνίας μεταξύ του κινητού και του σταθμού βάσης και ορίζεται ένα κανάλι και μία χρονοθυρίδα (timeslot) ώστε ο πομπός να γνωρίζει πού και πότε θα εκπέμψει. Τα συστήματα που εφαρμόζουν πολυπλεξία κώδικα (CDMA, CDMA-2000 και WCDMA) είναι περισσότερο πολύπλοκα, αλλά έχουν μεγάλα πλεονεκτήματα και γι’ αυτό αναπτύχθηκαν κατά πρώτον για στρατιωτικές εφαρμογές.

Στο WCDMA το ίδιο κανάλι χρησιμοποιείται ταυτόχρονα από ομάδες χρηστών με πολλαπλάσιο εύρος ζώνης από αυτό στο GSM. Γι' αυτό τον λόγο, ο χρήστης λαμβάνει ένα μοναδικό κωδικό (δηλαδή μια ακολουθία δυαδικών ψηφίων) σε μια συγκεκριμένη γεωγραφική περιοχή. Αυτός ο κωδικός μαζί με τα δεδομένα εισάγεται σε μια συνάρτηση που παράγει σαν έξοδο τα δεδομένα προς μετάδοση, διαμορφωμένα έτσι ώστε το φάσμα τους να απλώνεται σε ένα κανάλι μεγάλου εύρους ζώνης. Το εύρος ζώνης ενός WCDMA συστήματος είναι 5MHz ενώ ενός GSM συστήματος είναι 200KHz.

Στον δέκτη τα διάφορα σήματα του καναλιού διαχωρίζονται με βάση τον κωδικό, που χρησιμοποιήθηκε στην εκπομπή των δεδομένων, από τα υπόλοιπα σήματα. Το WCDMA είναι ιδιαίτερα ανθεκτικό σε παρεμβολές και διαλείψεις, αντίθετα από άλλα συστήματα όπως το GSM, το οποίο απαιτεί πολύ προσεκτικό σχεδιασμό του δικτύου. Επειδή όλοι οι χρήστες χρησιμοποιούν το ίδιο κανάλι η επικοινωνία είναι πιο σταθερή και απαλλαγμένη από την πολυπλοκότητα της πολυπλεξίας χρόνου και συχνότητας.

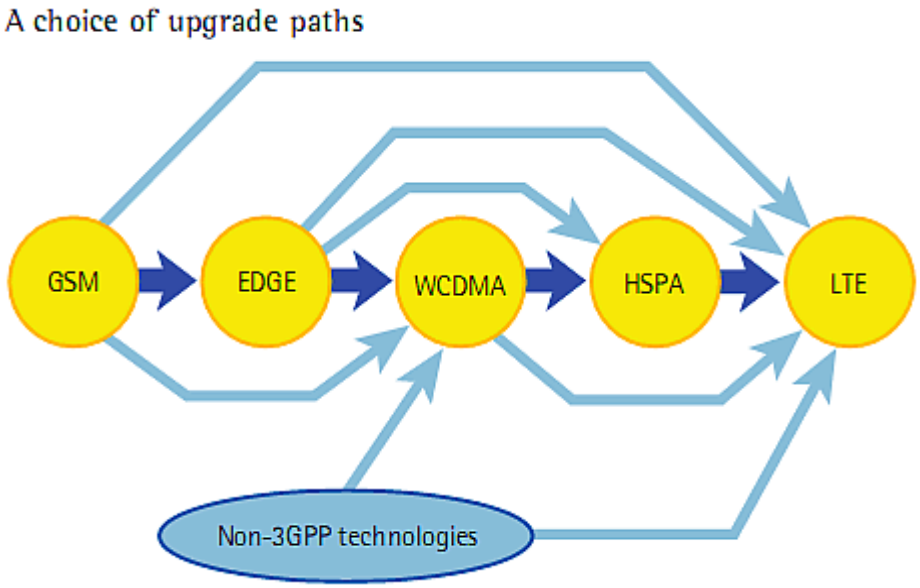
Το σύστημα EDGE (Enhanced Data rates for GSM)

Με το σύστημα EDGE τα υπάρχοντα δίκτυα τηλεφωνίας δεύτερης γενιάς μπορούν να αποκτήσουν την απαραίτητη ταχύτητα και χωρητικότητα ώστε να χρησιμοποιηθούν για παροχή υπηρεσιών, οι οποίες χαρακτηρίζουν τα δίκτυα τρίτης γενιάς. Πιο αναλυτικά, με το EDGE μπορούν να εξυπηρετήσουν τρείς φορές περισσότερους συνδρομητές από ότι στο GPRS με τριπλάσια ταχύτητα μεταφοράς δεδομένων ή απλά να απελευθερώσουν χώρο για την βελτίωση των φωνητικών υπηρεσιών. Έτσι, καθίστανται εφικτές υπηρεσίες όπως, η δρομολόγηση βίντεο (video streaming), η αναζήτηση στο διαδίκτυο (Internet browsing) ή η τηλεδιάσκεψη. Επειδή το EDGE είναι ουσιαστικά το GPRS αναβαθμισμένο δεν μπορεί να λειτουργήσει αυτόνομα μέσα σε κάποιο δίκτυο GSM. Λόγω των ιδιαίτερα αποδοτικών σημάτων διαμόρφωσης που χρησιμοποιεί μπορεί να φτάσει σε ταχύτητες έως και 384Kbits/sec. Είναι πιο εύκολο από πρακτική άποψη να λειτουργήσει ένα δίκτυο EDGE βασισμένο πάνω σε υποδομή συστήματος GSM παρά να εγκατασταθεί ένα δίκτυο WCDMA το οποίο κοστίζει πολύ περισσότερο. Επίσης, η εγκατάσταση είναι εύκολη και λόγω του ότι χρειάζεται αφενός μεν απλή αναβάθμιση του λογισμικού αφετέρου δε του υλικού μέσω αναβάθμισης των σταθμών βάσεων, πράγμα που δεν είναι ιδιαίτερα δύσκολο.

1.1.1.5 Τέταρτης Γενιάς Δίκτυα (4G)

Καθώς τα δίκτυα κινητής τηλεφωνίας δεν σταμάτησαν να εξελίσσονται, ήταν επόμενο τα δίκτυα 3G να έδιναν τη θέση τους στα δίκτυα τέταρτης γενιάς. Όμως, όπως και μεταξύ δεύτερης και τρίτης γενιάς παρεμβλήθηκαν τα μεταβατικά δίκτυα 2.5G έτσι και μετά τα δίκτυα 3G εμφανίζονται τα δίκτυα Pre-4G. Πιο συγκεκριμένα, τον Μάρτιο 2008, η International Telecommunications Union-Radio sector (ITU-R) καθόρισε ένα σύνολο χαρακτηριστικών, τα International Mobile Telecommunications Advanced Specification (ITU-R spec) τα οποία θα χαρακτήριζαν τα δίκτυα τέταρτης γενιάς. Το πιο σημαντικό στοιχείο αυτών των δικτύων θα ήταν να παρέχουν σε κινούμενους χρήστες με μεγάλη ταχύτητα (μέσα σε αυτοκίνητα, τραίνα, κ.λ.π.), ταχύτητες ροής δεδομένων μέχρι 100Mbits/sec και σε κινούμενους με μικρή ταχύτητα χρήστες, ροές δεδομένων μέχρι 1Gbits/sec. Τον Σεπτέμβρη 2008, παρουσιάστηκε στις ΗΠΑ το σύστημα WiMAX από την Sprint Nextel, το οποίο αν και δεν ικανοποιούσε όλα τα standards της ITU-R, spec παρουσιάστηκε ως σύστημα 4G. Λίγο αργότερα (2009) εμφανίστηκε το σύστημα LTE, το οποίο και αυτό δεν ήταν τυπικό δίκτυο 4G, παρόλο που

παρουσιάστηκε ως τέτοιο στην αγορά. Είναι λοιπόν κατανοητό πως για τους ανωτέρω λόγους γίνεται αναφορά σε αυτά τα συστήματα σαν τεχνολογίες Pre-4G. [3]



Εικόνα 4: Μετάβαση στα δίκτυα τέταρτης γενιάς. [3]

Το σύστημα Mobile WiMAX (IEEE 802.16e)

Αυτό το σύστημα προέρχεται από το WiMAX forum (2001) με βασικό στόχο εκτός από την αναβάθμιση των υπαρχόντων δικτύων για μεγαλύτερες ταχύτητες κατά την επικοινωνία, αλλά και να προσφέρει εναλλακτικό τρόπο πρόσβασης στο διαδίκτυο ως προς το DSL. Η εταιρία Sprint Nextel παρουσίασε ένα σύστημα WiMAX το 2008 ενώ ένα ανάλογο σύστημα είχε κατασκευαστεί στην Νότιο Κορέα δύο χρόνια πριν. Η τεχνολογία WiMAX παρέχει υψηλότερες ταχύτητες σχετικά με την 3G, οι οποίες φτάνουν τα 128Mbits/sec και 56Mbits/sec στο κατέβασμα και ανέβασμα δεδομένων αντίστοιχα. Αυτό επιτυγχάνεται μέσω καναλιών εύρους ζώνης 20MHz. Ύστερα από την αναβάθμισή του (WiMAX-2011) οι αναμενόμενες ταχύτητες θα μπορούσαν να φτάσουν το 1Gbits/sec για ακίνητους χρήστες. Το πρώτο κινητό τηλέφωνο με την τεχνολογία WiMAX ήταν της HTC και κυκλοφόρησε πρώτα στη Ρωσία (MAX 4G) και δύο χρόνια μετά σε όλο τον κόσμο (EVO 4G).

Το σύστημα 3GPP Long Term Evolution - LTE

Το LTE προέρχεται από τις τεχνολογίες GSM και EDGE με κατάλληλη αναβάθμιση ώστε να παρέχουν μεγαλύτερες ταχύτητες δεδομένων. Είναι σχεδιασμένο για να παρέχει ρυθμούς μεταφοράς δεδομένων της τάξης των 300 Mbps στην καθοδική ζεύξη (downlink) και στην ανοδική (uplink) μέχρι 75 Mbps. Το εύρος ζώνης του φέροντος σήματος είναι μεταβλητό, κυμαινόμενο από τα 1.4 έως τα 20 MHz και υποστηρίζονται τόσο η διπλεξία διαίρεσης συχνότητας (FDD) όσο και η διπλεξία διαίρεσης χρόνου (TDD). Η αρχιτεκτονική του δικτύου βασίζεται σε μια απλοποιημένη μορφή αρχιτεκτονικής IP, το Evolved Packet Core (EPC), το οποίο σχεδιάστηκε για να αντικαταστήσει το GPRS Core Network και υποστηρίζει την απρόσκοπτη μετάδοση τόσο δεδομένων όσο και φωνής ακόμα και σε δίκτυα με παλαιότερη

τεχνολογία δικτύου (GSM, UMTS, CDMA2000). Η απλούστερη αρχιτεκτονική αποσκοπεί σε χαμηλότερα λειτουργικά έξοδα. [4]

1.1.2 Αρχιτεκτονική δικτύων τρίτης γενιάς (3G)

Η αρχιτεκτονική UMTS 3G υποχρεούται να παρέχει ένα μεγαλύτερο επίπεδο απόδοσης συγκριτικά με το αρχικό δίκτυο GSM. Ωστόσο, δεδομένου ότι πολλά δίκτυα έχουν μετεξελιχθεί με τη χρήση του GPRS και EDGE, είχαν ήδη τη δυνατότητα να μεταφέρουν δεδομένα. Κατά συνέπεια, πολλά από τα στοιχεία που απαιτούνται για την αρχιτεκτονική του δικτύου WCDMA / UMTS θεωρήθηκαν μεταβατικά. Αυτό μείωσε σημαντικά το κόστος υλοποίησης του δικτύου UMTS, καθώς πολλά στοιχεία ήταν στη θέση τους ή χρειάζονταν απλά αναβάθμιση.

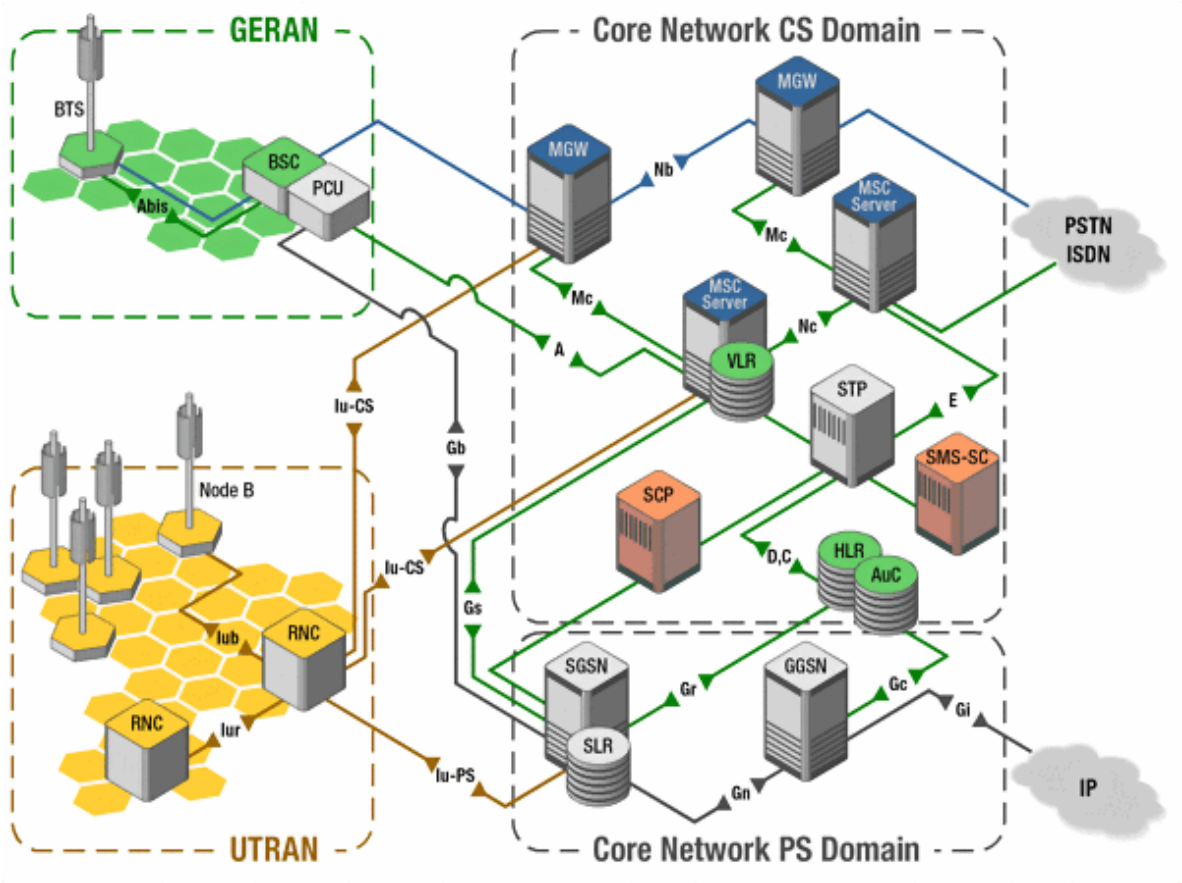
Ένας από τους κύριους στόχους του UMTS είναι να επιτρέπει την μεταφορά δεδομένων. Η αρχιτεκτονική του δικτύου UMTS σχεδιάστηκε για να επιτρέψει μια σημαντική βελτίωση της απόδοσης του ρυθμού λήψης/αποστολής δεδομένων συγκριτικά με το GSM.

1.1.2.1 Συστατικά δικτύων τρίτης γενιάς (3G) - UMTS

Η αρχιτεκτονική του δικτύου UMTS μπορεί να διαιρεθεί σε τρία κύρια στοιχεία:

- *Εξοπλισμός Χρήστη* (UE): Ο Εξοπλισμός Χρήστη, ή αλλιώς User Equipment, είναι το κινητό τηλέφωνο.
- *Radio Network Subsystem* (RNS): Το RNS είναι επίσης γνωστό και ως UMTS Radio Network Access, UTRAN. Είναι ισοδύναμο του Υποσυστήματος Σταθμού Βάσης , ή αλλιώς Base Station Subsystem (BSS), για το GSM. Το RNS παρέχει και διαχειρίζεται τη διεπαφή αέρα στο συνολικό δίκτυο.
- *Πυρήνας Δικτύου* (CN): Ο πυρήνας του δικτύου παρέχει όλη την κεντρική επεξεργασία και διαχείριση του δικτύου. Είναι ισοδύναμο της Μεταγωγής Υποσυστήματος Δικτύου GSM, ή αλλιώς Network Switching Subsystem (NSS).

Συνεπώς, ο πυρήνας του δικτύου (CN) είναι η συνολική οντότητα που διασυνδέεται σε εξωτερικά δίκτυα, συμπεριλαμβανομένου του δημόσιου τηλεφωνικού δικτύου και άλλων κυψελοειδών δικτύων τηλεπικοινωνιών.[6] Παρακάτω ακολουθεί σχηματική αναπαράσταση της αρχιτεκτονικής του δικτύου UMTS. Στη συνέχεια παρουσιάζεται συνοπτική ανάλυση των παραπάνω κύριων συστατικών στοιχείων.



Εικόνα 5: Η αρχιτεκτονική δικτύων τρίτης γενιάς. [5]

Εξοπλισμός Χρήστη (UE)

Ο Εξοπλισμός Χρήστη (UE), είναι ένα σημαντικό στοιχείο της συνολικής αρχιτεκτονικής του δικτύου 3G UMTS. Αποτελεί την τελική διασύνδεση με το χρήστη. Λαμβάνοντας υπόψιν τον κατά πολύ μεγαλύτερο αριθμό εφαρμογών και εγκαταστάσεων που μπορούν να εκτελεστούν, πάρθηκε η απόφαση να ονομαστεί *Εξοπλισμός Χρήστη* και όχι Κινητό. Ουσιαστικά, όμως, είναι η συσκευή κινητού τηλεφώνου (με την ευρύτερη ορολογία). Έχει πρόσβαση σε πολύ υψηλότερες ταχύτητες για επικοινωνία μέσω δεδομένων. Επίσης, μπορεί να είναι πολύ πιο ευέλικτο περιέχοντας πολλές άλλες εφαρμογές. Αποτελείται από μια ποικιλία διαφορετικών στοιχείων, όπως κύκλωμα RF, επεξεργαστές, κεραία, μπαταρία, κλπ. [6]

Υπάρχει ένας αριθμός από στοιχεία εντός του Εξοπλισμού Χρήστη που μπορούν να περιγράφουν ξεχωριστά:

1. *Κύκλωμα RF*: Οι περιοχές RF χειρίζονται όλα τα στοιχεία σημάτων, τόσο για το δέκτη όσο και για τον πομπό. Μία από τις μεγαλύτερες προκλήσεις για τον ενισχυτή ισχύος RF ήταν να μειωθεί η κατανάλωση ενέργειας. Η μορφή της διαμόρφωσης που χρησιμοποιείται για το W-CDMA απαιτεί τη χρήση ενός γραμμικού ενισχυτή. Οι γραμμικοί ενισχυτές εγγενώς καταναλώνουν περισσότερο ρεύμα συγκριτικά με τους μη γραμμικούς ενισχυτές (που χρησιμοποιούνται για τη μορφή της διαμόρφωσης που

χρησιμοποιείται για το GSM). Κατά συνέπεια ήταν απαραίτητο να διατηρηθεί η διάρκεια ζωής της μπαταρίας και να εξασφαλιστεί η βέλτιστη απόδοση.

2. *Επεξεργασία Βασικής Ζώνης*: Η επεξεργασία του σήματος βασικής ζώνης, ή αλλιώς Baseband processing, αποτελείται κυρίως από ψηφιακά κυκλώματα. Τα κυκλώματα αυτά είναι πολύ πιο περίπλοκα από τα κυκλώματα που χρησιμοποιήθηκαν στα κινητά τηλέφωνα για τις προηγούμενες γενιές. Επίσης, τα ψηφιακά κυκλώματα έχουν βελτιστοποιηθεί για να μειωθεί η κατανάλωση ρεύματος όσο το δυνατόν περισσότερο.

3. *Μπαταρία*: Ενώ η κατανάλωση ρεύματος έχει ελαχιστοποιηθεί όσο το δυνατόν περισσότερο μέσα στο κύκλωμα του τηλεφώνου, υπήρξε μια αύξηση στην κατανάλωση ρεύματος στην μπαταρία. Με τους χρήστες να αναμένουν την ίδια διάρκεια ζωής των μπαταριών όπως σε κινητά τηλέφωνα προηγούμενων γενιών, καθίσταται αναγκαία η χρήση νέων και βελτιωμένων τεχνολογιών. Για την επίτευξη αυτού του στόχου, χρησιμοποιούνται μπαταρίες ιόντων λιθίου (Li-ion). Κατ' αυτόν τον τρόπο, τα κινητά τηλέφωνα παραμένουν μικρά και σχετικά ελαφριά, ενώ παράλληλα διατηρούν ή ακόμα και βελτιώνουν τη συνολική διάρκεια ζωής της μπαταρίας μεταξύ των φορτίσεων.

4. *Universal Subscriber Identity Module*, USIM: Ο Εξοπλισμός Χρήστη περιλαμβάνει επίσης μια κάρτα SIM, η οποία στην περίπτωση του UMTS ονομάζεται USIM (Universal Subscriber Identity Module). Είναι μια πιο προηγμένη εκδοχή της κάρτας SIM που χρησιμοποιείται σε δίκτυα GSM και σε άλλα συστήματα, αλλά ενσωματώνει τους ίδιους τύπους πληροφοριών. Περιέχει τον αριθμό International Mobile Subscriber Identity (IMSI), καθώς και ο κινητός σταθμός Διεθνής Αριθμός ISDN (MSISDN). Τέλος, η USIM περιέχει μια μικρή μνήμη η οποία επιτρέπει την αποθήκευση μηνυμάτων και τηλεφωνικών αριθμών.

Radio Network Subsystem (RNS)

Το Radio Network Subsystem (RNS), ή αλλιώς UMTS Radio Network Access (UTRAN) περιλαμβάνει δύο κύρια συστατικά:

1. *Radio Network Controller* (RNC): Το RNC αναλαμβάνει τη διαχείριση ραδιοφωνικών πόρων και μερικές από τις λειτουργίες διαχείρισης της κινητικότητας, αλλά όχι όλες. Είναι επίσης το σημείο στο οποίο πραγματοποιείται η κρυπτογράφηση/αποκρυπτογράφηση για την προστασία των δεδομένων του χρήστη από υποκλοπές.

2. *Κόμβος Β*: Ο Κόμβος Β, ή αλλιώς Node B, είναι ο όρος που χρησιμοποιείται στο UMTS για να υποδηλωθεί ο πομποδέκτης σταθμού βάσης. Αυτό το τμήμα του UTRAN περιέχει τον πομπό και δέκτη για να επικοινωνούν με τα UEs εντός της κυψέλης. Επιπλέον, συμμετέχει με το RNC στη διαχείριση των πόρων. [6]

Πυρήνας Δικτύου (CN)

Ο Πυρήνας Δικτύου, ή αλλιώς Core Network (CN), παρέχει όλη την κεντρική επεξεργασία και διαχείριση του δικτύου. Είναι ισοδύναμο της Μεταγωγής Υποσυστήματος Δικτύου GSM, ή αλλιώς Network Switching Subsystem (NSS). Λόγω των διαφορετικών τρόπων με τους οποίους μπορούν να μεταφέρονται δεδομένα, ο Πυρήνας Δίκτυο μπορεί να χωριστεί σε δύο διαφορετικές περιοχές: [8]

1. *Κύκλωμα Μεταγωγής Στοιχείων* (Circuit switched elements): Αυτά τα στοιχεία βασίζονται κυρίως στις οντότητες του δικτύου GSM και μεταφέρουν δεδομένα μέσω μεταγωγής κυκλώματος, δηλαδή ένα μόνιμο κανάλι για ολόκληρη τη διάρκεια μιας κλήσης.

2. *Στοιχεία Μεταγωγής Πακέτων* (Packet switched elements): Αυτές οι οντότητες του δικτύου έχουν σχεδιαστεί για τη μεταφορά πακέτων δεδομένων. Αυτό επιτρέπει πολύ μεγαλύτερη χρήση του δικτύου και την δυνατότητα τα δεδομένα να μεταφέρονται ως πακέτα δρομολογούμενα προς τον προορισμό τους.

Στη συνέχεια παρουσιάζονται αναλυτικότερα οι παραπάνω δύο διαφορετικές περιοχές.

Κύκλωμα Μεταγωγής Στοιχείων (Circuit switched elements)

Το Κύκλωμα Μεταγωγής Στοιχείων περιλαμβάνει τις εξής οντότητες δικτύου:

- *Κέντρο Μεταγωγής Κινητού* (MSC): Το Κέντρο Μεταγωγής Κινητού, ή αλλιώς Mobile switching centre (MSC), ουσιαστικά διαχειρίζεται το κύκλωμα μεταγωγής κλήσεων σε εξέλιξη.

- *Πύλη MSC* (GMSC): Η Πύλη MSC, ή αλλιώς Gateway MSC (GMSC), είναι ουσιαστικά η διασύνδεση με τα εξωτερικά δίκτυα.

Στοιχεία Μεταγωγής Πακέτων (Packet switched elements)

Τα Στοιχεία Μεταγωγής Πακέτων περιλαμβάνουν τις εξής οντότητες δικτύου:

- *Serving GPRS Support Node* (SGSN): Όπως υποδηλώνει το όνομα, αυτή η οντότητα αναπτύχθηκε για πρώτη φορά στο GPRS, και η χρήση του μεταφέρθηκε στην αρχιτεκτονική του δικτύου UMTS. Το SGSN παρέχει μια σειρά από λειτουργίες μέσα στην αρχιτεκτονική του δικτύου UMTS, οι οποίες θα αναλυθούν εκτενώς σε επόμενο κεφάλαιο της παρούσας εργασίας. [9]

- *Κόμβος Υποστήριξης Πύλης GPRS* (GGSN): Ο Κόμβος Υποστήριξης Πύλης GPRS, ή αλλιώς Gateway GPRS Support Node (GGSN), εισήχθη για πρώτη φορά στο δίκτυο GPRS (όπως και το SGSN παρομοίως). Το GGSN είναι υπεύθυνο για τη δια συνεργασία μεταξύ του δικτύου GPRS και των εξωτερικά δικτύων μεταγωγής πακέτων. Κατά τη λειτουργία του, όταν το GGSN λαμβάνει δεδομένα που απευθύνονται σε ένα συγκεκριμένο χρήστη, ελέγχει εάν ο χρήστης είναι ενεργός και στη συνέχεια προωθεί τα δεδομένα στο SGSN που εξυπηρετεί το συγκεκριμένο UE.[7]

1.1.3 Το SGSN στα Δίκτυα

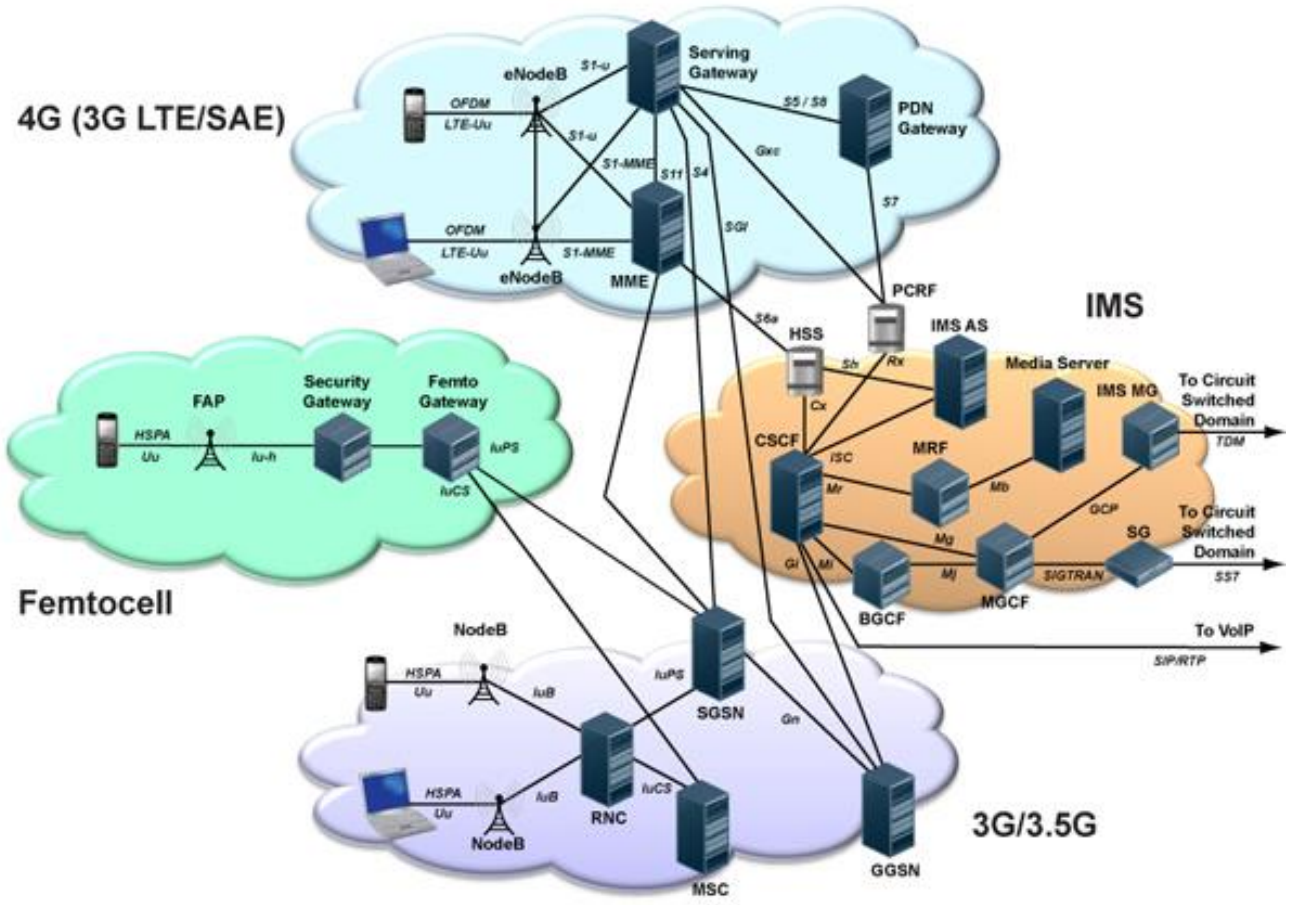
Ο Κόμβος εξυπηρέτησης υποστήριξης GPRS, ή αλλιώς Serving GPRS Support Node (SGSN) αποτελεί ένα κύριο κομμάτι του πυρήνα των δικτύων GPRS και UMTS. Το SGSN μέσα από τις βασικές λειτουργίες του εκτελεί τα ακόλουθα: 1) πραγματοποιεί την πιστοποίηση των χρηστών και των συσκευών που αυτοί χρησιμοποιούν για την πρόσβασή τους στο δίκτυο 2) διαχειρίζεται την κινητικότητα και τη θέση των συσκευών μέσα στο δίκτυο, 3) χειρίζεται και προωθεί όλα τα πακέτα δεδομένων που μετάγονται εντός του δικτύου.

Αντίστοιχες λειτουργίες για την κίνηση φωνής πραγματοποιεί το MSC. Συχνά το SGSN και το MSC συστεγάζονται στον ίδιο χώρο.

Το SGSN συνδέεται με το BSC και το RNC και αποτελεί το σημείο πρόσβασης υπηρεσίας στον πυρήνα του δικτύου GPRS για τον κινητό χρήστη. Από την άλλη πλευρά το SGSN ανταλλάσσει πακέτα δεδομένων με το GGSN. Το SGSN χειρίζεται την μετατροπή πρωτοκόλλου από το IP που χρησιμοποιείται μέσα στο δίκτυο κορμού προς το πρωτόκολλο υπό-δικτύου (Sub-Network-Dependent Convergence Protocol - SNDCP) και ελέγχου λογικού συνδέσμου (Logical Link Control - LLC), πρωτόκολλα που χρησιμοποιούνται μεταξύ του SGSN και των χρηστών κινητής τηλεφωνίας. Αυτά τα πρωτόκολλα υποστηρίζουν, εκτός των άλλων, μηχανισμούς συμπίεσης και κρυπτογράφησης των δεδομένων. Το SGSN είναι επίσης υπεύθυνο για την εξακρίβωση της γνησιότητας των κινητών τηλεφώνων. Όταν ο έλεγχος ταυτότητας είναι επιτυχής, το SGSN χειρίζεται την καταχώρηση του κινητού με το δίκτυο GPRS/UMTS και φροντίζει για τη διαχείριση της κινητικότητας του. [9]

Το SGSN παρέχει ένα πλήθος από λειτουργίες εντός της UMTS αρχιτεκτονικής, όπως παρουσιάζονται παρακάτω:

- *Διαχείριση της κινητικότητας*: Όταν ο χρήστης συνδέεται μέσω του εξοπλισμού του στην περιοχή Packet Switched του κεντρικού δικτύου UMTS, το SGSN δημιουργεί και συντηρεί πληροφορίες που βασίζονται στην τρέχουσα θέση του κινητού (Mobility Management - MM).
- *Διαχείριση της συνεδρίας*: Το SGSN διαχειρίζεται τις συνεδρίες δεδομένων (Session Management – SM) παρέχοντας σύνδεση με το ζητούμενο εξωτερικό δίκτυο δεδομένων με την απαιτούμενη ποιότητα υπηρεσιών, καθώς και τη διαχείριση των λεγόμενων PDP (Πρωτόκολλο Πακέτων δεδομένων) πλαισίων, δηλαδή τους αγωγούς πάνω από τους οποίους αποστέλλονται τα δεδομένα.
- *Αλληλεπίδραση με άλλες περιοχές του δικτύου*: Το SGSN μπορεί να διαχειρίζεται τα στοιχεία του εντός του δικτύου μόνο από την επικοινωνία του με άλλα στοιχεία/περιοχές του δικτύου, π.χ. MSC και άλλα κυκλώματα μεταγωγής περιοχών (circuit switched areas).
- *Χρεώσεις στους συνδρομητές*: Το SGSN είναι επίσης υπεύθυνο για τη συλλογή και αποστολή αρχείων καταγραφής χρεώσεων. Αυτό επιτυγχάνεται με την παρακολούθηση της ροής των δεδομένων των χρηστών σε όλο το δίκτυο. Τα CDRs (Call Detail Records) παράγονται από το SGSN προτού μεταφερθούν στους φορείς χρέωσης του παρόχου για επιπλέον επεξεργασία (Charging Gateway Function - CGF). [8]



Εικόνα 6: Αρχιτεκτονικές σύγχρονων δικτύων τρίτης και τέταρτης γενιάς. [10]

1.2 Τεχνολογία Λογισμικού

Ορισμός

Η τεχνολογία λογισμικού αποτελεί τον τομέα που πραγματεύεται τεχνικές, μεθοδολογίες, πρακτικές και εργαλεία για την συστηματική και μεθοδική σχεδίαση, την υλοποίηση, τον έλεγχο, και τη συντήρηση συστημάτων λογισμικού υψηλής ποιότητας εντός δεδομένου προϋπολογισμού και χρόνου εκτέλεσης. [11]

1.2.1 Κύκλος ζωής ανάπτυξης λογισμικού

Ο κύκλος ζωής ανάπτυξης λογισμικού ή αλλιώς SDLC (Software Development Life Cycle ή Software Development process) είναι μια διαδικασία που χρησιμοποιείται από την βιομηχανία παραγωγής λογισμικού για τον σχεδιασμό, την ανάπτυξη και τον έλεγχο λογισμικού υψηλής ποιότητας [12]. Ο SDLC αποσκοπεί στην παραγωγή ενός λογισμικού υψηλής ποιότητας που πληροί ή υπερβαίνει τις προσδοκίες των πελατών, αρκεί να ολοκληρωθεί εντός του προβλεπόμενου χρονοδιαγράμματος και του εκτιμώμενου κόστους.

Ο SDLC είναι μια διαδικασία που ακολουθείται σε ένα έργο λογισμικού. Αποτελείται από ένα λεπτομερές σχέδιο που περιγράφει την ανάπτυξη, την διατήρηση, την αντικατάσταση ή την

ενίσχυση του συγκεκριμένου λογισμικού. Ο κύκλος ζωής ορίζει μια μεθοδολογία για τη βελτίωση της ποιότητας του λογισμικού και τη συνολική διαδικασία ανάπτυξής του.

Το ακόλουθο σχήμα είναι μια γραφική αναπαράσταση των διαφόρων σταδίων μιας τυπικής διαδικασίας ανάπτυξης λογισμικού.

Εικόνα 7: Ο κύκλος ζωής της ανάπτυξης λογισμικού.

Ένας κύκλος ζωής Ανάπτυξης Λογισμικού τυπικά αποτελείται από τα ακόλουθα στάδια:

- **1ο Στάδιο: Σχεδιασμός και Ανάλυση Απαιτήσεων**

  Η Ανάλυση απαιτήσεων είναι το πιο σημαντικό και θεμελιώδες στάδιο του SDLC. Γίνεται από τα ανώτερα μέλη της εταιρείας με την βοήθεια του πελάτη, του τμήματος πωλήσεων και την έρευνα της αγοράς από εμπειρογνώμονες του τομέα της βιομηχανίας. Η πληροφορία αυτή χρησιμοποιείται στη συνέχεια για την βασική προσέγγιση του έργου και για τη διεξαγωγή μελέτης σκοπιμότητας του προϊόντος σε οικονομικό, επιχειρησιακό και τεχνικό επίπεδο.

  Ο Προγραμματισμός των απαιτήσεων για την διασφάλιση της ποιότητας και την αναγνώριση των κινδύνων που συνδέονται με το έργο, γίνεται επίσης στο στάδιο του σχεδιασμού. Το αποτέλεσμα της τεχνικής μελέτης σκοπιμότητας είναι να καθοριστούν οι διάφορες τεχνικές προσεγγίσεις που μπορούν να ακολουθηθούν για την επιτυχή υλοποίηση του έργου με ελάχιστους κινδύνους.

- **2ο Στάδιο: Καθορισμός των Απαιτήσεων**

Μόλις ολοκληρωθεί η ανάλυση των απαιτήσεων, το επόμενο βήμα είναι ο ακριβής καθορισμός και η τεκμηρίωση των απαιτήσεων του προϊόντος, καθώς και η έγκριση των παραπάνω από τον πελάτη ή τους αναλυτές της αγοράς. Αυτό επιτυγχάνεται μέσω ενός ειδικού εγγράφου που ορίζει το σύνολο των απαιτήσεων για τα προϊόντα που πρόκειται να σχεδιαστούν και να αναπτυχθούν κατά τη διάρκεια του κύκλου ζωής του έργου, το λεγόμενο SRS(Software Requirement Specification).

- **3ο Στάδιο: Σχεδιάζοντας την αρχιτεκτονική του προϊόντος**

Το SRS είναι το σημείο αναφοράς που πρόκειται να στηριχθούν οι αρχιτέκτονες λογισμικού ώστε το παραγόμενο λογισμικό-προϊόν να έχει την καλύτερη αρχιτεκτονική. Με βάση τις απαιτήσεις που καθορίζονται στο SRS, συνήθως, προτείνονται και τεκμηριώνονται περισσότερες από μία σχεδιαστικές προσεγγίσεις για την αρχιτεκτονική του προϊόντος. Η παραπάνω διαδικασία επιτυγχάνεται με το σχεδιασμό των εγγράφων προδιαγραφών, τα DDS(Design Document Specification).

Τα DDS εξετάζονται με βάση διάφορες παραμέτρους, όπως η αξιολόγηση των κινδύνων, η ευρωστία, ο σχεδιασμός της δομής, ο προϋπολογισμός και οι χρονικοί περιορισμοί. Με βάση τις παραπάνω παραμέτρους, επιλέγεται η καλύτερη προσέγγιση του σχεδιασμού για το προϊόν από όλα τα σημαντικά εμπλεκόμενα μέλη του έργου. Μια σχεδιαστική προσέγγιση ορίζει σαφώς όλες τις αρχιτεκτονικές ενότητες του προϊόντος καθώς και την επικοινωνία του, μέσω των ροών δεδομένων, με άλλες οντότητες(προϊόντα) αν υπάρχουν.

- **4ο Στάδιο: Ανάπτυξη του προϊόντος**

Σε αυτό το στάδιο του SDLC ξεκινά η πραγματική ανάπτυξη του προϊόντος. Ο κώδικας του προγράμματος δημιουργείται σύμφωνα με το DDS κατά το στάδιο αυτό. Αν ο σχεδιασμός γίνεται με λεπτομέρεια και οργάνωση, η παραγωγή κώδικα μπορεί να επιτευχθεί χωρίς μεγάλη ταλαιπωρία.

Οι προγραμματιστές οφείλουν να ακολουθούν τις κατευθυντήριες γραμμές που ορίζονται από την εταιρία. Επιπλέον, κατευθυντήριες γραμμές ορίζουν και τα εργαλεία προγραμματισμού, όπως μεταφραστές, διερμηνείς, αποσφαλματωτές που χρησιμοποιούνται για τη συγγραφή του κώδικα. Διαφορετικές γλώσσες προγραμματισμού υψηλού επιπέδου ,όπως η C, C ++, C#, Java, PHP, .NET, χρησιμοποιούνται για την δημιουργία παραγόμενου κώδικα. Η γλώσσα προγραμματισμού επιλέγεται σε σχέση με τον τύπο του λογισμικού που αναπτύσσεται και την πολιτική της εταιρίας.

- **5ο Στάδιο: Εξετάζοντας το Προϊόν**

Το στάδιο αυτό είναι συνήθως ένα υποσύνολο του συνόλου των σταδίων όπως στα σύγχρονα μοντέλα του SDLC. Οι δραστηριότητες ελέγχου ασχολούνται κυρίως με όλα τα στάδια του SDLC. Εντούτοις, σε αυτό το στάδιο, το προϊόν υποβάλλεται σε έλεγχο. Τα ελαττωματικά μέρη του προϊόντος, τα οποία παρουσίασαν σφάλματα, παρακολουθούνται και επανεξετάζονται, μέχρι το προϊόν να πληροί τα πρότυπα ποιότητας που ορίζονται στο SRS.

- **6ο Στάδιο: Η καθιέρωση του προϊόντος στην αγορά και η συντήρησή του**

Όταν το προϊόν έχει δοκιμαστεί και είναι τυπικά έτοιμο απελευθερώνεται στην κατάλληλη αγορά. Ορισμένες φορές, η ανάπτυξη γίνεται σε στάδια σύμφωνα με την στρατηγική της εταιρίας. Το προϊόν μπορεί πρώτα να απελευθερωθεί σε ένα περιορισμένο μέρος της αγοράς για να δοκιμαστεί πειραματικά (UAT) (δοκιμή αποδοχής χρηστών ή αλλιώς User acceptance testing). Στη συνέχεια, με βάση τις εντυπώσεις των χρηστών, το προϊόν μπορεί είτε να κυκλοφορήσει ως έχει, είτε να εφαρμοστούν προτεινόμενες βελτιώσεις. Μετά την είσοδο του προϊόντος στην αγορά, οι πελάτες του δικαιούνται υπηρεσίες συντήρησης για το προϊόν. [13][15]

1.2.2 Μοντέλα ανάπτυξης Λογισμικού

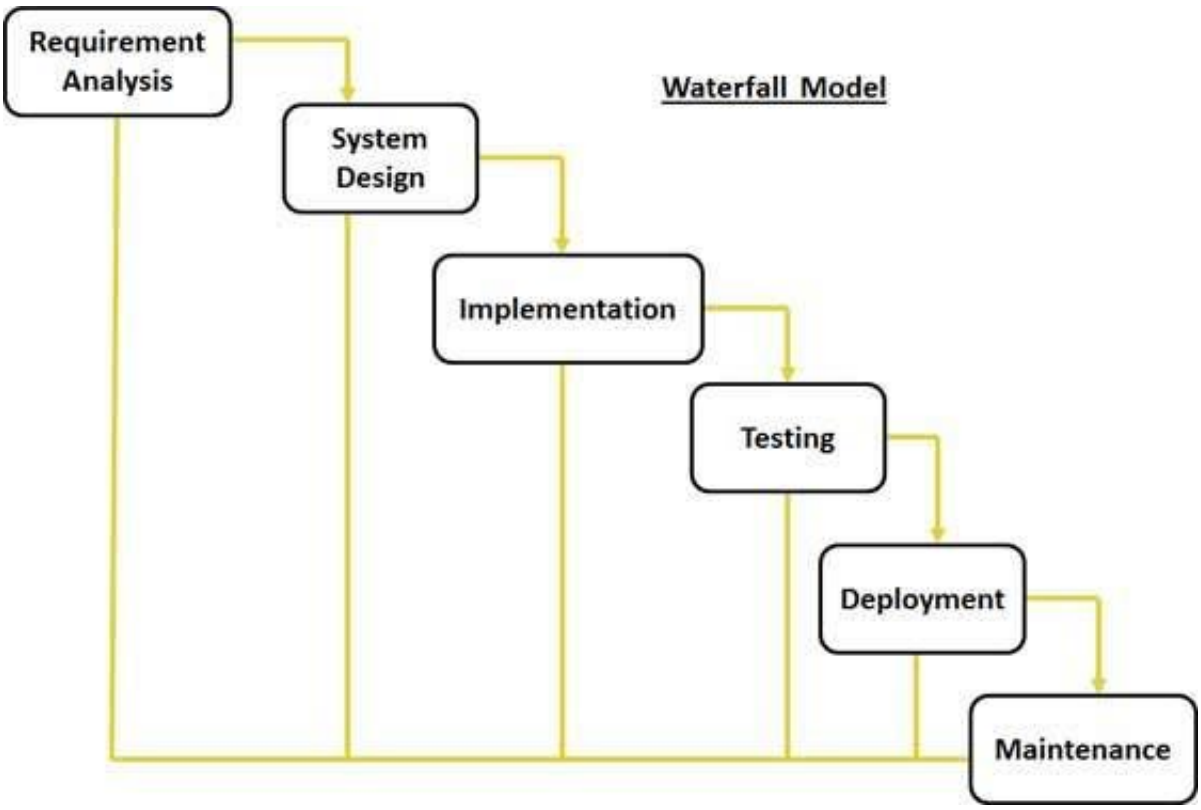
Τα Μοντέλα λογισμικού μπορούν να θεωρηθούν ως πολύ σημαντικά μέρη της οργανωμένης ανάπτυξης προϊόντων λογισμικού. Πιο συγκεκριμένα, είναι απαραίτητα καθώς καθορίζουν ένα σταθερό πλαίσιο και την περιγραφή υψηλού επιπέδου του τρόπου διαχείρισης της διάρκειας του κύκλου ζωής του προϊόντος λογισμικού.

Παρακάτω, θα περιγραφούν οι βασικές αρχές των δημοφιλών μοντέλων ανάπτυξης Λογισμικού. Σε αυτό το σημείο είναι απαραίτητο να ειπωθεί, πως από όλες τις φάσεις του κύκλου ζωής των προϊόντων λογισμικού, αυτή η εργασία επικεντρώνεται κυρίως στην ολοκλήρωση(integration) και την απελευθέρωση(releasing).

1.2.2.1 Μοντέλο Καταρράκτη

Το μοντέλο καταρράκτη, ή αλλιώς Waterfall Model, ήταν το πρώτο μοντέλο που χρησιμοποιήθηκε ευρέως στην Τεχνολογία Λογισμικού για την επιτυχή ολοκλήρωση του έργου. Σε αυτό το μοντέλο, η όλη διαδικασία ανάπτυξης λογισμικού χωρίζεται σε διακριτές φάσεις. Συνήθως, το αποτέλεσμα της μίας φάσης συμβάλει στην επόμενη φάση διαδοχικά.

Ακολουθεί μια διαγραμματική αναπαράσταση των διαφόρων φάσεων του μοντέλου καταρράκτη.



Εικόνα 8: Διαγραμματική αναπαράσταση του μοντέλου καταρράκτη.

Οι διαδοχικές φάσεις στο μοντέλο καταρράκτη είναι οι εξής:

Απαιτούμενη συλλογή και ανάλυση πληροφοριών (Requirement Gathering and analysis):

Όλες οι πιθανές απαιτήσεις του συστήματος που πρόκειται να αναπτυχθεί, καταγράφονται σε αυτό το στάδιο και τεκμηριώνονται σε ένα έγγραφο προδιαγραφών απαίτησης.

Σχεδιασμός Συστήματος (System Design):

Οι απαιτούμενες προδιαγραφές που μελετήθηκαν στο προηγούμενο βήμα χρησιμοποιούνται στο σχεδιασμό του συστήματος. Το παρόν βήμα βοηθά στον προσδιορισμό του υλικού και του συστήματος καθώς επίσης βοηθά στον καθορισμό της συνολικής αρχιτεκτονικής του συστήματος.

Υλοποίηση (Implementation):

Με τα αποτελέσματα του προηγούμενου βήματος, το σύστημα αναπτύσσεται αρχικά σε μικρά προγράμματα, που ονομάζονται μονάδες(units), τα οποία συνενώνονται στην επόμενη φάση. Κάθε μονάδα έχει αναπτυχθεί και δοκιμαστεί για τη λειτουργικότητα της, η οποία αναφέρεται ως Μονάδα Ελέγχου, ή αλλιώς Unit Testing.

Ένταξη και Δοκιμές (Integration and Testing):

Όλες οι μονάδες που αναπτύχθηκαν κατά τη φάση υλοποίησης, αφού υποβληθούν σε επιτυχή έλεγχο ξεχωριστά, εντάσσονται στο σύστημα. Μετά την ένταξη όλων των μονάδων, ολόκληρο το σύστημα ελέγχεται για τυχόν λάθη και παραλείψεις.

Εγκατάσταση του Συστήματος (Deployment of system):

Μόλις ολοκληρωθεί η λειτουργική και μη λειτουργική δοκιμή, το προϊόν εγκαθίσταται στο περιβάλλον του πελάτη ή κυκλοφορεί στην αγορά.

Συντήρηση (Maintenance):

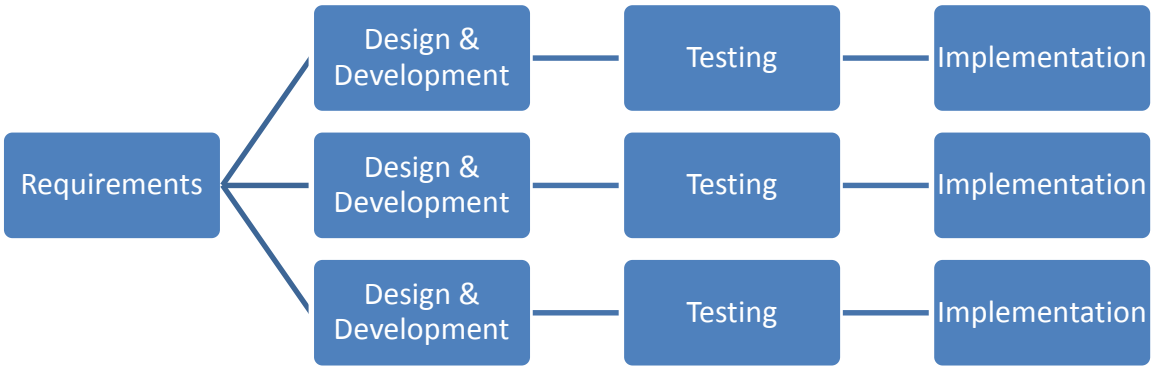
Μερικές φορές προκύπτουν ορισμένα ζητήματα στο περιβάλλον του πελάτη. Για να διορθωθούν αυτά τα ζητήματα γίνονται ενημερώσεις στο προϊόν με βοηθητικά τμήματα κώδικα(patches). Επιπλέον, για να ενισχυθεί το προϊόν κυκλοφορούν νέες εκδόσεις του(releases). Το βήμα της Συντήρησης υπάρχει για να ενσωματωθούν οι παραπάνω αλλαγές στο περιβάλλον του πελάτη.

Όλα αυτά τα στάδια κλιμακώνονται μεταξύ τους, με αποτέλεσμα η πρόοδος του συστήματος να ρέει σταθερά προς τα κάτω (όπως σε έναν καταρράκτη) μέσα από τις παραπάνω φάσεις. Η επόμενη φάση ξεκινά μόνο μετά την ολοκλήρωση των καθορισμένων στόχων για την προηγούμενη φάση. Συμπερασματικά, στο μοντέλο καταρράκτη οι φάσεις δεν συμπίπτουν. [14]

1.2.2.2 Επαναληπτικό Μοντέλο

Ένα επαναληπτικό μοντέλο κύκλου ζωής λογισμικού, ή αλλιώς Iterative Model, δεν ξεκινά με πλήρη περιγραφή των απαιτήσεων του συστήματος. Αντ 'αυτού, η ανάπτυξη αρχίζει με τον καθορισμό και την ανάπτυξη μόνο τμήματος του λογισμικού, το οποίο στη συνέχεια θα αναθεωρηθεί προκειμένου να εντοπιστούν περαιτέρω απαιτήσεις. Αυτή η διαδικασία επαναλαμβάνεται συνεχώς, με βασικό παράγοντα μια νέα έκδοση του λογισμικού στο τέλος κάθε επανάληψης του μοντέλου. [16]

Παρακάτω παρατίθεται η γραφική απεικόνιση του Επαναληπτικού μοντέλου.



Εικόνα 9: Γραφική αναπαράσταση του επαναληπτικού μοντέλου

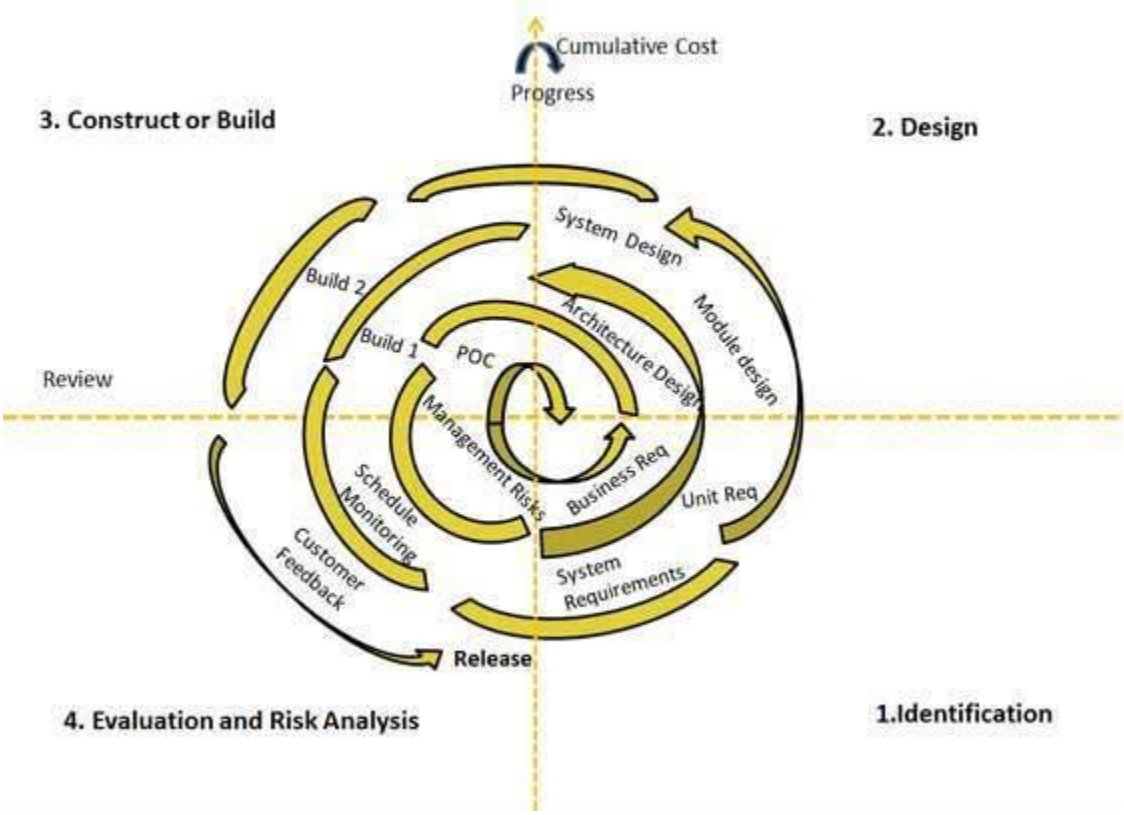
Όπως και σε άλλα μοντέλα SDLC, η επαναληπτική και σταδιακή ανάπτυξη έχει κάποιες ειδικές εφαρμογές στη βιομηχανία λογισμικού. Αυτό το μοντέλο χρησιμοποιείται πιο συχνά στα ακόλουθα σενάρια:

- Όταν οι απαιτήσεις ολόκληρου του συστήματος είναι καθορισμένες με σαφήνεια και κατανοητές.
- Όταν οι σημαντικές απαιτήσεις είναι καθορισμένες. Ωστόσο, σε περίπτωση που ορισμένες λειτουργίες ή επιπλέον βελτιώσεις ζητηθούν αργότερα, πρέπει να μπορούν να υλοποιηθούν.
- Μια νέα τεχνολογία που η ομάδα ανάπτυξης προσπαθεί να εξοικειωθεί μαζί της, ενώ εργάζεται παράλληλα για το συγκεκριμένο έργο.
- Πόροι (υλικοί και ανθρώπινοι) με αναγκαίες δεξιότητες που δεν είναι διαθέσιμοι και προβλέπεται να χρησιμοποιηθούν βάσει σύμβασης για συγκεκριμένες επαναλήψεις.
- Υπάρχουν μερικά χαρακτηριστικά υψηλής επικινδυνότητας και στόχους που μπορούν να αλλάξουν στο μέλλον.

1.2.2.3 Σπειροειδές Μοντέλο

Το Σπειροειδές Μοντέλο, ή αλλιώς Spiral Model, συνδυάζει την ιδέα της επαναληπτικής ανάπτυξης με τις ελεγχόμενες πτυχές του μοντέλου καταρράκτη. Πιο συγκεκριμένα, είναι ένας συνδυασμός του επαναληπτικού μοντέλου ανάπτυξης και της διαδοχικής γραμμικής ανάπτυξης του μοντέλου καταρράκτη. Δηλαδή, με πολύ μεγάλη έμφαση στην ανάλυση κινδύνου. Το μοντέλο παρέχει τη δυνατότητα σταδιακής κυκλοφορίας του προϊόντος, ή σταδιακής βελτίωσης μέσα από κάθε επανάληψη γύρω από ένα φαύλο κύκλο. [18]

Ακολουθεί διαγραμματική αναπαράσταση του σπειροειδούς μοντέλου και των δραστηριοτήτων του σε κάθε φάση.



Εικόνα 10: Διαγραμματική αναπαράσταση του σπειροειδούς μοντέλου και των δραστηριοτήτων του σε κάθε φάση.

Το σπειροειδές μοντέλο έχει τέσσερις φάσεις. Ένα έργο λογισμικού περνά επανειλημμένα μέσω αυτών των φάσεων σε επαναλήψεις που ονομάζεται σπείρες.

- ***Αναγνώριση*** **(Identification):** Η φάση αυτή ξεκινά με τη συλλογή των απαιτήσεων της επιχειρηματικότητας στην αρχική σπείρα. Ο προσδιορισμός των απαιτήσεων του συστήματος γίνεται σε αυτήν την φάση μέσα από επόμενες σπείρες. Επιπλέον, στην φάση αυτή περιλαμβάνεται η κατανόηση των απαιτήσεων του συστήματος με τη συνεχή επικοινωνία μεταξύ του πελάτη και του αναλυτή συστήματος. Στο τέλος της σπείρας το προϊόν έχει αναπτυχθεί εντός της οριοθετημένης αγοράς.

- ***Σχεδιασμός*** **(Design):** Η Φάση του σχεδιασμού περιλαμβάνει τον αρχιτεκτονικό σχεδιασμό, τον λογικό σχεδιασμό των τμημάτων(modules), φυσικό σχεδιασμό των προϊόντων και της οριστικής μελέτης στις επόμενες σπείρες.

- ***Δημιουργία*** **(Build):** Η συγκεκριμένη φάση αναφέρεται στην παραγωγή του λογισμικού σε κάθε σπείρα. Όταν το προϊόν είναι έτοιμο να δημιουργηθεί, αναπτύσσεται ένα πειραματικό μοντέλο (Proof of Concept) για να πάρει τα σχόλια των πελατών. Στη συνέχεια, στις επόμενες σπείρες παράγονται εκδόσεις(με έναν αριθμό έκδοσης για κάθε έκδοση) του προϊόντος που πλησιάζουν τις απαιτήσεις και τις

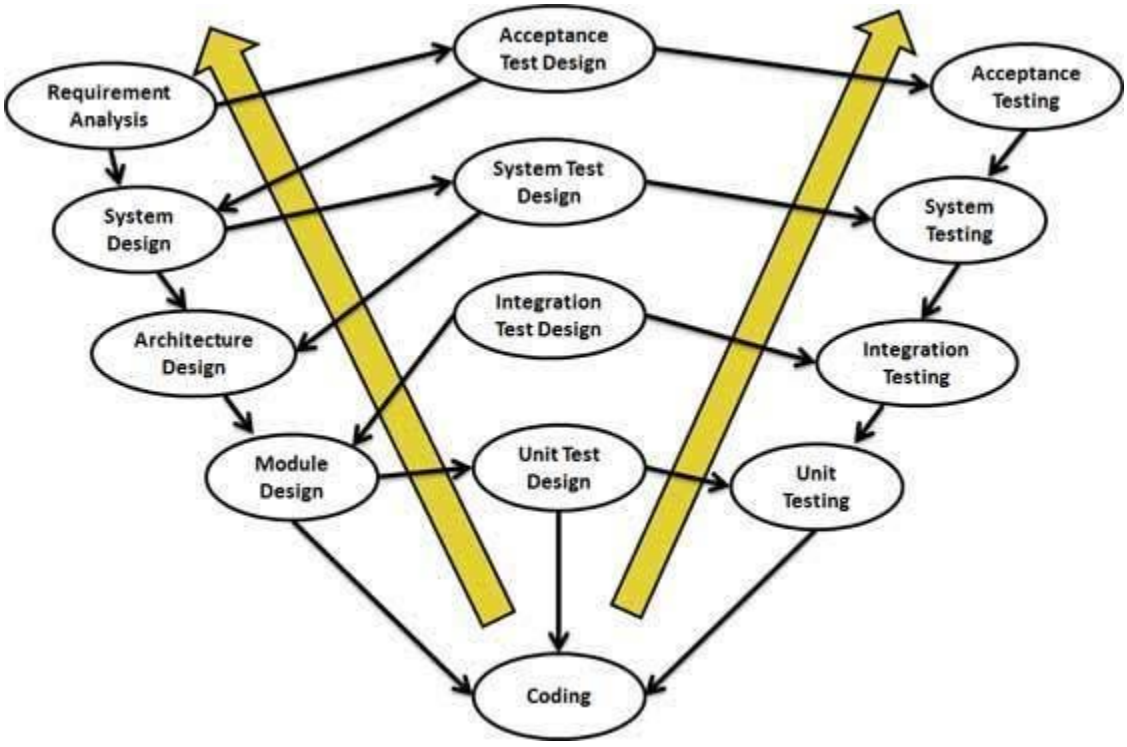
λεπτομέρειες σχεδιασμού. Αυτές δημιουργούνται(Build) και αποστέλλονται στον πελάτη για την ανατροφοδότηση (feedback).

- ***Αξιολόγηση και Ανάλυση Κινδύνου*** **(Evaluation and Risk Analysis):** Η ανάλυση επικινδυνότητας περιλαμβάνει τον εντοπισμό, την εκτίμηση, την παρακολούθηση και τη διαχείριση των κινδύνων ως πρός την μεταβολή του χρονοδιαγράμματος και την υπέρβαση του κόστους. Μετά τη δοκιμή της έκδοσης του προϊόντος, ο πελάτης αξιολογεί το λογισμικό και παρέχει ανατροφοδότηση κατά το τέλος της πρώτης επανάληψης. [17]

1.2.2.4 Μοντέλο V

Στο μοντέλο V, ή αλλιώς V-Model, η εκτέλεση των διαδικασιών συμβαίνει με ένα διαδοχικό τρόπο σχήματος V. Το συγκεκριμένο μοντέλο είναι γνωστό και ως Μοντέλο Επαλήθευσης και Επικύρωσης (Verification and Validation Model). Το V - μοντέλο είναι μια επέκταση του μοντέλου καταρράκτη και βασίζεται στην ένωση της φάσης των δοκιμών για κάθε αντίστοιχο στάδιο ανάπτυξης. Αυτό σημαίνει ότι για κάθε φάση του κύκλου ανάπτυξης συνδέεται άμεσα φάση δοκιμών. Είναι ένα πολύ πειθαρχημένο μοντέλο και η επόμενη φάση ξεκινά μόνο μετά την ολοκλήρωση της προηγούμενης φάσης. [18]

Το παρακάτω σχήμα απεικονίζει τις διάφορες φάσεις V-Μοντέλου.



Εικόνα 11: Το μοντέλο V.

1.2.2.5 Μοντέλο Μεγάλης Έκρηξης

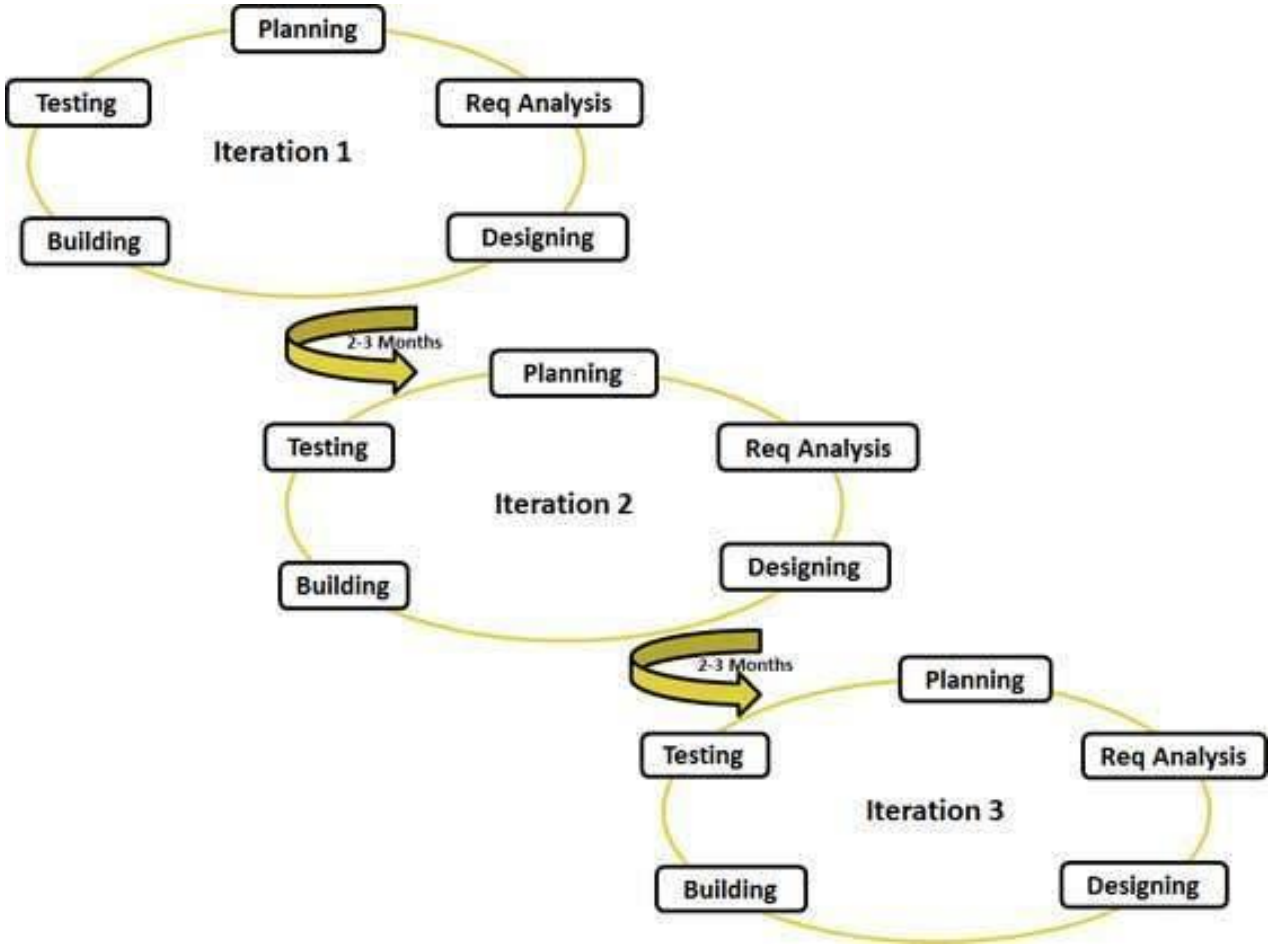
Το μοντέλο Μεγάλης Έκρηξης, ή αλλιώς Big Bang Model, είναι ένα μοντέλο SDLC όπου δεν ακολουθείται κάποια συγκεκριμένη διαδικασία. Η ανάπτυξη ξεκινά με τα απαραίτητα χρήματα και οφέλη και το παραγόμενο αποτέλεσμα είναι το προϊόν λογισμικού όπου ο πελάτης μπορεί να μην καλύπτεται. Επιπλέον, δεν ακολουθείται κάποια επίσημη τεχνική ανάπτυξης του προϊόντος και απαιτείται ελάχιστος σχεδιασμός. Ο πελάτης σε αυτήν την περίπτωση πιθανόν να μην γνωρίζει ακριβώς τις προδιαγραφές του προϊόντος, οι οποίες μπορούν να προκύπτουν κατά την δημιουργία χωρίς ιδιαίτερη ανάλυση. Συνήθως, το μοντέλο αυτό ακολουθείται σε μικρά έργα όπου οι ομάδες ανάπτυξης είναι περιορισμένες.

1.2.2.6 Ευέλικτο Μοντέλο

Το Ευέλικτο μοντέλο, ή αλλιώς Agile Model, είναι ένας συνδυασμός του επαναληπτικού και αυξητικών μοντέλων διαδικασίας, με έμφαση στις διαδικασίες προσαρμοστικότητας και την ικανοποίηση των πελατών από την ταχεία παράδοση των προϊόντων λογισμικού.

Το μοντέλο αυτό «σπάει» το προϊόν σε μικρά προσαυξανόμενα τμήματα(Builds). Αυτά τα τμήματα παράγονται σε επαναλήψεις. Κάθε επανάληψη τυπικά διαρκεί από μία έως τρείς εβδομάδες. Σε κάθε επανάληψη περιλαμβάνονται πολλαπλές ομάδες που εργάζονται ταυτόχρονα σε διάφορους τομείς, όπως τον σχεδιασμό, την ανάλυση απαιτήσεων, την σχεδίαση, την συγγραφή του κώδικα, τον έλεγχο των μονάδων(unit testing) και την αποδοχή των δοκιμών. Στο τέλος της επανάληψης ένα προϊόν που λειτουργεί εμφανίζεται στον πελάτη και στους άμεσα ενδιαφερόμενους. [19]

Ακολουθεί μία γραφική αναπαράσταση του Ευέλικτου μοντέλου.



Εικόνα 12: Γραφική αναπαράσταση του ευέλικτου μοντέλου.

Οι πιο δημοφιλείς ευέλικτες μέθοδοι είναι: Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, και Dynamic Systems Development Method (DSDM) (1995). Οι παραπάνω μέθοδοι αναφέρονται συλλογικά ως Ευέλικτες μεθοδολογίες, μετά το Agile Manifesto που δημοσιεύτηκε το 2001. [21]

Παρακάτω καταγράφονται οι βασικοί κανόνες του Agile Manifesto.

- **Άτομα και Αλληλεπιδράσεις:** Στην Ευέλικτη Ανάπτυξη, η αυτό-οργάνωση και τα κίνητρα είναι σημαντικά, όπως και οι αλληλεπιδράσεις και η συστέγαση και ο προγραμματισμός ζεύγους.(pair programming)
- **Λογισμικό που λειτουργεί:** Ένα πειραματικό λογισμικό θεωρείται το καλύτερο μέσο για την επικοινωνία με τον πελάτη για γίνουν αντιληπτές οι απαιτήσεις τους.
- **Συνεργασία του πελάτη:** Δεδομένου ότι οι απαιτήσεις δεν μπορούν να καλυφθούν απόλυτα από την αρχή του έργου (οφείλεται σε διάφορους παράγοντες

συνήθως), η συνεχής επικοινωνία με τους πελάτες είναι πολύ σημαντική στην καταγραφή των σωστών απαιτήσεων του προϊόντος.

- **Ανταπόκριση στις αλλαγές:** Η Ευκίνητη Ανάπτυξη εστιάζεται στην εγρήγορση για πιθανές αλλαγές και τη συνεχή ανάπτυξη. [20]

1.2.2.7 Μοντέλο Ταχείας Ανάπτυξης Εφαρμογών

Το Μοντέλο Ταχείας Ανάπτυξης Εφαρμογών, ή αλλιώς RAD Model (Rapid Application Development), βασίζεται στην προτυποποίηση και την επαναληπτική ανάπτυξη χωρίς συγκεκριμένο σχεδιασμό. Η διαδικασία της συγγραφής του ίδιου του λογισμικού περιλαμβάνει το σχεδιασμό που απαιτείται για την ανάπτυξη του προϊόντος. Εστιάζει στη συλλογή των απαιτήσεων των πελατών μέσα από εργαστήρια ή ομάδες εστίασης, τον έγκαιρο έλεγχο των πρωτοτύπων από τον πελάτη χρησιμοποιώντας πειραματικά μοντέλα, την επαναχρησιμοποίηση των υφιστάμενων προτύπων (components), η συνεχής ολοκλήρωση(**Continuous Integration**) και την ταχεία παράδοση.

Στο συγκεκριμένο μοντέλο, η ανάλυση, ο σχεδιασμός, η δημιουργία και το στάδιο των δοκιμών διανέμεται σε μία σειρά επαναλαμβανόμενων κύκλων ανάπτυξης. Παρακάτω ακολουθούν οι φάσεις του μοντέλου.

- **Επιχειρηματικό Μοντέλο (Business Modeling):** Το επιχειρηματικό μοντέλο για το αναπτυσσόμενο προϊόν έχει σχεδιαστεί από την άποψη της ροής των πληροφοριών και τη διανομή των πληροφοριών μεταξύ των διαφόρων επιχειρηματικών καναλιών. Μια πλήρης επιχειρηματική ανάλυση γίνεται για να βρεθούν οι ζωτικής σημασίας πληροφορίες για τις επιχειρήσεις, όπως το πώς μπορεί να επιτευχθεί και ποιοι είναι οι παράγοντες που οδηγούν την επιτυχή ροή των πληροφοριών.

- **Μοντελοποίηση των Δεδομένων (Data Modeling):** Οι πληροφορίες που συγκεντρώθηκαν κατά το στάδιο της μοντελοποίησης επιχειρηματικών διαδικασιών εξετάζονται και αναλύονται σε ομάδες δεδομένων ζωτικής σημασίας για την επιχείρηση. Τα χαρακτηριστικά όλων των συνολικών στοιχείων αναγνωρίζονται και καταγράφονται. Η σχέση μεταξύ αυτών των δεδομένων καθορίζεται λεπτομερώς πρός όφελος του επιχειρηματικού μοντέλου.

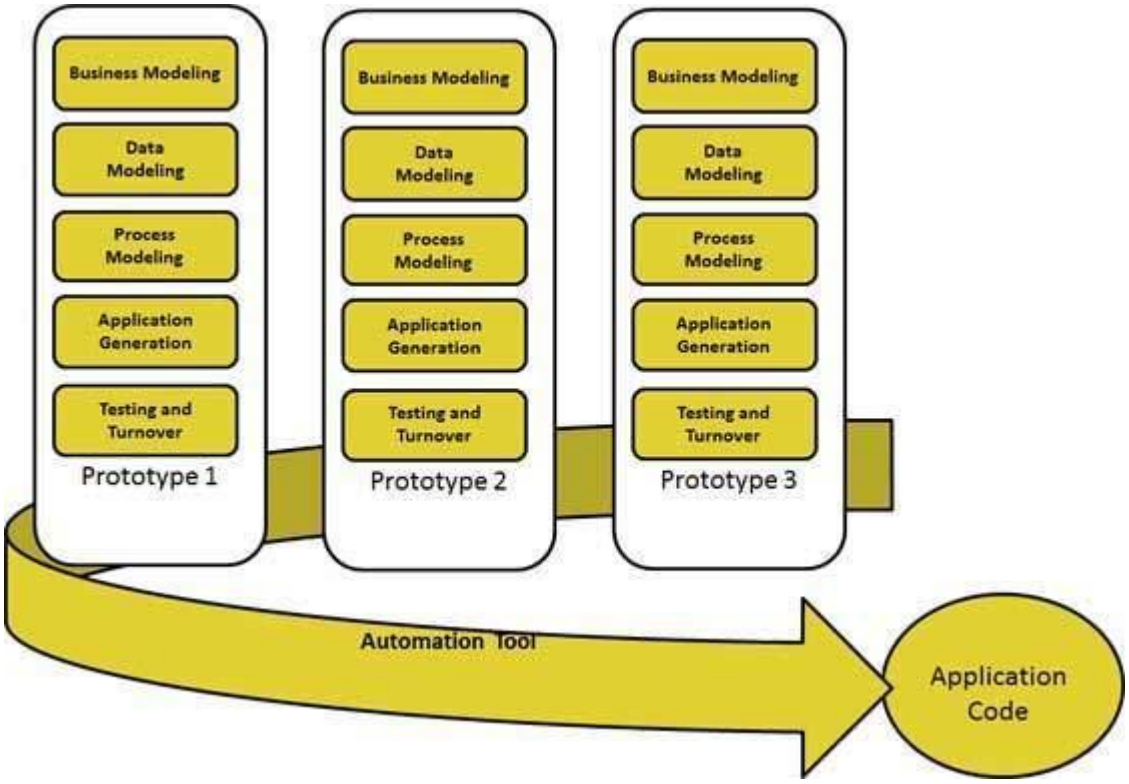
- **Μοντελοποίηση Διαδικασιών (Process Modeling):** Τα σύνολα των δεδομένων που ορίζονται στη φάση Μοντελοποίηση Δεδομένων μετατρέπονται για τον καθορισμό της ροής των επιχειρηματικών πληροφοριών που είναι απαραίτητες για την επίτευξη συγκεκριμένων επιχειρηματικών στόχων, σύμφωνα με το επιχειρηματικό μοντέλο. Σε αυτή τη φάση το Μοντέλο Διαδικασιών ορίζεται για τυχόν αλλαγές ή βελτιώσεις των δεδομένων. Η περιγραφή της διαδικασίας για την προσθήκη, διαγραφή, ανάκτηση ή τροποποίηση ενός αντικειμένου δεδομένων είναι δεδομένη.

- **Γέννηση Εφαρμογών (Application Generation):** Το εκάστοτε σύστημα δημιουργείται(Built) με τη χρήση εργαλείων αυτοματισμού για τη μετατροπή των διαδικασιών και των μοντέλων των δεδομένων σε πραγματικά πρωτότυπα.

- **Δοκιμές και κύκλος εργασιών (Testing and Turnover):** Ο συνολικός χρόνος δοκιμής μειώνεται στο RAD μοντέλο, αφού τα πρωτότυπα δοκιμάζονται ανεξάρτητα

κατά τη διάρκεια κάθε επανάληψης του κύκλου. Ωστόσο, η ροή των δεδομένων και οι διασυνδέσεις μεταξύ όλων των συνιστωσών πρέπει να ελεγχθεί διεξοδικά με πλήρη κάλυψη. [13]

Παρακάτω ακολουθεί γραφική αναπαράσταση του RAD μοντέλου.



Εικόνα 13: Γραφική αναπαράσταση του RAD μοντέλου.

2. STATE OF THE ART

Σε αυτό το κεφάλαιο, θα παρουσιαστεί αρχικά το SGSN όπως υπάρχει στις μέρες μας από άποψη υλικού και λογισμικού. Έπειτα, θα αναλυθεί το CI και θα παρατεθούν τα εργαλεία που συμβάλουν στην υλοποίησή του. Τέλος, έχοντας λάβει υπόψιν την ανάλυση των στοιχείων από τα προηγούμενα κεφάλαια, προκύπτει το πρωταρχικό ερώτημα και οι περαιτέρω στόχοι που πραγματεύεται η παρούσα εργασία.

2.1 Το SGSN

Μέχρι πρόσφατα, οι κατασκευαστές των τηλεπικοινωνιακών συστημάτων είχαν αναπτύξει επιτυχώς και είχαν βελτιώσει πολλές γενιές εξοπλισμού με βάση ιδιόκτητες πλατφόρμες συστημάτων (proprietary system platforms). Έτσι, οι κατασκευαστές έχουν επενδύσει σε πρόσθετες δαπάνες για την έρευνα και ανάπτυξη (R&D) ιδιόκτητου εξοπλισμού σε επίπεδο υλικού (hardware). Ωστόσο, στα σύγχρονα τηλεπικοινωνιακά δίκτυα δίνεται περισσότερη έμφαση στις δυνατότητες που δημιουργούνται από το λογισμικό. Συνεπώς, η αξία τους καθορίζεται όλο και περισσότερο από τις σύνθετες εφαρμογές και υπηρεσίες που βασίζονται σε λογισμικό. Επιπλέον, κάθε νέα καινοτομία στον τομέα των τηλεπικοινωνιακών δικτύων αποτελεί πρόκληση για τους κατασκευαστές. Κατά συνέπεια, οι κατασκευαστές καλούνται να παράξουν νέες πλατφόρμες συστημάτων με πιο εξελιγμένες τεχνολογίες γενικών εφαρμογών, πιο ευέλικτη υποστήριξη πολλαπλών πρωτοκόλλων, μεγαλύτερη ικανότητα ανταπόκρισης στις αυξανόμενες απαιτήσεις της κυκλοφορίας των νέων υπηρεσιών, ενώ παράλληλα να παραδώσουν όλα αυτά πολύ πιο γρήγορα και πιο αποδοτικά.

Σύμφωνα με τα παραπάνω γίνεται αντιληπτό, ότι δημιουργείται ένα νέο λειτουργικό πρότυπο, το οποίο οι κατασκευαστές τηλεπικοινωνιακών δικτύων οφείλουν να ακολουθήσουν. Έτσι, καλούνται να επενδύσουν στην ανάπτυξη ανοικτών προτύπων για τον εξοπλισμό της επόμενης γενιάς. Τα οφέλη των ανοικτών προτύπων έχουν γίνει ήδη δεκτά σε άλλες βιομηχανίες:

- Μειώνεται το κόστος έρευνας και ανάπτυξης (R&D), αφού το συνολικό κόστος διαμοιράζεται σε μια ευρύτερη κοινότητα.
- Μειώνεται το χρονικό διάστημα που το προϊόν μπαίνει στην αγορά (time to market).
- Επιταχύνεται η υιοθέτηση νέων τεχνολογιών.

Αξιοποιώντας τα παραπάνω ισχυρά οφέλη, οι κατασκευαστές τηλεπικοινωνιακών δικτύων έφεραν «επανάσταση» στον τρόπο με τον οποίο αναπτύσουν τα συστήματά τους προσφέροντας μεγαλύτερης αξίας προϊόντα στους πελάτες τους. Μία τέτοια ανοιχτή πλατφόρμα είναι η AdvancedTCA (Advanced Telecom Computing Architecture - ATCA). [22]

2.1.1 Αρχιτεκτονική του SGSN από άποψη υλικού

Η διαθεσιμότητα, η αξιοπιστία και η λειτουργικότητα είναι οι τρεις σημαντικές πτυχές για τα υλικά (hardware) τηλεπικοινωνιών που πρέπει να πληρούν τα αυστηρά πρότυπα που καθορίζονται από τους εθνικούς και διεθνείς βιομηχανικούς οργανισμούς. Ο PICMG (PCI Industrial Computer Manufacturers Group) [23] ορίζει την ATCA ως νέα αρχιτεκτονική γενιάς για την οικοδόμηση ολοκληρωμένου εξοπλισμού.

Η ATCA, είναι η πρώτη ανοικτή αρχιτεκτονική που παρέχει μια εξαιρετικά εκλεπτυσμένη και ισχυρή αρχιτεκτονική διαχείρισης του συστήματος, που επιτρέπει συστήματα υψηλής διαθεσιμότητας να συνεχίζουν να λειτουργούν σε περίπτωση που μεμονωμένο συστατικό ή υπό-σύστημα βγει εκτός λειτουργίας. Αυτό επιτρέπει, επίσης, "on-the-fly" αναβαθμίσεις λογισμικού, δηλαδή ενώ το σύστημα βρίσκεται σε λειτουργία. Η ATCA, αναπτύχθηκε από την (PICMG).[24] Η αρχιτεκτονική ATCA υλοποιείται χρησιμοποιώντας σε ράφια (shelves) και πλακέτες με πολλές υποδοχές (slots) ή υπό-ράφια, όπου μπορούν να εισαχθούν πίνακες (boards) ή blades. Αυτοί οι πίνακες μπορεί να είναι πίνακες ελέγχου, ειδικοί για εφαρμογές ή πίνακες διανομής ηλεκτρικής ενέργειας. Συχνά υπάρχει τουλάχιστον μία κύρια πλακέτα ελέγχου η οποία ελέγχει τη λειτουργία των συγκεκριμένων πινάκων εφαρμογών σε ένα ράφι ATCA. Η κύρια επικοινωνία μεταξύ των σανίδων διανύει τον κύριο δίαυλο οπίσθιου τοιχώματος διασύνδεσης.

2.1.2 Αρχιτεκτονική του SGSN από άποψη λογισμικού

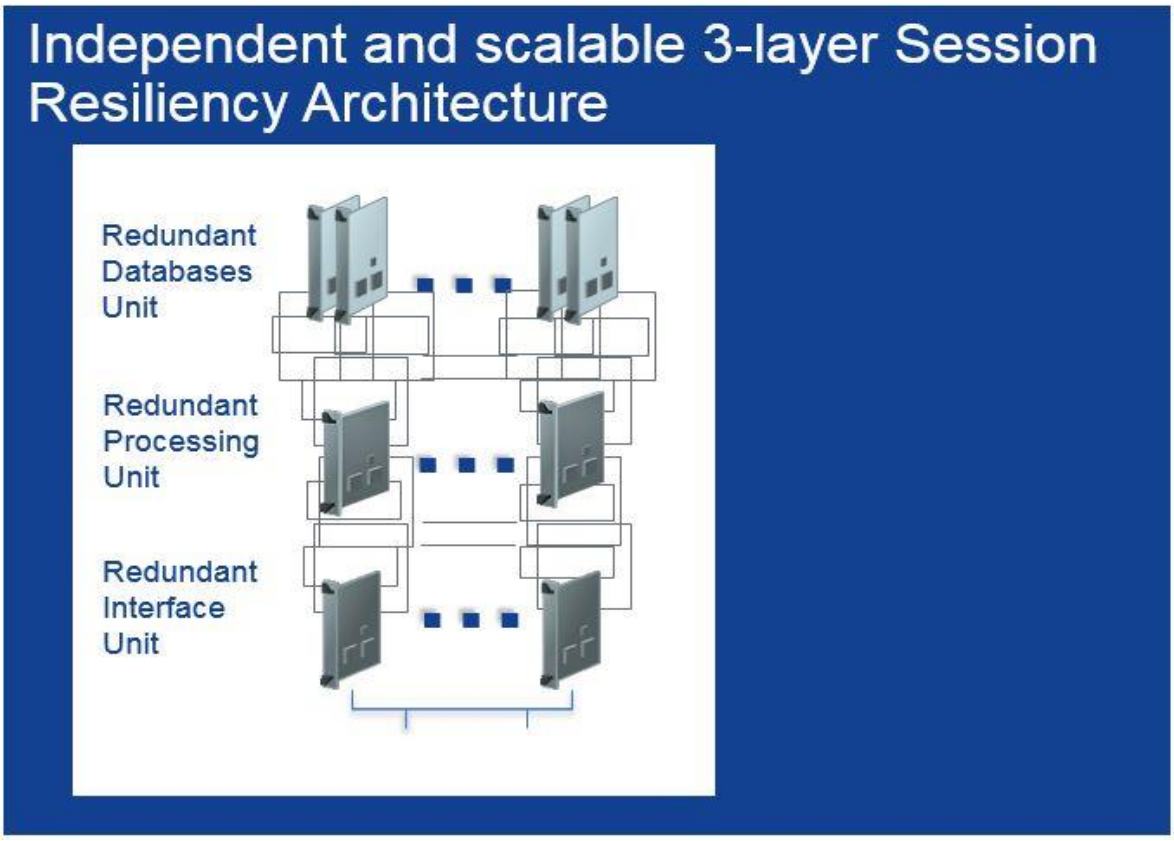
Το λογισμικό του SGSN αποτελείται από δύο τμήματα, την πλατφόρμα λογισμικού και τις εφαρμογές δικτύου. Οι πλατφόρμες λογισμικού διαχωρίζουν το λογισμικό των εφαρμογών δικτύου από το υλικό. Από την οπτική των εφαρμογών δικτύου, ο διαχωρισμός αυτός επιτρέπει στους προγραμματιστές να επικεντρώνονται στις εφαρμογές που αναπτύσουν. Στην πραγματικότητα, αυτές οι πλατφόρμες λογισμικού αποτελούν μια διεπαφή ανάμεσα στο υλικό και τις εφαρμογές δικτύου και παρέχουν τη δυνατότητα ανάπτυξης λογισμικού ανεξάρτητα από την υλικό πάνω στο οποίο θα εφαρμοστεί. Αυτό επιτρέπει τη βέλτιστη επαναχρησιμοποίηση του λογισμικού από τις υπάρχουσες ώριμες εφαρμογές, όπως το υλικό και τα υψηλότερα επίπεδα εφαρμογών σε ένα επίπεδο που δεν υπάρχει αλληλεξάρτηση.[25] Η παρούσα εργασία φέρει ως παράδειγμα λογισμικού του SGSN το FlexiNS. Το FlexiNS αποτελεί προϊόν της NOKIA. Η ικανότητα διαχείρισης υψηλού αριθμού transactions που παρέχει το Flexi NS παίζει σημαντικό ρόλο στην επίπεδη αρχιτεκτονική λόγω των αυξημένων απαιτήσεων για σηματοδοσία (signaling) και αποτελεί σημαντικό παράγοντα διαφοροποίησης σε σύγκριση με άλλα λογισμικά SGSN τα οποία επικεντρώνονται μόνο στο μέγεθος της βάσης δεδομένων του συνδρομητή. [26]

Εικόνα 14: Υλικός εξοπλισμός ATCA NOKIA.

Πιο συγκεκριμένα, μερικά αξιοσημείωτα χαρακτηριστικά του FlexiNS είναι [27]:

- Ενσωματώνει τεχνολογίες 2G, 3G, 4G (TripleAccessControl).
- Η αρχιτεκτονική του έχει σχεδιαστεί με γνώμονα την ανθεκτικότητα, ώστε να μπορεί για να χειριστεί οποιαδήποτε αποτυχία σε μονάδα του υλικού με μηδενική απώλεια για τις συνεδρίες και τις υπηρεσίες.
- Η απλοποίηση του δικτύου με λειτουργικότητα και συνδεσιμότητα με την βοήθεια πλήρους ενσωματωμένου πλαισίου αναμετάδοσης (βελτίωση της αξιοπιστίας και της αποτελεσματικότητας) και της απλοποιημένης συνδεσιμότητας IP.
- Η υποστήριξη φωνής πάνω από IP (VoIP).

Παρακάτω παρουσιάζεται σχηματικά η Αρχιτεκτονική Ανάκτησης.



Εικόνα 15: Λειτουργία αρχιτεκτονικής ανάκτησης. [27]

Στη συνέχεια θα αναφερθούν, ορισμένα τεχνικά χαρακτηρίστηκα του Flexi NS σχετικά με την χωρητικότητα που μπορεί να υποστηρίξει. [27]

- 9M Συνδρομητές.
- 10M PDP contexts.
- 25k Signaling trans/sec.
- 3.6 Gbps 2G Throughput.
- 16.8 Gb/s 3G Throughput.

2.2 Συνεχής Ενσωμάτωση (CI)

Στις πρώιμες εποχές της βιομηχανίας λογισμικού, μια από τις πιο δύσκολες στιγμές ενός project λογισμικού ήταν η ενοποίηση των τμημάτων λογισμικού. Ήταν το μέρος του project το οποίο συνήθως αποτύγχανε. Σύμφωνα με το οποίο, όλα τα τμήματα που λειτουργούσαν ατομικά έπρεπε να συγκεντρωθούν και να συνενωθούν προκειμένου να αποτελέσουν το προϊόν λογισμικού.

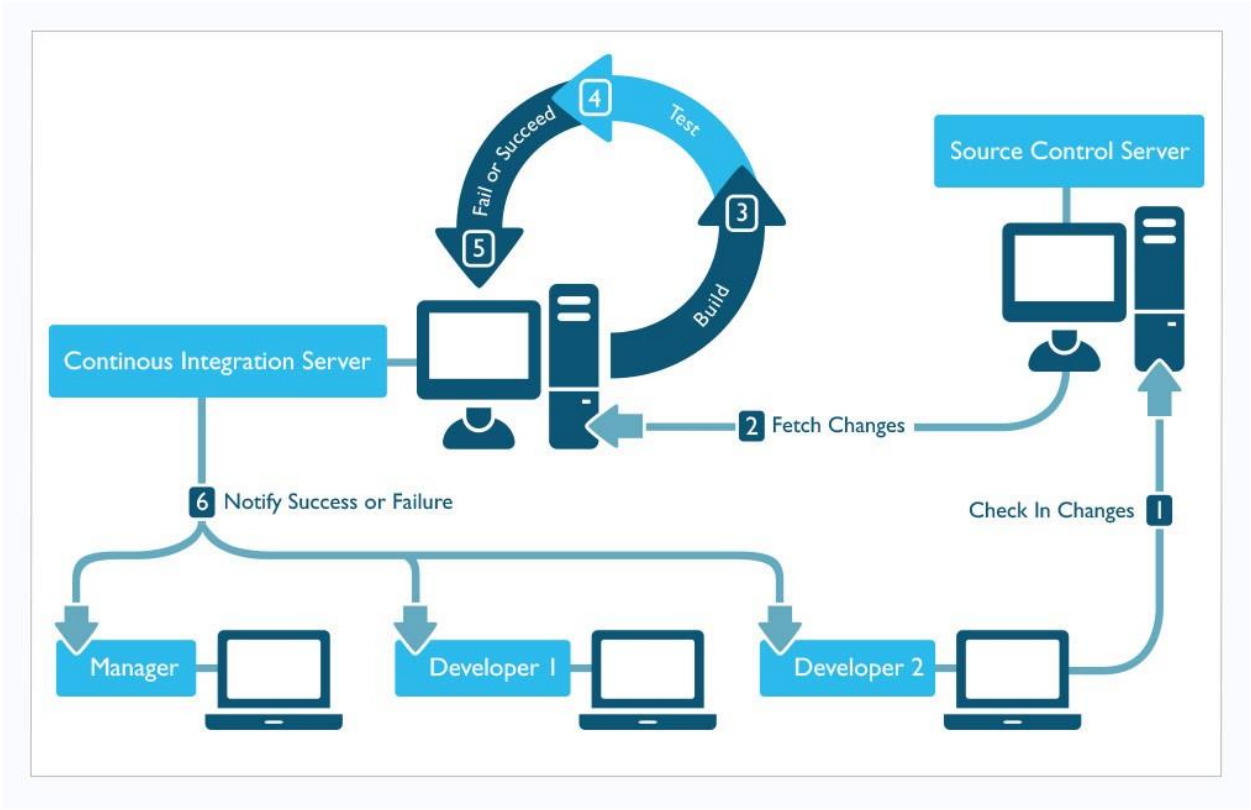
Το CI είναι μια ευέλικτη πρακτική μέθοδος για την συνεχή ενσωμάτωση μικρών αυξήσεων κώδικα (είτε χειροκίνητα είτε αυτόματα), με στόχο την αποφυγή μεγάλων προσαυξήσεων κώδικα όπως στα παραδοσιακά μοντέλα ανάπτυξης λογισμικού (μοντέλο καταρράκτη). [28]

Η Συνεχής Ενσωμάτωση, ή αλλιώς Continuous Integration, είναι μια αναπτυξιακή πρακτική που απαιτεί από τους προγραμματιστές να ενσωματώνουν τον κώδικά τους σε ένα κοινό αποθετήριο ανά τακτά χρονικά διαστήματα. Η έννοια αυτή είχε ως στόχο να εξαλείψει την μελλοντική εμφάνιση προβλημάτων στην κατασκευή ενός έργου λογισμικού. Η Συνεχής Ενσωμάτωση απαιτεί την ύπαρξη της συχνής κατασκευής τμημάτων λογισμικού. Η κοινή πρακτική είναι ότι κάθε φορά που ένα τμήμα κώδικα αλλάζει, μια κατασκευή θα πρέπει να ενεργοποιηθεί. [28]

Τα μέλη μιας ομάδας προγραμματιστών ενσωματώνουν την εργασία τους ανά τακτά χρονικά διαστήματα-συνήθως κάθε άτομο προσθέτει τουλάχιστον καθημερινά- πράγμα το οποίο οδηγεί σε πολλαπλές προσθήκες ανά μέρα. Κάθε ενσωμάτωση είναι έγκυρη από ένα αυτοματοποιημένο έλεγχο που ανιχνεύει όσο πιο σύντομα γίνεται λανθασμένες προσθήκες. Πολλές ομάδες βρίσκουν πως αυτή η προσέγγιση οδηγεί σε δραστικά μειωμένη ένταξη προβλημάτων, ενώ επιτρέπει σε μια ομάδα να αναπτύξει ένα συνεκτικό λογισμικό ακόμα πιο γρήγορα. [28]

2.2.1 Developer's workflow

Η διαδικασία CI υλοποιεί τα αυτοματοποιημένα SW Builds. Ωστόσο, συνήθως η μέθοδος αυτή χρησιμοποιεί αυτοματοποιημένες δοκιμές ανά μονάδα, μετρήσεις για διασφάλιση της ποιότητας και την εξαγωγή αποτελεσμάτων. [28]



Εικόνα 16: Παράδειγμα εγκατάστασης CI υψηλού επιπέδου.

Ο Προγραμματιστής υποβάλλει την αλλαγή του στο κύριο αποθετήριο (repository). Λίγα δευτερόλεπτα αργότερα, ο διακομιστής CI διαπιστώνει τη νέα αλλαγή και αρχίζει τη κατασκευή νέου Build. Η υποβολή αυτή επανεξετάζεται από ορισμένα αυτοματοποιημένα εργαλεία ή από άλλους προγραμματιστές. Μετά την δημιουργία του νέου Build, ο προγραμματιστής ενημερώνεται για την πορεία του (π.χ. ειδοποίηση μέσω ηλεκτρονικού ταχυδρομείου). Συγκεκριμένα, αν το Build είναι χωρίς σφάλματα και οι δοκιμές στις οποίες υποβλήθηκε είναι επιτυχείς, το κομμάτι κώδικα ενσωματώνεται στο κεντρικό αποθετήριο. Σε κάθε άλλη περίπτωση ο κώδικας επιστρέφεται στον προγραμματιστή για διορθώσεις. Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι ο κώδικας να είναι «καθαρός» (χωρίς σφάλματα).

Στη συνέχεια ακολουθούν δύο ρεαλιστικά σενάρια απόπειρας υποβολής αλλαγή κώδικα στο κεντρικό αποθετήριο:

Σενάριο 1: Παραδοσιακός τρόπος ανάπτυξης λογισμικού.

> Ο Προγραμματιστής Α αρχίζει να δουλεύει σε ένα μέρος του λογισμικού, το οποίο εξαρτάται από ορισμένα μέρη του κώδικα/ βιβλιοθήκες άλλου επιπέδου. Ο Προγραμματιστής Β έχει κάνει αλλαγές στα μέρη του κώδικα/ βιβλιοθήκες που χρησιμοποιεί ο Προγραμματιστής Α, αλλά δεν έχει υποβάλει τις αλλαγές του στο κεντρικό αποθετήριο, διότι η ομάδα συνένωσης μερών κώδικα πραγματοποιεί την διαδικασία συνένωσης μία φορά την εβδομάδα. Κατ' αυτόν τον τρόπο, ο Προγραμματιστής Α πρέπει να περιμένει κάποιες μέρες για να ενσωματωθούν οι αλλαγές του Προγραμματιστή Β στο κεντρικό αποθετήριο και να είναι διαθέσιμες σε αυτόν μέσω της ομάδας εκδόσεων (Release Team).

Σενάριο 2: Τρόπος ανάπτυξης λογισμικού με χρήση συστήματος CI.

> Ο Προγραμματιστής Β όταν τελειώσει τις αλλαγές του στα μέρη του κώδικα/ βιβλιοθήκες, υποβάλει τις αλλαγές του στο κεντρικό αποθετήριο. Ο διακομιστής CI αντιλαμβάνεται τις αλλαγές και ξεκινά την δημιουργία και τον έλεγχο ενός «φρέσκου» Build. Όταν το Build έχει ένα ανεκτό ποσοστό επιτυχίας οι αλλαγές είναι έτοιμες προς συγχώνευση. Έπειτα, ο Προγραμματιστής Β ενημερώνει τον Προγραμματιστή Α, ο οποίος μπορεί να ξεκινήσει την ανάπτυξη του μέρους λογισμικού. Ο παραπάνω κύκλος ενσωμάτωσης μπορεί να πραγματοποιηθεί αρκετές φορές μέσα σε μια μέρα.

Όπως γίνεται κατανοητό από τα παραπάνω παραδείγματα σεναρίων, ενσωματώνοντας συχνά κώδικα, μπορεί να επιταχύνει τη διαδικασία ανάπτυξης λογισμικού. Η καλύτερη ποιότητα σε σχέση με τα παραδοσιακά μοντέλα έρχονται σε μια μορφή εκτεταμένης χρήσης των αυτοματοποιημένων ελέγχων και της συνεχής ανατροφοδότησης, τόσο από αυτοματοποιημένα εργαλεία όσο και από την αξιολόγηση από ομότιμους. [28]

2.2.2 Βέλτιστες πρακτικές

Αυτή η ενότητα παραθέτει τις βέλτιστες πρακτικές που προτείνονται από διάφορους συγγραφείς για το πώς θα επιτευχθεί η συνεχής ενσωμάτωση, και πώς μπορεί να αυτοματοποιηθεί αυτή η πρακτική. Η κατασκευή αυτοματοποιημένων SW Builds αποτελεί μία βέλτιστη πρακτική. [29]

Ένας άλλος παράγοντας είναι η ανάγκη για ένα σύστημα ελέγχου εκδόσεων που υποστηρίζει την ατομική κατάθεση κώδικα (Commit). Όλες οι αλλαγές ενός έργου μπορούν να θεωρηθούν ως μία ενιαία κατάθεση κώδικα στο κεντρικό αποθετήριο.

Για την επίτευξη των παραπάνω στόχων, η συνεχής ενσωμάτωση βασίζεται στις ακόλουθες αρχές. [30]

1. Διατήρηση ενός αποθετηρίου κώδικα

Η πρακτική αυτή συνηγορεί υπέρ της χρήσης ενός συστήματος ελέγχου αναθεώρησης, για τον πηγαίο κώδικα του έργου. Όλα τα αντικείμενα που απαιτούνται για την κατασκευή του έργου θα πρέπει να τοποθετηθούν στο αποθετήριο. Σε αυτήν την πρακτική υπάρχει η σύμβαση ότι το σύστημα θα πρέπει να είναι άρτιο και δεν απαιτούνται πρόσθετες εξαρτήσεις.

2. Αυτοματοποίηση των SW Builds

Μια απλή εντολή θα πρέπει να έχει τη δυνατότητα της οικοδόμησης του συστήματος. Η αυτοματοποίηση της κατασκευής θα πρέπει να περιλαμβάνει την αυτοματοποίηση της ολοκλήρωσης, η οποία συχνά περιλαμβάνει την ανάπτυξη σε ένα περιβάλλον παραγωγής. Σε πολλές περιπτώσεις, το σενάριο κατασκευής όχι μόνο δημιουργεί εκτελέσιμα αρχεία, αλλά παράγει επίσης έγγραφα, ιστοσελίδες, στατιστικές και μέσα διανομής των αποτελεσμάτων.

3. Δημιουργία αυτοματοποιημένων ελέγχων

Μόλις ο κώδικας χτιστεί (Build), όλες οι δοκιμές πρέπει να πραγματοποιηθούν για να επιβεβαιωθεί ότι συμπεριφέρεται όπως οι προγραμματιστές τον σχεδίασαν.

4. Κατάθεση κώδικα καθημερινά στην βασική ροή

Καταθέτοντας τακτικά τον κώδικά τους, οι προγραμματιστές, μπορούν να μειώσουν τον αριθμό των αλληλοσυγκρουόμενων αλλαγών. Ο εβδομαδιαίος έλεγχος εργασίας διατρέχει τον κίνδυνο να έρχεται σε αντίθεση με άλλα χαρακτηριστικά που μπορούν να είναι πολύ δύσκολο να επιλυθούν. Η κατάθεση όλων των αλλαγών, τουλάχιστον μία φορά την ημέρα (μία φορά ανά ενσωματωμένο χαρακτηριστικό) θεωρείται μέρος του ορισμού της Συνεχούς Ολοκλήρωσης (CI). Επιπλέον, οι νυχτερινές εκτελέσεις πιο σύνθετων ελέγχων συνιστάται γενικά.

5. Για κάθε κατάθεση κώδικα στη βασική ροή πρέπει να δημιουργείται ένα SWBuild

Το σύστημα θα πρέπει να δεσμεύεται για την σωστή ενσωμάτωση της τρέχουσας έκδοσης εργασίας. Μια κοινή πρακτική είναι η χρήση των αυτομάτων συστημάτων Συνεχούς Ενσωμάτωσης όπου παρατηρούν την συνεχή κατάθεση κώδικα στο αποθετήριο και τότε εκτελούν αυτόματα τη διαδικασία κατασκευής.

6. Διατήρηση της διαδικασίας δημιουργίας σε γρήγορους ρυθμούς

Η κατασκευή πρέπει να ολοκληρώνεται γρήγορα, έτσι ώστε αν υπάρχει πρόβλημα με την ενσωμάτωση, να είναι γρήγορη η ενημέρωση του προγραμματιστή που κατέθεσε τον προβληματικό κώδικα.

7. Δοκιμή σε έναν κλώνο του περιβάλλοντος παραγωγής

Ένα περιβάλλον δοκιμής μπορεί να οδηγήσει σε βλάβες στα δοκιμασμένα συστήματα όταν αναπτυχθούν σε περιβάλλον παραγωγής, διότι το περιβάλλον παραγωγής μπορεί να διαφέρει από το περιβάλλον δοκιμής κατά ένα σημαντικό τρόπο. Συνεπώς, το περιβάλλον δοκιμής, ή ένα ξεχωριστό περιβάλλον προ-παραγωγής θα πρέπει να κατασκευαστεί για να είναι μια κλιμακούμενη έκδοση του πραγματικού περιβάλλοντος παραγωγής (εικονικό εργαστήριο δοκιμών).

8. Να είναι εύκολη η πρόσβαση στα πιο πρόσφατα παραδοτέα στοιχεία

Κάνοντας τα SW Builds άμεσα διαθέσιμα στους ενδιαφερόμενους μειώνονται οι πιθανότητες τα ελαττώματα να επιβιώσουν από την ανάπτυξη μέχρι και την διάθεση. Η εύρεση σφαλμάτων νωρίτερα, σε ορισμένες περιπτώσεις, μειώνει την ποσότητα της εργασίας που χρειάζεται για την επίλυσή τους.

Όλοι οι προγραμματιστές θα πρέπει να ξεκινήσουν τη μέρα τους με την «ανανέωση» του έργου από το αποθετήριο (update). Με αυτόν τον τρόπο, όλοι οι προγραμματιστές θα έχουν την νεότερη έκδοση του έργου μέχρι σήμερα.

9. Ο καθένας να μπορεί να δει τα αποτελέσματα της τελευταίας κατασκευής

Θα πρέπει να ενημερώνονται αυτοί που υπέβαλαν τις αλλαγές στο αποθετήριο για τα αποτελέσματα των ελέγχων ώστε να υπάρχει άμεση αντιμετώπιση σε πιθανό σφάλμα δοκιμής.

10. Αυτοματοποίηση εγκατάστασης

Τα περισσότερα συστήματα CI επιτρέπουν την εκτέλεση εντολών μετά από την δημιουργία του Build. Στις περισσότερες περιπτώσεις, είναι δυνατόν να γραφτεί ένα σενάριο για τον έλεγχο του Build. Μια άλλη πρόοδος σε αυτόν τον τρόπο σκέψης είναι η συνεχής ανάπτυξη, η οποία ζητά το λογισμικό να αναπτυχθεί άμεσα στην παραγωγή, συχνά με επιπλέον αυτοματισμό για την πρόληψη ελαττωμάτων ή παλινδρομήσεις. [31] [32]

2.2.3 Πλεονεκτήματα του CI

Η χρήση της Συνεχούς Ενσωμάτωσης σε έργα λογισμικού έχει σκοπό να παρέχει οφέλη, όπως [28]:

- Τα σφάλματα ενσωμάτωσης ανιχνεύονται νωρίς και είναι εύκολο να εντοπιστούν λόγω των μικρών αλλαγών που προστίθενται σε κάθε ενσωμάτωση. Αυτό εξοικονομεί χρόνο και χρήμα κατά τη διάρκεια ζωής του έργου.
- Δημιουργία αναπτύξιμου λογισμικού ανά πάσα στιγμή χωρίς σφάλματα.
- Συμβάλει στην καλύτερη ορατότητα του έργου, αφού η συνεχής ενσωμάτωση προάγει την δημιουργία λιγότερο περίπλοκου κώδικα.
- Εγκαθιδρύει μεγαλύτερη εκτίμηση στο παραγόμενο προϊόν λογισμικού από την ομάδα που το αναπτύσσει.

Επιπλέον, με την χρήση αυτοματοποιημένης Συνεχούς Ενσωμάτωσης προστίθενται τα παρακάτω οφέλη, όπως:

- Επιβάλλεται η πειθαρχία των συχνών αυτοματοποιημένων δοκιμών. Κατ' αυτόν τον τρόπο όλοι οι προγραμματιστές είναι πιο προσεκτικοί κατά την δημιουργία του κώδικα.
- Υπάρχει άμεση ανατροφοδότηση σχετικά με τα αποτελέσματα των αλλαγών, είτε αυτά είναι θετικά, είτε είναι αρνητικά.
- Κατά την αυτοματοποιημένη δοκιμή, δημιουργούνται μετρήσεις, όπως μετρήσεις για την κάλυψη του κώδικα και την πολυπλοκότητά του, οι οποίες βοηθούν

τους προγραμματιστές να εστιάσουν στην ανάπτυξη λειτουργικών εφαρμογών και στην καλύτερη ποιότητα κώδικα. [33]

Για την κατασκευή ενός αυτοματοποιημένου συστήματος Συνεχούς Ενσωμάτωσης απαιτείται ένα σημαντικό ποσό εργασίας, συμπεριλαμβανομένης της συνεχούς προσπάθειας για να καλυφθούν τα νέα χαρακτηριστικά. Η Δοκιμή θεωρείται βέλτιστη πρακτική για την ανάπτυξη λογισμικού από μόνη της, ανεξάρτητα από το αν ή όχι η συνεχής ενσωμάτωση χρησιμοποιείται, και η αυτοματοποίηση αποτελεί αναπόσπαστο μέρος της μεθοδολογίας του έργου, όπως δοκιμή με γνώμονα την ανάπτυξη.

Με ένα σύστημα CI, στο έργο λογισμικού δεν μειώνονται απλώς οι κίνδυνοι που υπάρχουν σε μία φάση ενσωμάτωσης μεγάλου τμήματος κώδικα, αλλά εξαλείφεται ολόκληρη η παραπάνω φάση ενσωμάτωσης. Επιπλέον, στο έργο λογισμικού εξαλείφονται πολλά προβλήματα που αφορούν την προβλεψιμότητα, διότι συνήθως είναι πολύ δύσκολο να είναι γνωστό το χρονικό διάστημα που θα διαρκέσει η φάση ολοκλήρωσης.

Σχετικά με τα σφάλματα που εισάγονται κατά την φάση υλοποίησης, το CI δεν βοηθά στην απαλλαγή από τα σφάλματα εντελώς, αλλά καθιστά πολύ πιο εύκολη την ανεύρεσή τους και την αφαίρεσή τους. Επίσης, η συσσώρευση σφαλμάτων στο έργο λογισμικού, καθιστά την ανεύρεσή τους και την αφαίρεσή τους σε μεταγενέστερα στάδια ανάπτυξης του έργου σχεδόν αδύνατη.

Επιπλέον, το CI ενεργοποιεί την συνεχή ανάπτυξη του έργου, η οποία είναι πολύ σημαντική διότι επιτρέπει στους χρήστες να πάρουν περισσότερα χαρακτηριστικά(Features) γρήγορα. Έτσι, επιτρέπεται η έγκαιρη ανάδραση από τους χρήστες στους προγραμματιστές και κατά αυτόν τον τρόπο «σπάνε» τα εμπόδια μεταξύ της ανάπτυξης και της ικανοποίησης των απαιτήσεων των πελατών.

2.2.4 Εργαλεία για την υλοποίηση CI συστήματος

Η Συνεχής Ολοκλήρωσης προέρχεται από τις πρακτικές του Ακραίου Προγραμματισμού, ή αλλιώς Extreme Programming (XP), από τα τέλη του 1990. Ωστόσο, παρόμοιες προσεγγίσεις χρησιμοποιήθηκαν νωρίτερα σε κρίσιμα έργα λογισμικού, για παράδειγμα, από τη NASA. Οι πρώτοι που υιοθέτησαν την συνεχή ενσωμάτωση ήταν ο Kent Beck και ο Martin Fowler, οι οποίοι δημοσίευσαν διάσημο υλικό σχετικά με το αντικείμενο (το βιβλίο του Beck "Extreme Programming Explained" και του Φάουλερ " Practices of Continuous Integration”).

Τα πράγματα έχουν εξελιχθεί πολύ από τότε, και τώρα έχουμε πρόσβαση σε μια πληθώρα εργαλείων για την κατασκευή ενός συστήματος συνεχούς ενσωμάτωσης. Αρκετά από αυτά τα εργαλεία είναι δωρεάν και ανοιχτού κώδικα.

Κάθε σύστημα που καλύπτει ανάγκες, για να υλοποιηθεί και να είναι αποδοτικό χρειάζεται τα απαραίτητα εργαλεία για την δημιουργία του. Συμφωνά με τα παραπάνω η υλοποίηση ενός συστήματος συνεχούς ενσωμάτωσης κρίνεται απαραίτητη. Ιδιαιτέρως όταν αυτό είναι αυτοματοποιημένο. Για την δημιουργία ενός τέτοιου συστήματος δύο ειδών είναι τα κύρια εργαλεία:

1. Οι Διαχειριστές Εκδόσεων Κεντρικού Αποθετηρίου
2. Οι Διακομιστές Συνεχούς Ενσωμάτωσης

Στη συνέχεια θα παρουσιαστούν μερικά από τα βασικότερα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση του CI Συστήματος της παρούσας εργασίας.

2.2.4.1 Συστήμα Ελέγχου Εκδόσεων Κεντρικού Αποθετηρίου

Τα Συστήματα Ελέγχου Εκδόσεων είναι ένα λογισμικό που βοηθά τους προγραμματιστές λογισμικού να συνεργαστούν και να διατηρήσουν ένα πλήρες ιστορικό της εργασίας τους.

Παρακάτω αναφέρονται οι στόχοι ενός Συστήματος Ελέγχου Εκδόσεων.

- Επιτρέπει στους προγραμματιστές να λειτουργούν ταυτόχρονα.
- Δεν επιτρέπει την αντικατάσταση των αλλαγών από άλλους προγραμματιστές.
- Διατηρεί ιστορικό της κάθε έκδοσης.

Τα Συστήματα Ελέγχου Εκδόσεων είναι διαιρεμένα σε δύο κατηγορίες.

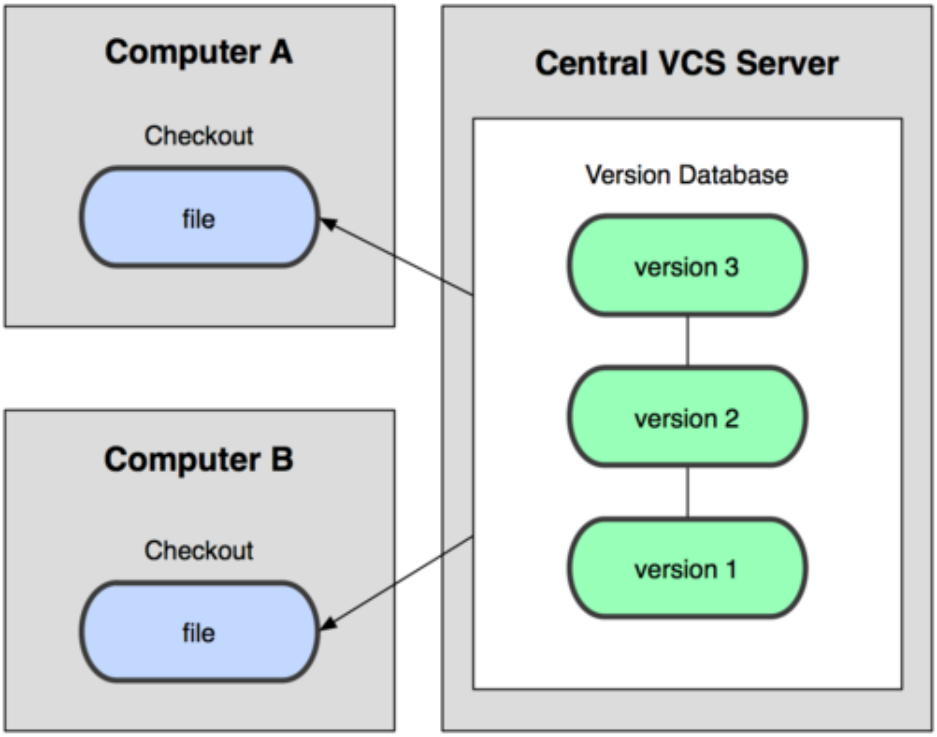
1. Τα Κεντρικοποιημένα Συστήματα Ελέγχου Εκδόσεων (Centralized Version Control System).
2. Τα Κατανεμημένα Συστήματα Ελέγχου Εκδόσεων (Distributed/Decentralized Version Control System).

Στη συνέχεια θα παρουσιαστούν τα δύο πιο δημοφιλή στην αγορά Συστήματα Ελέγχου Εκδόσεων, το Subversion και το Git.

2.2.4.1.1 Subversion

Ένα σημαντικό θέμα που αντιμετωπίζουν οι άνθρωποι είναι ότι πρέπει να συνεργαστούν με τους προγραμματιστές άλλων συστημάτων. Για την αντιμετώπιση αυτού του προβλήματος αναπτύχθηκαν τα Κεντρικοποιημένα Συστήματα Ελέγχου Εκδόσεων (CVCS). Αυτά τα συστήματα, όπως τα CVS, Subversion, και Perforce, έχουν ένα μόνο διακομιστή που περιέχει όλα τα αρχεία εκδόσεων, καθώς και τον αριθμό των πελατών που αντλούν αρχεία από αυτό το κεντρικό σημείο. Για πολλά χρόνια, αυτό ήταν το πρότυπο για τον έλεγχο των εκδόσεων. [34]

Παρακάτω προβάλλεται η βασική δομή ενός κεντρικοποιημένου συστήματος ελέγχου εκδόσεων.



Εικόνα 17: Βασική δομή ενός κεντρικοποιημένου συστήματος ελέγχου εκδόσεων. [34]

Το Subversion, ή αλλιώς SVN, είναι ένα Κεντρικοποιημένο Σύστημα Ελέγχου Εκδόσεων. Τα Κεντικοποιημένα Σύστηματα Ελέγχου Εκδόσεων βασίζονται στην ιδέα ότι υπάρχει ένα ενιαίο «κεντρικό» αντίγραφο του έργου κάπου (πιθανώς σε ένα server), και οι προγραμματιστές θα «δεσμεύσουν» (Commit) τις αλλαγές τους σε αυτό το κεντρικό αντίγραφο.

"Δεσμεύοντας" τις αλλαγές, καταγράφονται οι μεταβολές στο κεντρικό σύστημα. Άλλοι προγραμματιστές μπορούν στη συνέχεια να δουν την αλλαγή αυτή. Μπορούν επίσης να αναιρέσουν τις αλλαγές και το εργαλείο ελέγχου έκδοσης θα ενημερώσει αυτόματα τα περιεχόμενα των αρχείων που είχαν αλλάξει. Οι προγραμματιστές δεν χρειάζεται πλέον να διατηρούν πολλά αντίγραφα των αρχείων τους στο σκληρό δίσκο χειροκίνητα, επειδή το εργαλείο ελέγχου εκδόσεων μπορεί να επικοινωνεί με το κεντρικό αντίγραφο και να ανακτά κάθε έκδοση που απαιτείται.

Σε ένα κεντρικό σύστημα ελέγχου εκδόσεων, η ροή εργασιών για την προσθήκη ενός νέου χαρακτηριστικού ή για τον καθορισμό ενός σφάλματος (bug) στο έργο ακολουθεί την παρακάτω διαδικασία [35]:

- Αντιγραφή του αρχείου που βρίσκεται από το κεντρικό αποθετήριο (Server-Repository) τοπικά στον υπολογιστή του προγραμματιστή.
- Εισαγωγή των αλλαγών στο αντιγραμμένο αρχείο και έλεγχος για την ομαλή λειτουργία του.
- Δέσμευση των αλλαγών στο κεντρικό αποθετήριο. Έτσι, οι υπόλοιποι προγραμματιστές μπορούν να δουν τις αλλαγές που υπέστη το αρχείο.

Στη συνέχεια θα παρουσιαστούν τα βασικά συστατικά και έννοιες των Κεντρικοποιημένων Συστημάτων Ελέγχου Εκδόσεων.

Version Control Terminologies

The repository

Ένας χώρος αποθήκευσης είναι η καρδιά του κάθε συστήματος ελέγχου έκδοσης. Είναι το κεντρικό σημείο όπου οι προγραμματιστές έχουν αποθηκεύσει όλες τις εργασίες τους. Στο αποθετήριο δεν αποθηκεύονται μόνο τα αρχεία, αλλά και το ιστορικό τους.. Το αποθετήριο είναι προσβάσιμο μέσω δικτύου, λειτουργώντας ως εργαλείο τύπου διακομιστή και το εργαλείο ελέγχου εκδόσεων ενεργεί ως πελάτης. Οι πελάτες μπορούν να συνδεθούν στο χώρο αποθήκευσης, και στη συνέχεια μπορούν να αποθηκεύουν/ ανακτούν τις αλλαγές τους προς/ από το αποθετήριο. Με την αποθήκευση αλλαγών, ένας πελάτης κάνει αυτές τις αλλαγές ορατές στους άλλους ανθρώπους. Οι υπόλοιποι πελάτες ανακτώντας τις αλλαγές, λαμβάνουν αλλαγές άλλων ανθρώπων ως ένα αντίγραφο εργασίας.

Trunk

Ο κορμός ή Trunk είναι ένας κατάλογος, όπου εκεί συμβαίνει όλη η κύρια ανάπτυξη του έργου και συνήθως ελέγχεται από τους προγραμματιστές που εργάζονται για το έργο.

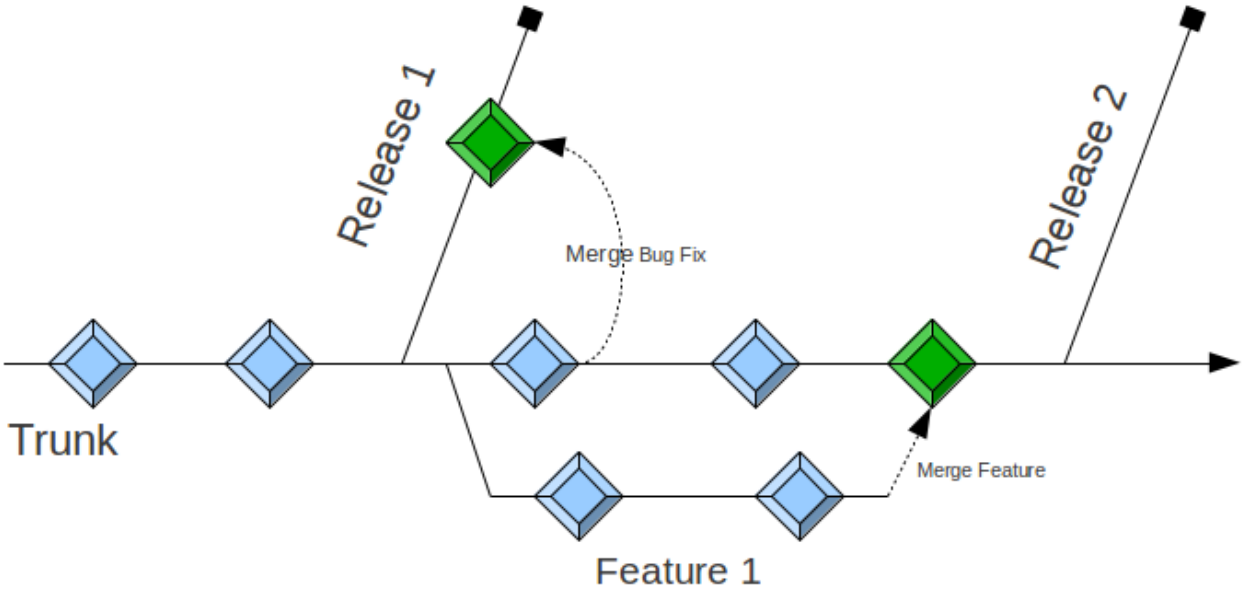
Tags

Ο κατάλογος ετικέτες, ή αλλιώς Tags, χρησιμοποιείται για να αποθηκεύσει στιγμιότυπα του έργου με συγκεκριμένο όνομα. Η λειτουργία Tag επιτρέπει να δώσει ονόματα βάση περιγραφικής και μνημονικού για τη συγκεκριμένη έκδοση στο αποθετήριο.

Branches

Η λειτουργία Κλαδιού ή Branch, χρησιμοποιείται για να δημιουργηθεί μια άλλη γραμμή ανάπτυξης. Αυτό είναι χρήσιμο όταν χρειάζεται αναπτυξιακή διαδικασία σε δύο διαφορετικές κατευθύνσεις. [37]

Για παράδειγμα, γίνεται η ανάπτυξη του έργου στο Trunk όπως φαίνεται στο παρακάτω σχήμα. Όταν κυκλοφορήσει η έκδοση 1 (Release 1), μπορεί να χρειάζεται να δημιουργηθεί ένα κλαδί (branch) ώστε η ανάπτυξη της έκδοσης 2 (Release 2) να έχει χαρακτηριστικά που μπορούν να φυλάσσονται ξεχωριστά από την διορθωμένη έκδοση 1 (Release 1 bug-fixes). Στο συγκεκριμένο παράδειγμα, τα νέα χαρακτηριστικά της δεύτερης έκδοσης αναπτύσσονται σε ένα κλαδί με ονομασία «Feature 1».



Εικόνα 18: Παράδειγμα ανάπτυξης λογισμικού σε σύστημα ελέγχου εκδόσεων. [34]

Working copy

Το Αντίγραφο Εργασίας, ή Working Copy, είναι ένα στιγμιότυπο του αποθετηρίου. Το αποθετήριο είναι κοινό για όλες τις ομάδες εργασίας, αλλά τα μέλη των ομάδων δεν το τροποποιούν άμεσα. Παρόλα' αυτά, κάθε προγραμματιστής(μέλος της ομάδας) μπορεί να ελέγχει το αντίγραφο εργασίας. Το αντίγραφο εργασίας είναι ένας ιδιωτικός χώρος εργασίας, όπου οι προγραμματιστές μπορούν να κάνουν τη δουλειά τους και να παραμένει απομονωμένη από την υπόλοιπη ομάδα μέχρι να δεσμεύσουν (Committing) τις αλλαγές τους. [36]

Commit changes

Η υποβολή των αλλαγών, ή Commit, είναι μια διαδικασία της αποθήκευσης των αλλαγών από τον ιδιωτικό χώρο εργασίας στο αποθετήριο. Μετά την τέλεση, οι αλλαγές που έγιναν είναι διαθέσιμες σε όλα τα μέλη της ομάδας. Οι προγραμματιστές-μέλη μπορούν να ανακτήσουν αυτές τις αλλαγές με ενημέρωση των αντιγράφων τους. Η υποβολή είναι μια ατομική εργασία. Είτε το σύνολο της υποβολής οριστικοποιείται είτε αναιρείται (επανέρχεται στην προηγούμενη έκδοση). Τέλος, οι χρήστες δεν βλέπουν ημιτελή υποβολή. [38]

2.2.4.1.2 Git

Κατανεμημένα Συστήματα Ελέγχου Εκδόσεων

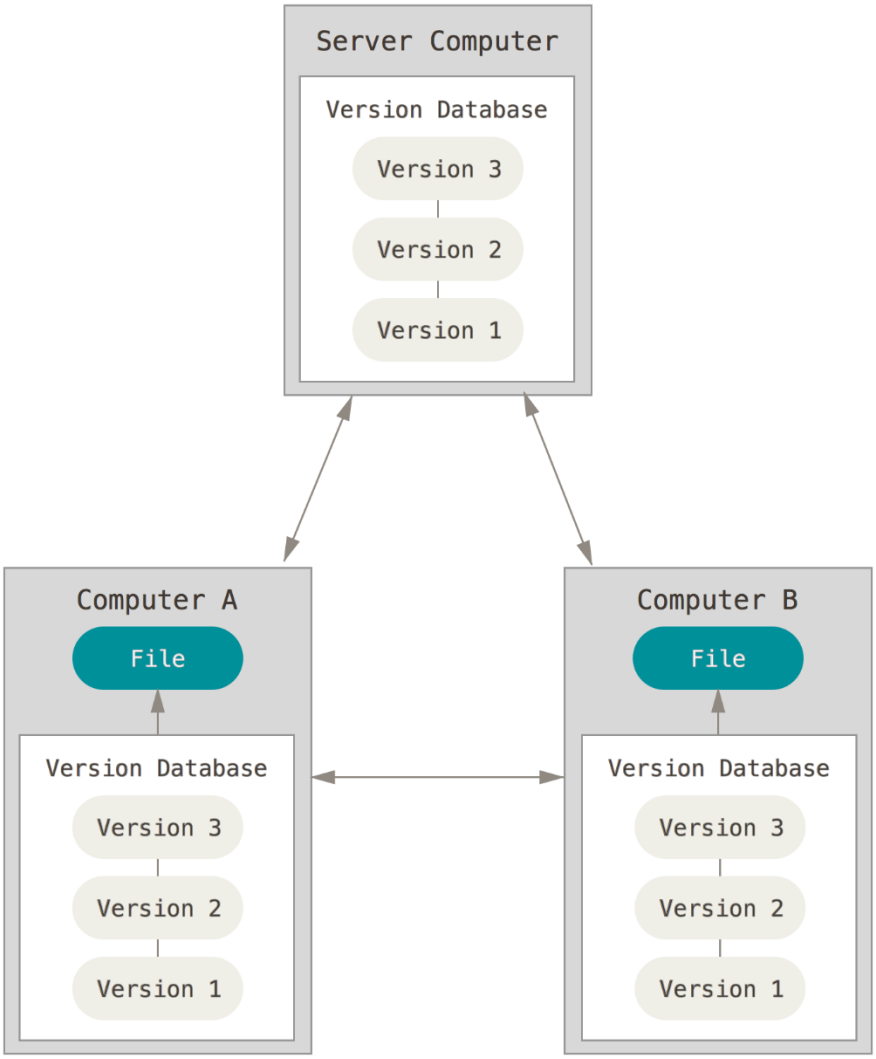
Τα Κεντρικά Συστήματα Ελέγχου Εκδόσεων (CVCs) χρησιμοποιούν έναν κεντρικό διακομιστή για την αποθήκευση όλων των αρχείων και κατ' αυτόν τον τρόπο επιτρέπουν την ομαδική συνεργασία. Το μεγαλύτερο μειονέκτημα των παραπάνω συστημάτων είναι το ενιαίο σημείο της αποτυχίας, δηλαδή, η αποτυχία του κεντρικού διακομιστή. Δυστυχώς, εάν ο κεντρικός διακομιστής ενδέχεται να βγει εκτός λειτουργίας για μία ώρα, κατά τη διάρκεια της

ώρας αυτής, κανείς δεν μπορεί να χρησιμοποιήσει το όλο σύστημα. Επιπλέον, στην χειρότερη περίπτωση, εάν ο σκληρός δίσκος του κεντρικού διακομιστή δημιουργήσει αλλοιωμένο αντίγραφο ασφαλείας των αρχείων, τότε θα χάσει ολόκληρο το ιστορικό των αρχείων και κατά συνέπεια και του έργου. Αντίθετα, τα Κατανεμημένα Συστήματα Ελέγχου Εκδόσεων (DVCS) δεν έχουν τέτοιου είδους προβλήματα.

Οι προγραμματιστές που χρησιμοποιούν Κατανεμημένα Συστήματα Ελέγχου Εκδόσεων δεν ενημερώνονται μόνο για τις τελευταίες αλλαγές του αρχείου που δουλεύουν, αλλά ολόκληρου του αποθετηρίου. Αν ο κεντρικός διακομιστής έρθει εκτός λειτουργίας, τότε το αποθετήριο από κάθε πελάτη/ προγραμματιστή μπορεί να αντιγραφεί στο διακομιστή για να το επαναφέρει στην αρχική κατάστασή του πριν την διακοπή λειτουργίας του. Κάθε ολοκλήρωση κλωνοποίησης (checkout) είναι ένα πλήρες αντίγραφο ασφαλείας του χώρου αποθήκευσης.

Το Git είναι ένα Κατανεμημένο Σύστημα Ελέγχου Εκδόσεων. Δεν στηρίζεται στον κεντρικό εξυπηρετητή και αυτός είναι ο λόγος για τον οποίο μπορούν να εκτελεστούν πολλές λειτουργίες όταν βρίσκεται εκτός σύνδεσης κάποιο μέλος της ομάδας. Μπορούν να δεσμευτούν αλλαγές, να δημιουργηθούν κλαδιά (Branches) και να εκτελεστούν άλλες λειτουργίες σε κατάσταση εκτός σύνδεσης. Για να δημοσιευθούν οι αλλαγές που έγιναν στο διάστημα «εκτός σύνδεσης» και να ενημερωθεί το υπόλοιπο αποθετήριο (τοπικό) απαιτείται σύνδεση. [39]

Παρακάτω προβάλλεται η βασική δομή ενός κατανεμημένου συστήματος ελέγχου εκδόσεων.



Εικόνα 19: Βασική δομή ενός κατανεμημένου συστήματος ελέγχου εκδόσεων.[34]

Στη συνέχεια θα παρουσιαστούν τα βασικά συστατικά και έννοιες των Κατανεμημένων Συστημάτων Ελέγχου Εκδόσεων.

DVCS Terminologies

Local Repository

Το Τοπικό Αποθετήριο ή αλλιώς Local Repository, αποτελεί ένα ιδιωτικό «χώρο» εργασίας. Οι Προγραμματιστές κάνουν αλλαγές στον ιδιωτικό χώρο εργασίας τους και μετά την δέσμευση, αυτές οι αλλαγές γίνονται ένα μέρος του κεντρικού αποθετηρίου. Το Git παρέχει στους προγραμματιστές ένα ιδιωτικό αντίγραφο ολόκληρου του αποθετηρίου. Οι χρήστες μπορούν να εκτελέσουν πολλές λειτουργίες σε αυτό το αποθετήριο, όπως πρόσθεση αρχείου, αφαίρεση το αρχείου, μετονομασία του αρχείου, μετακίνηση του αρχείο, δέσμευση των αλλαγών και πολλά άλλα.

Working Directory and Staging Area or Index

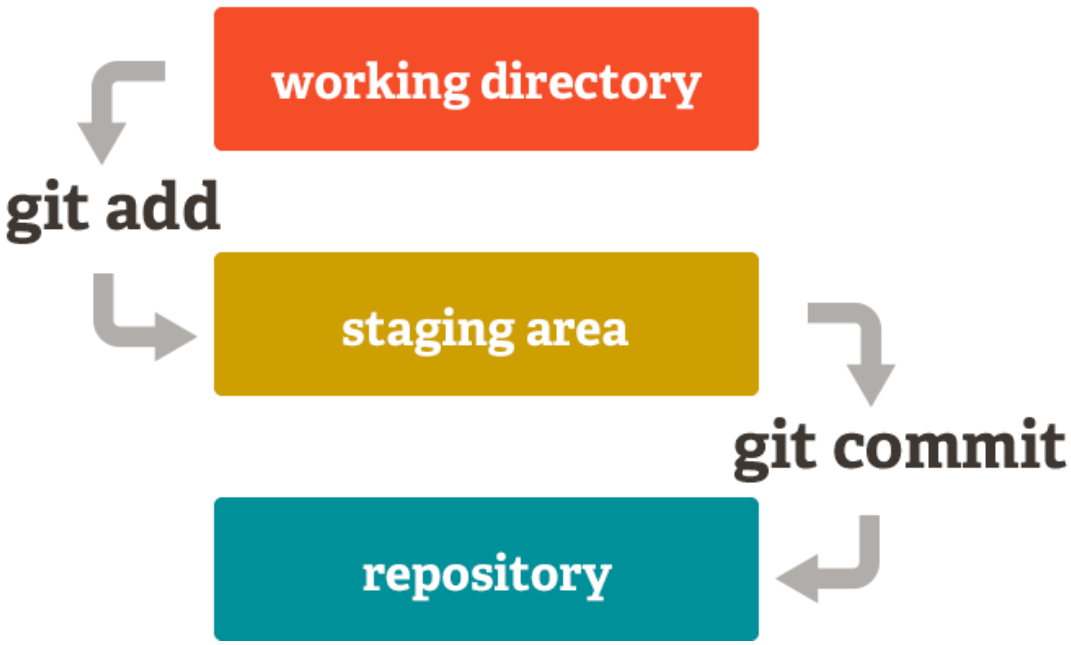
Ο κατάλογος εργασίας, ή αλλιώς Working Directory, είναι ο τόπος όπου τα αρχεία αποθηκεύονται (checked out). Σε άλλα Συστήματα Ελέγχου Εκδόσεων (CVCs), οι προγραμματιστές κυρίως κάνουν τροποποιήσεις και δεσμεύουν τις αλλαγές τους απευθείας στο αποθετήριο, το οποίο είναι κεντρικό. Το Git όμως, χρησιμοποιεί μια διαφορετική στρατηγική. Πιο συγκεκριμένα, το Git δεν παρακολουθεί κάθε τροποποιημένο αρχείο. Κάθε φορά που πραγματοποιείται μια δέσμευση (Commit), το Git αναζητά τα τρέχοντα αρχεία που υπάρχουν στο χώρο ανεφοδιασμού (staging area). Για την πραγματοποίηση της δέσμευσης (του Commit), μόνο τα αρχεία που υπάρχουν στο χώρο ανεφοδιασμού λαμβάνονται υπόψη και όχι όλα τα τροποποιημένα αρχεία του χρήστη. [38]

Η βασική ροή του Git είναι η εξής:

Βήμα 1ο: Τροποποίηση ενός αρχείου από τον κατάλογο εργασίας.

Βήμα 2ο: Πρόσθεση του παραπάνω τροποποιημένου αρχείου στο χώρο ανεφοδιασμού.

Βήμα 3ο: Πραγματοποίηση της δέσμευσης του αρχείου (από το χρήστη), η οποία μετακινεί το αρχείο από το χώρο ανεφοδιασμού. Μετά την μετακίνηση αυτή, αποθηκεύονται μόνιμα οι αλλαγές του αρχείου στο αποθετήριο του Git.



Εικόνα 20: Λειτουργία κατανεμημένων συστημάτων ελέγχου εκδόσεων. [34]

Blobs

Κάθε έκδοση ενός αρχείου αντιπροσωπεύεται από μία σταγόνα, ή αλλιώς ένα blob. Ένα blob διατηρεί τα δεδομένα του αρχείου αλλά δεν περιέχει μετα-δεδομένα (metadata) σχετικά με το αρχείο. Είναι ένα δυαδικό αρχείο, και στη βάση δεδομένων του Git, ονομάζεται ως «SHA1

hash» του αρχείου. Στο Git, τα αρχεία δεν αποκτούν διεύθυνση με βάση τα ονόματα αλλά με βάση το περιεχόμενο.

Trees

Δέντρο, ή αλλιώς Tree, είναι ένα αντικείμενο το οποίο αντιπροσωπεύει έναν κατάλογο ο οποίος περιέχει Blobs, καθώς και άλλους υπό-καταλόγους. Ένα δέντρο είναι ένα δυαδικό αρχείο που αποθηκεύει αναφορές σε Blobs.

Commits

Το Commit (η δέσμευση) «παγώνει» την τρέχουσα κατάσταση του αποθετηρίου. Μια δέσμευση ονομάζεται επίσης «SHA1 hash». Μια δέσμευση παρομοιάζεται ως κόμβος μιας συνδεδεμένης λίστας. Κάθε δέσμευση έχει ένα δείκτη προς τη μητρική δέσμευση. Από έναν κόμβο-δέσμευση είναι εμφανές το ιστορικό των δεσμεύσεων που έχουν γίνει. Εάν μία δέσμευση έχει πολλαπλά γονικά στοιχεία, τότε η συγκεκριμένη δέσμευση έχει δημιουργηθεί από τη συγχώνευση δύο κλάδων (Branches). [39]

Branches

Τα Κλαδιά, ή αλλιώς Branches, χρησιμοποιούνται για να δημιουργήσουν μια άλλη γραμμή ανάπτυξης του έργου. Από προεπιλογή, το Git έχει ένα κύριο κλάδο, τον Master, ο οποίος είναι αντίστοιχος με το Trunk του Subversion. Συνήθως, ένας κλάδος δημιουργείται για να αναπτυχθεί ένα νέο χαρακτηριστικό του έργου. Μόλις το χαρακτηριστικό έχει ολοκληρωθεί, συγχωνεύεται (Merge) με τον κύριο κλάδο και διαγράφεται το κλαδί. Κάθε κλαδί αναφέρεται από μία Κεφαλή, ή αλλιώς HEAD, η οποία λειτουργεί ως δείκτης στην τελευταία δέσμευση του κλαδιού. Κάθε φορά που πραγματοποιείται μια δέσμευση, η κεφαλή είναι ενημερωμένη με τις τελευταίες αλλαγές.

Tags

Μία Ετικέτα, ή αλλιώς ένα Tag, αντιστοιχίζει ένα όνομα με μια συγκεκριμένη έκδοση στο αποθετήριο. Οι ετικέτες είναι παρόμοιες με τα Κλαδιά, αλλά η διαφορά τους είναι ότι οι Ετικέτες είναι αμετάβλητες. Αυτό σημαίνει ότι μία ετικέτα είναι ένα κλαδί, το οποίο κανείς δεν πρόκειται να τροποποιήσει. Μόλις μια ετικέτα δημιουργηθεί για μία συγκεκριμένη δέσμευση, οποιαδήποτε άλλη νεότερη δέσμευση δεν θα μεταβάλλει την προηγούμενη. Συνήθως, οι προγραμματιστές δημιουργούν ετικέτες όταν πρόκειται να δημιουργηθεί έκδοση του προϊόντος.

Clone

Η λειτουργία Κλωνοποίησης, ή αλλιώς Cloning, δημιουργεί ένα πιστό αντίγραφο του αποθετηρίου. Η παραπάνω λειτουργία δεν δημιουργεί μόνο το αντίγραφο εργασίας, αλλά αντικατοπτρίζει το πλήρες αποθετήριο. Οι χρήστες μπορούν να εκτελέσουν πολλές λειτουργίες με αυτό το τοπικό αποθετήριο. Η μόνη δικτύωση (networking) που εμπλέκεται είναι όταν τα αποθετήρια ολόκληρης της ομάδας συγχρονίζονται.

Pull

Η λειτουργία έλξης, ή αλλιώς Pull, αντιγράφει τις αλλαγές άλλων απομακρυσμένων αποθετηρίων σε ένα τοπικό αποθετήριο. Η παραπάνω λειτουργία χρησιμοποιείται για το συγχρονισμό δύο αποθετηρίων.

Push

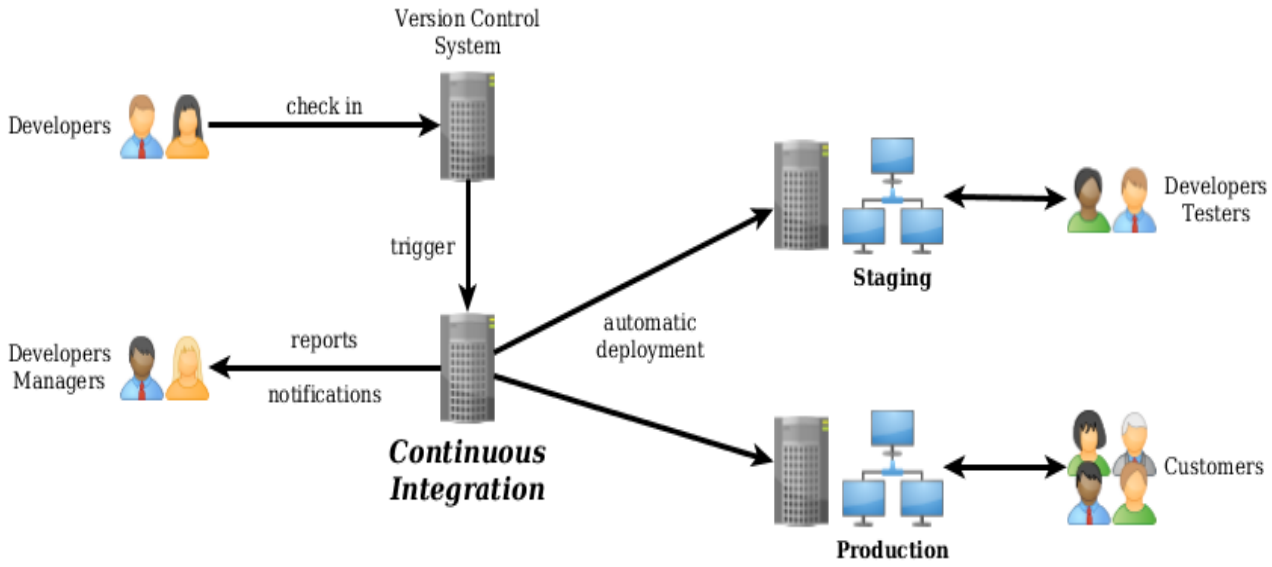
Η λειτουργία Ώθησης, ή αλλιώς Push, αντιγράφει τις αλλαγές από ένα τοπικό αποθετήριο σε ένα απομακρυσμένο αποθετήριο. Η συγκεκριμένη λειτουργία χρησιμοποιείται για την μόνιμη αποθήκευση των αλλαγών στο αποθετήριο Git. Επιπλέον, η λειτουργία Ώθησης είναι παρόμοια με την λειτουργία Δέσμευσης (Commit) στο Subversion. [34] [35][37]

2.2.4.2 Διακομιστές Συνεχούς Ενσωμάτωσης (CI Servers)

Οι Διακομιστές Συνεχούς Ενσωμάτωσης, ή αλλιώς CI Servers, είναι λογισμικό διακομιστών που βοηθά τους προγραμματιστές λογισμικού να πραγματοποιήσουν το έλεγχο της συνένωσης των εργασιών τους. Παρακάτω αναφέρονται οι στόχοι των Διακομιστών Συνεχούς Ενσωμάτωσης.

- Έλεγχος του αποθετηρίου για αλλαγές των αρχείων. Αυτό μπορεί να γίνει σε ένα κανονικό πρόγραμμα (π.χ. κάθε μία ώρα), όταν εντοπιστεί κάποια αλλαγή ή χειροκίνητα.
- Κατασκευή ή μεταγλώττιση του κώδικα. Προϋπόθεση απαραίτητη είναι αυτή η διαδικασία να μπορεί να υποστηριχθεί χωρίς κάποια ανθρώπινη παρέμβαση (αυτόματα).
- Εκτέλεση των δοκιμών χωρίς ανθρώπινη παρέμβαση.
- Ειδοποιήσεις στους ενδιαφερόμενους (προγραμματιστές, managers) εάν εντοπίσουν τυχόν προβλήματα.
- Συγκέντρωση των μετρήσεων ανάλογα με την περίπτωση δοκιμών. [40]

Παρακάτω ακολουθεί σχηματική αναπαράσταση ενός CI Συστήματος.



Εικόνα 21: Σχηματική αναπαράσταση CI συστήματος.

Οι Διακομιστές Συνεχούς Ενσωμάτωσης διαχωρίζονται σε δύο κατηγορίες:

1. Φιλοξενούμενους Διακομιστές Συνεχούς Ενσωμάτωσης (Hosted CI Servers).
2. Αυτό-Φιλοξενούμενους Διακομιστές Συνεχούς Ενσωμάτωσης (Self-Hosted CI Servers).

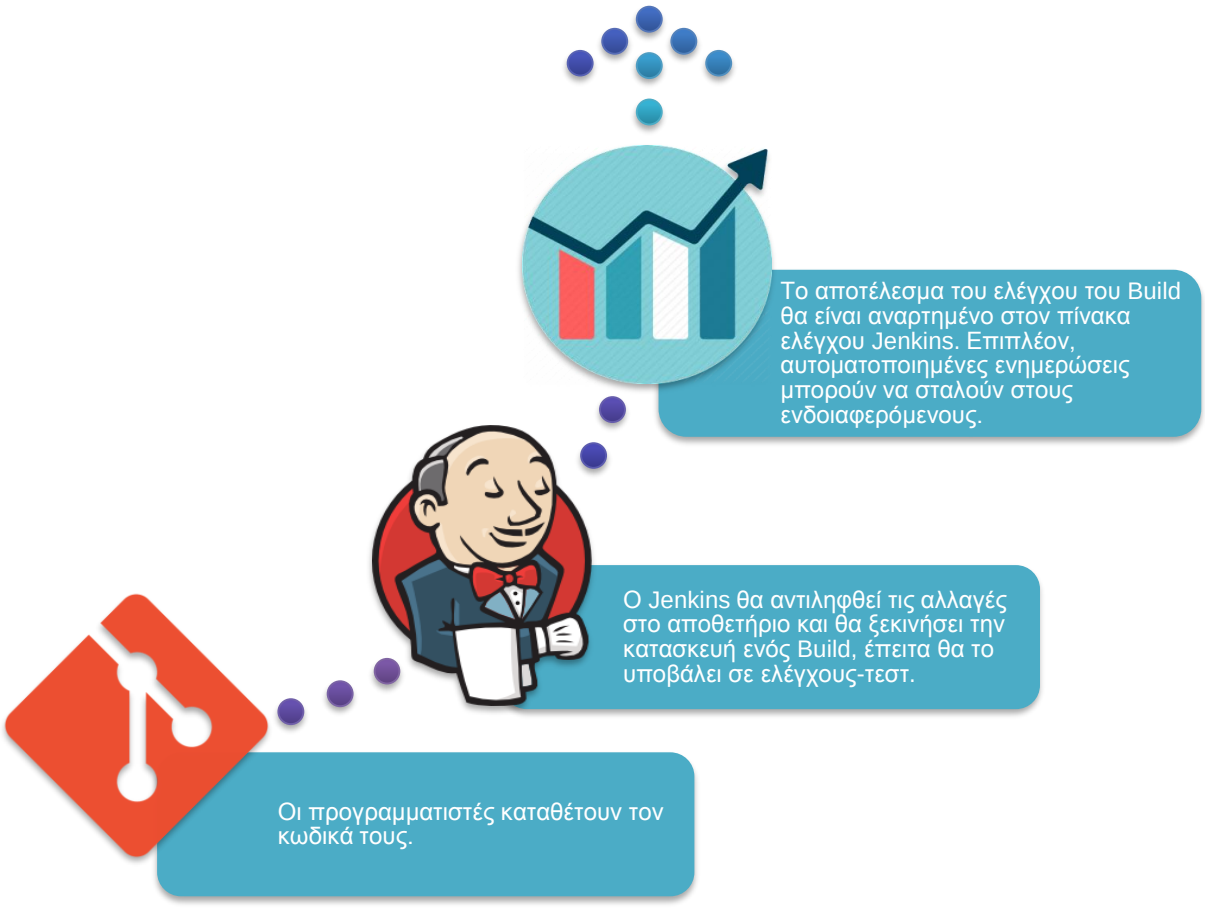
Στη συνέχεια θα παρουσιαστούν οι δύο πιο δημοφιλής στην αγορά Διακομιστές Συνεχούς Ενσωμάτωσης, ο Jenkins και ο Bamboo.

2.2.4.2.1 Jenkins

Ο Jenkins ανήκει στην κατηγορία των Αυτό-Φιλοξενούμενων Διακομιστών Συνεχούς Ενσωμάτωσης, ή αλλιώς non-Hosted CI Servers. Δηλαδή, εγκαθίσταται σε έναν διακομιστή (server) όπου ανήκει στην ομάδα εργασίας.

Ο Jenkins είναι ένα λογισμικό «ανοιχτού κώδικα» που δημιουργήθηκε από τον Kohsuke Kawaguchi το 2004 με σκοπό την συνεχή ενσωμάτωση(Continuous Integration). Ο Jenkins είναι σήμερα ο πιο δημοφιλής διακομιστής CI με πάνω από το 49% του μεριδίου της αγοράς.[41] Η δημοτικότητά του μπορεί να εξηγηθεί λόγω του χαμηλού κόστους (ελεύθερο προς χρήση) και της επεκτασιμότητας που υποστηρίζει. [42]

Το παρακάτω διάγραμμα δείχνει μια πολύ απλή ροή εργασίας για το πώς λειτουργεί Jenkins.



Εικόνα 22: Διάγραμμα ροής λειτουργίας του Jenkins.

Στον Jenkins η κεντρική ιδέα είναι γύρω από μία ρυθμιζόμενη εργασία, ή αλλιώς Job, που μπορεί με τη σειρά της να προκαλέσει την εκτέλεση άλλων εργασιών. Κάθε εκτέλεση μιας εργασίας αναφέρεται σε μία Κατασκευή, ή αλλιώς Build, με ένα μοναδικό αναγνωριστικό αριθμό. Κάθε εργασία αποτελείται από καμία ή περισσότερες προ-Κατασκευές (pre-Build), Κατασκευές (Builds) ή μετά-Κατασκευές (post-Build) βημάτων . Ένα βήμα Κατασκευής, ή αλλιώς Build step, μπορεί να είναι οποιαδήποτε επιθυμητή αυτοματοποιημένη εργασία, από την εκτέλεση ενός Shell Script μέχρι κάποιο artifact deployment..

Επιπλέον, ο Jenkins διαθέτει μία δυνατότητα που δημιουργεί κατανεμημένα Builds, όπου ο φόρτος εργασίας των Builds ανατίθεται σε πολλαπλούς κόμβους "σκλάβους", ή αλλιώς Slave Nodes. Έτσι μπορεί να δημιουργηθεί μια ενιαία εγκατάσταση Jenkins για την φιλοξένηση μεγάλου αριθμού έργων ή διαφορετικών περιβαλλόντων που απαιτούνται για τα Builds και τις Δοκιμές [43]. Αυτό σημαίνει ότι κατά την έναρξη της κάθε κατασκευής (Build) ο Jenkins αναζητά έναν ελεύθερο κόμβο-σκλάβο για την εκτέλεση. Αυτό είναι ιδιαίτερα χρήσιμο για την επιτάχυνση της φάσης των δοκιμών, διότι διαφορετικές δοκιμές μπορούν να εκτελεστούν παράλληλα σε διαφορετικούς κόμβους-σκλάβους. Συνεπώς, ταχύτερα αποτελέσματα των δοκιμών σημαίνουν, ταχύτερος κύκλος ανατροφοδότησης και ταχύτατη αναφορά πιθανών σφαλμάτων.

2.2.4.2.2 Bamboo

Το Bamboo ανήκει στην κατηγορία των Φιλοξενούμενων Διακομιστών Συνεχούς Ενσωμάτωσης, ή αλλιώς Hosted CI Servers. Δηλαδή, είναι προ εγκατεστημένος σε έναν διακομιστή (server) απομακρυσμένο εκτός της ομάδας εργασίας.

Το Bamboo είναι ένα εμπορικό προϊόν που αναπτύχθηκε από την εταιρία Atlassian. Στο συγκεκριμένο προϊόν η εστίαση δεν είναι τόσο πολύ σε μια συγκεκριμένη εργασία (Job), αλλά σε μια πολύ αφηρημένη έννοια ενός σχεδίου. Ένα σχέδιο μπορεί να περιέχει μία ή περισσότερες διαδοχικές φάσεις, οι οποίες με τη σειρά τους μπορούν να περιέχουν οποιοδήποτε αριθμό διαδοχικών εργασιών. Κάθε εργασία μπορεί να περιέχει οποιοδήποτε αριθμό των διαδοχικών εργασιών. Επιπλέον, το Bamboo υποστηρίζει κατανεμημένες κατασκευές (Builds).

Κάθε σχέδιο προορίζεται για να εργαστεί σε ένα συγκεκριμένο αποθετήριο κώδικα (με επιλογή για να ρύθμιση πολλαπλών αποθετηρίων)[44]. Το Bamboo είναι επίσης επεκτάσιμο μέσω plugins όπως και στον Jenkins, αλλά υπάρχουν σημαντικά λιγότερα add-ons.

2.3 Στόχοι της παρούσας εργασίας

Στη συνέχεια θα παρουσιαστούν οι στόχοι της παρούσας εργασίας.

2.3.1 Πρόβλημα του ελέγχου της ποιότητας του προϊόντος

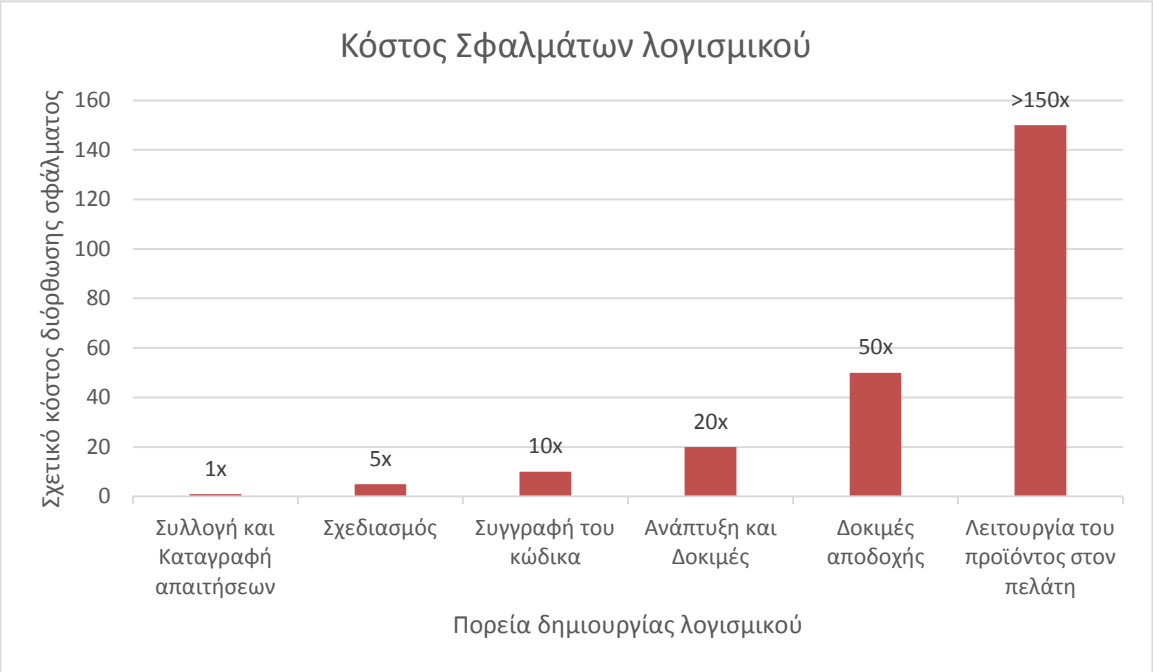
Ένα προϊόν λογισμικού, συχνά, αποτελείται από πολλαπλό αριθμό τμημάτων λογισμικού. Η συνένωση των τμημάτων αυτών αποτελεί ίσως το πολυπλοκότερο στάδιο στην παραγωγή λογισμικού. Πολλές φορές παρόλο που η συνένωση των τμημάτων λογισμικού είναι επιτυχής, το ίδιο το προϊόν παρουσιάζει σφάλματα. Συμπληρωματικά, τα σφάλματα αυτά μπορούν να παρουσιαστούν είτε σε επόμενο στάδιο ελέγχου ποιότητας του λογισμικού, είτε

στον ίδιο τον πελάτη. Συνεπώς, δεν είναι θεμιτό σε καμία περίπτωση να αντιληφθεί ο πελάτης σφάλμα στο λογισμικό ή να δυσχεράνει τη λειτουργικότητα και την απόδοσή του περιβάλλοντος του πελάτη. Το προϊόν λογισμικού πρέπει να χαρακτηρίζεται από τις παρακάτω αρχές: αξιοπιστία, λειτουργικότητα και ευχρηστία. Ένα σφάλμα λογισμικού, ή αλλιώς Software Bug, μπορεί να διαταράξει τις παραπάνω αρχές καθώς και την εμπιστοσύνη του πελάτη στον παραγωγό του λογισμικού(εταιρία).

Ένα σφάλμα λογισμικού, ανάλογα το στάδιο στο οποίο θα βρεθεί, θα έχει και το αντίστοιχο κόστος. Πιο συγκεκριμένα, τα στάδια που απαιτούνται για την ανάπτυξη και την παραγωγή ενός προϊόντος λογισμικού είναι τα ακόλουθα:

1. Συλλογή και Καταγραφή απαιτήσεων
2. Σχεδιασμός
3. Συγγραφή του κώδικα
4. Ανάπτυξη και Δοκιμές
5. Δοκιμές αποδοχής
6. Λειτουργία του προϊόντος στον πελάτη

Παρακάτω ακολουθεί σχηματική αναπαράσταση του ποσοστού του κόστους σφάλματος σε συνάρτηση με το στάδιο που βρίσκεται το σφάλμα.



Εικόνα 23: Διάγραμμα κόστους σφαλμάτων λογισμικού. [45]

Παρατηρείται ότι το κόστος στα τελευταία στάδια εκτοξεύεται στα ύψη περίπου 150 φορές περισσότερο κόστος απ' ότι στα αρχικά στάδια. Συνεπώς, η ανάγκη για μείωση του κόστους και η αποφυγή εμφάνισης σφάλματος στον πελάτη είναι υψίστης σημασίας.

2.3.2 Στόχοι και συνεισφορά της παρούσας εργασίας

Εξετάζοντας τα παραπάνω προβλήματα, προκύπτει ο προβληματισμός και κατ' επέκταση και στόχος της παρούσας εργασίας. Ειδικότερα, με ποιο τρόπο είναι δυνατό να ανιχνευθεί σε πιο πρώιμο στάδιο η πιθανή ύπαρξη σφαλμάτων του λογισμικού, ώστε να μειωθεί:

1. το κόστος και
2. το ποσοστό σφαλμάτων του λογισμικού που παράγεται.

Προκειμένου να επιτευχθεί αυτό το αποτέλεσμα της μείωσης, είναι απαραίτητη η αύξηση της ποιότητας του παραγόμενου λογισμικού.

Για να υλοποιηθούν οι παραπάνω στόχοι, αναπτύχθηκε ένα σύστημα ελέγχου λογισμικού. Δηλαδή, ένα Σύστημα Συνεχούς Ενσωμάτωσης, ή αλλιώς «CI Σύστημα», το οποίο τοποθετήθηκε ένα στάδιο πριν το σημείο ενσωμάτωσής στο κεντρικό αποθετήριο.

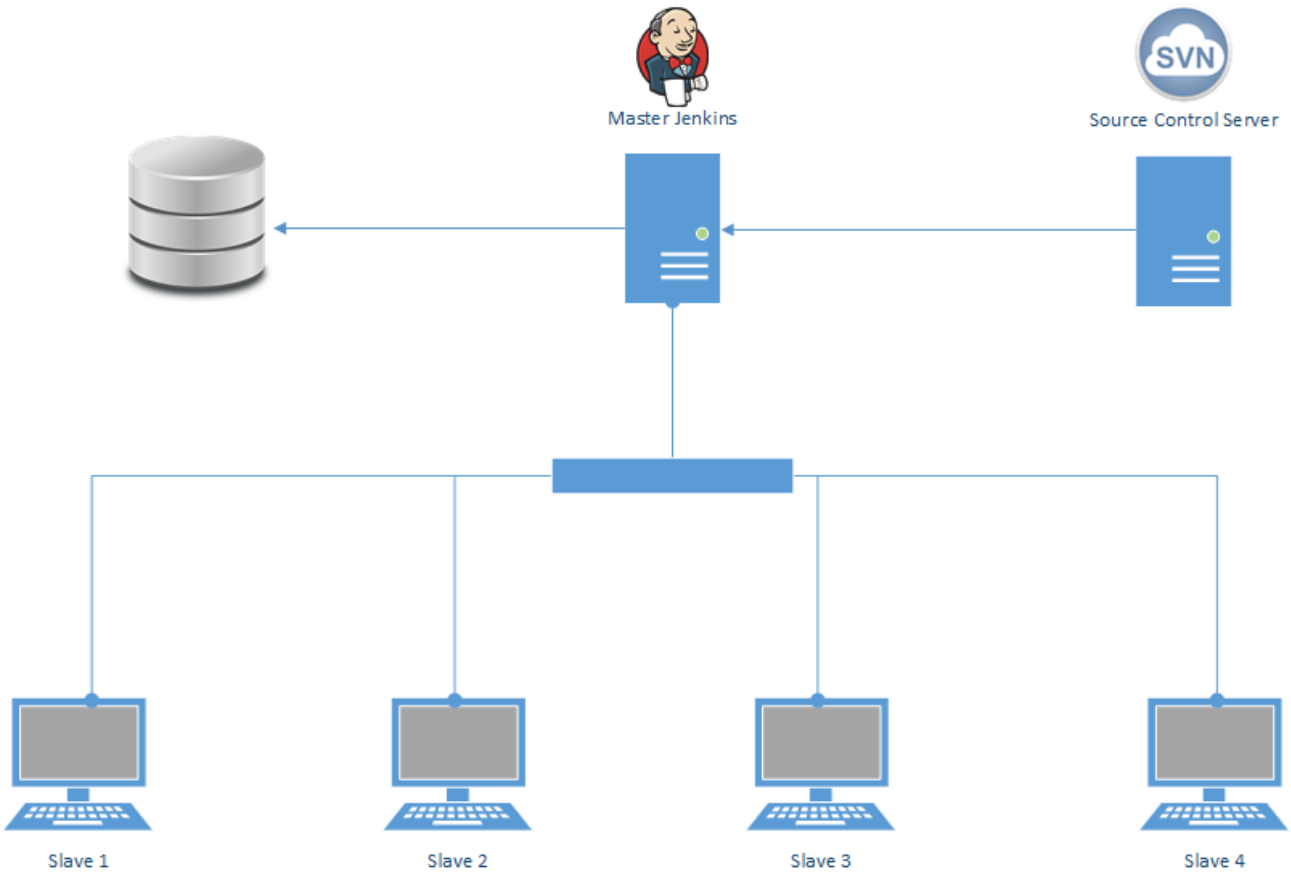
3. ΑΝΑΠΤΥΞΗ CI ΣΥΣΤΗΜΑΤΟΣ

Σε αυτό το κεφάλαιο, θα παρουσιαστεί η ανάπτυξη του CI συστήματος. Αρχικά, θα περιγραφεί η διαδικασία ελέγχου κατά την εισαγωγή μιας νέας εργασίας(στο ήδη υπάρχον σύστημα). Στη συνέχεια θα αναλυθεί η δομή και η λειτουργία του CI συστήματος που υλοποιήθηκε. Τέλος, θα παρουσιάσουμε την λειτουργία του μέσα στις ομάδες ανάπτυξης λογισμικού καθώς και την καλύτερη αξιοποίησή του από αυτές.

3.1 Δομή συστήματος

Σε αυτήν την υποενότητα θα παρουσιαστεί η Δομή των εργασιών του CI Συστήματος. Αρχικά, θα παρουσιαστεί η αρχιτεκτονική ελέγχου του προϊόντος λογισμικού που μελετάται και εν συνεχεία, θα γίνει αναφορά στην ροή των εργασιών.

Για να γίνει περισσότερο κατανοητό το CI θα γίνει μια μικρή παρουσίαση της διάταξής του από άποψη υλικού. Στο παρακάτω σχήμα, υπάρχει ένας κεντρικός διακομιστής Jenkins (Master Jenkins) ο οποίος αναλαμβάνει το ρόλο του οργανωτή του CI. Ο Jenkins επικοινωνεί με μία βάση δεδομένων, τον διακομιστή του κεντρικού αποθετηρίου (Source Control Server) και με τους κόμβους (Slaves) οι οποίοι πραγματοποιούν τα σενάρια ελέγχου.



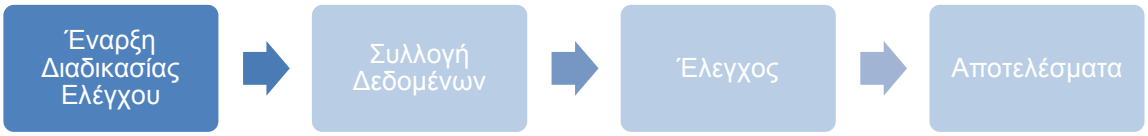
Εικόνα 24: Διάταξη του CI συστήματος από την άποψη υλικού.

3.2 Ανάλυση - Υλοποίηση συστήματος

Σε αυτήν την υποενότητα θα παρουσιαστεί η ανάλυση και λειτουργικότητα των επιμέρους στοιχείων που αποτελούν την σύσταση του CI Συστήματος.

3.2.1 Έναρξη διαδικασίας ελέγχου

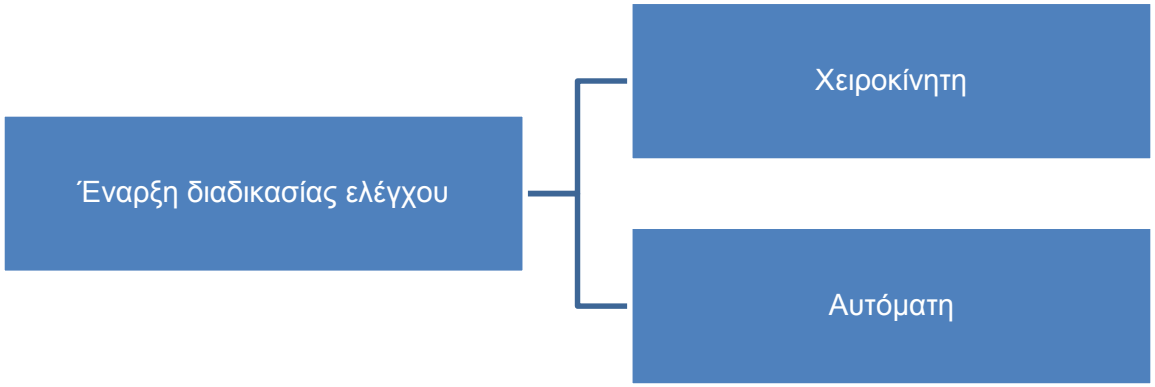
Η έναρξη της διαδικασίας ελέγχου είναι το πρώτο βήμα στην ροή του CI συστήματος. Ο ρόλος αυτού του βήματος είναι να εκκινεί το CI σύστημα. Η συγκεκριμένη διαδικασία μπορεί να ξεκινήσει είτε αυτόματα, είτε χειροκίνητα. Παρακάτω αναλύονται αυτές οι περιπτώσεις εκκίνησης του CI συστήματος αναλυτικότερα.



Εικόνα 25: Η έναρξη διαδικασίας ελέγχου στη ροή του CI συστήματος.

3.2.1.1 Χειροκίνητη

Ορισμένες φορές οι προγραμματιστές θέλουν να πραγματοποιήσουν προκαταρκτικό έλεγχο στο τμήμα κώδικα που καλούνται να υποβάλουν(commit στο repository). Η χειροκίνητη λειτουργία τους δίνει την δυνατότητα να ελέγξουν το παραδοτέο τους πριν την υποβολή του. Έτσι, κατά την αυτόματη εκκίνηση του CI (θα αναλυθεί στην συνέχεια), ο κώδικας που εισήλθε στο repository είναι χωρίς πιθανά σφάλματα, δηλαδή «καθαρός».



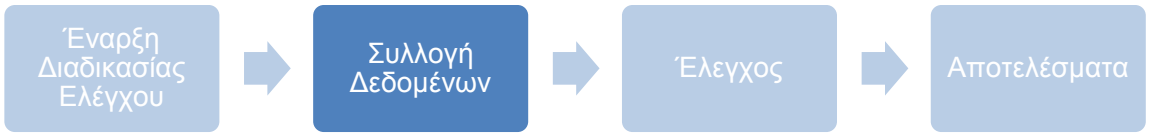
Εικόνα 26: Περιπτώσεις έναρξης της διαδικασίας ελέγχου.

3.2.1.2 Αυτόματη

Κάθε φορά που υποβάλλεται ένα παραδοτέο κώδικα στο repository, το CI εκκινείται αυτόματα. Πιο συγκεκριμένα, το CI εκκινείται αυτόματα όταν αντιληφθεί αλλαγές στο repository. Οι αλλαγές στο repository γίνονται αισθητές στο σύστημα μέσα από μηχανισμούς που θα αναλύσουμε στην συνέχεια, οι οποίοι ονομάζονται «SVN Listeners».

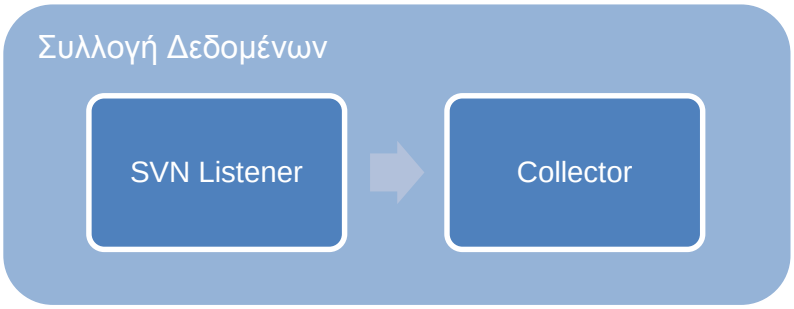
3.2.2 Συλλογή Δεδομένων

Η συλλογή δεδομένων είναι το επόμενο βήμα στην ιεραρχία της διαδικασίας ελέγχου. Στο συγκεκριμένο βήμα το CI σύστημα ενημερώνεται για πιθανές αλλαγές στο repository και στην συνέχεια συλλέγει τα κατάλληλα δεδομένα για την διαδικασία ελέγχου.



Εικόνα 27: Η συλλογή δεδομένων στη ροή του CI συστήματος.

Η διαδικασία αυτή αποτελείται από δύο επιμέρους υπό διαδικασίες: την ανίχνευση των εκδόσεων των δεδομένων που απαιτούνται για τα επόμενα βήματα, καθώς και την συλλογή και μεταφορά αυτών των δεδομένων στους κόμβους εκτέλεσης ελέγχων.

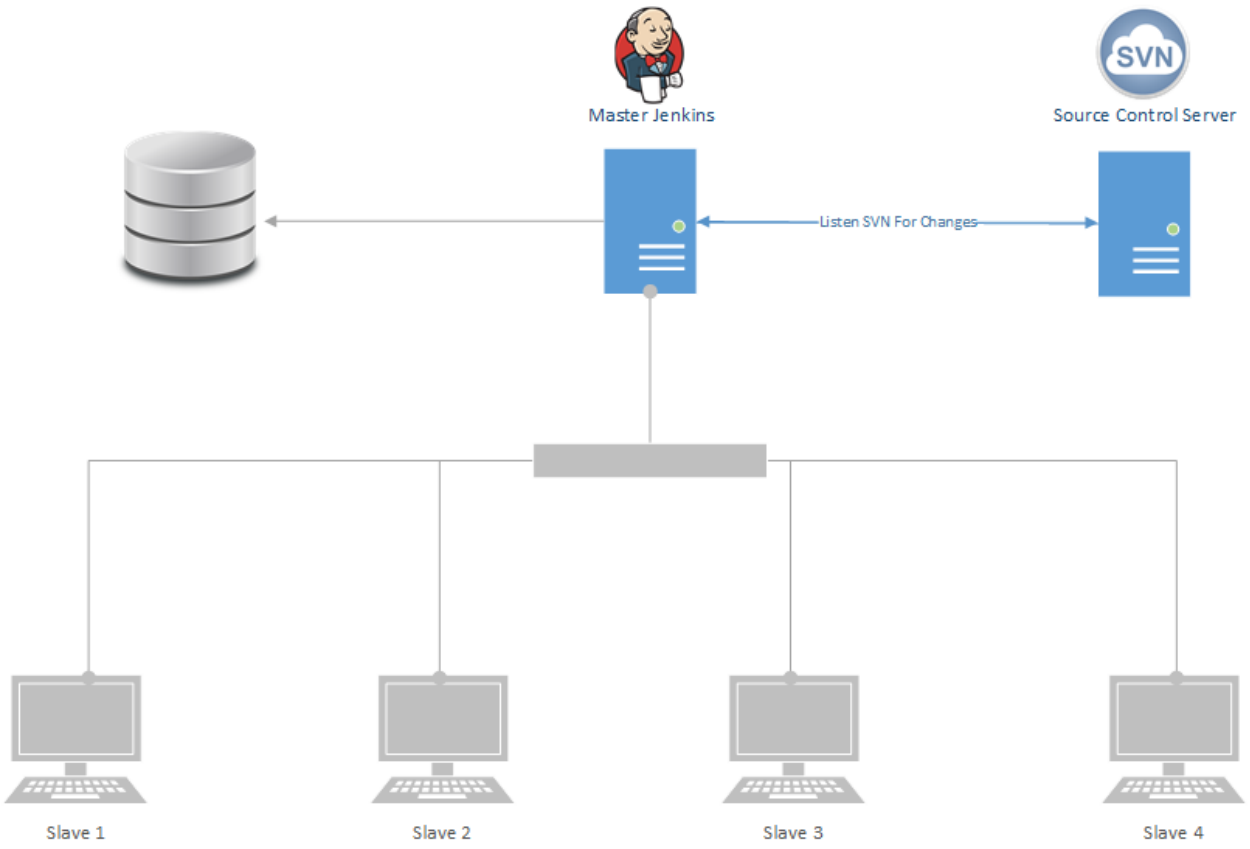


Εικόνα 28: Η συλλογή δεδομένων στη ροή του CI συστήματος.

3.2.3 SVN Listener

Οι SVN Listeners έχουν τον ρόλο του επόπτη στο repository. Πιο συγκεκριμένα, ενημερώνουν το CI μόλις αντιληφθούν κάποια εισαγωγή ή αλλαγή σε κώδικα του repository. Για να έχουν άμεση αντίδραση, οι Listeners «ακούν» (poll SCM) το repository κάθε 2 λεπτά. Συνεπώς, αν μία υποβολή κώδικα γίνει στη χρονική στιγμή t (λεπτά), η εκκίνηση του CI θα

πραγματοποιηθεί, το πολύ, στη χρονική στιγμή t+2 (λεπτά). Οι SVN Listeners μπορούν να υπάρχουν σε μεγάλο πλήθος. Το πλήθος τους εξαρτάται από πώς είναι δομημένο το repository. Τα παραπάνω αφορούν την αυτόματη εκτέλεση και τα παρακάτω εφαρμόζονται σε κάθε περίπτωση. Οι Listeners ενημερώνουν το CI για τις τρέχουσες εκδόσεις των τμημάτων κώδικα που απαιτούνται για την δημιουργία ενός SW Build. Η παραπάνω πληροφορία μεταδίδεται στο επόμενο μέρος της Συλλογής Δεδομένων που ονομάζεται συλλογέας, ή αλλιώς Collector. Στο παρακάτω σχήμα αναφέρεται η θέση και ο ρόλος των SVN Listeners στο CI.



Εικόνα 29: Η θέση και ο ρόλος των SVN Listeners στο CI σύστημα.

3.2.4 Collector

Με την μεταβίβαση των τρέχουσων εκδόσεων κώδικα (svn header), από τους SVN Listeners όπως είδαμε παραπάνω, φτάνουμε στο στάδιο του Collector. Ο ρόλος του Collector είναι να συλλέξει όλα τα απαραίτητα αρχεία που απαιτούνται για την δημιουργία του SW Build. Τα αρχεία που συλλέγονται είναι στην έκδοση που έχει υποδειχθεί από τους Listeners. Τα αρχεία που συλλέγονται μπορούν να είναι:

- Αρχεία Κώδικα (Code)
- Σύνολο από σενάρια ελέγχου
- Πληροφορίες σχετικά με το περιβάλλον του ελέγχου(testing)

- Αρχεία δημιουργίας εκτελέσιμου αρχείου

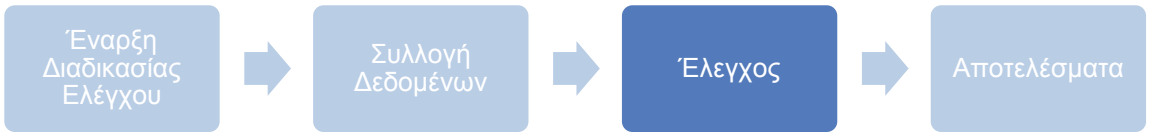
Μετά το πέρας της συλλογής των απαιτούμενων αρχείων, ο Collector μεταφέρει το σύνολο αυτών των αρχείων στον κόμβο που θα πραγματοποιηθούν οι έλεγχοι λογισμικού. Η επιλογή του κόμβου για την πραγματοποίηση των ελέγχων λογισμικού γίνεται με δυναμικό τρόπο λαμβάνοντας υπόψιν παράγοντες όπως:

- Το πλήθος των ελεύθερων κόμβων την δεδομένη χρονική στιγμή
- Το είδος σεναρίων που θα ακολουθηθούν για την πραγματοποίηση των ελέγχων
- Την προτεραιότητα που έχει μία εργασία(στην ουρά) για πραγματοποίηση ελέγχου

Την σκυτάλη παραλαμβάνει ο εκτελεστής του ελέγχου λογισμικού ή αλλιώς Executor.

3.2.5 Executor

Είναι το πιο κρίσιμο σημείο στην διαδικασία ελέγχου. Σε αυτό το σημείο της διαδικασίας ελέγχου γίνονται όλες οι απαραίτητες ενέργειες για να διεξαχθεί ο έλεγχος λογισμικού. Με την ολοκλήρωση του συγκεκριμένου σταδίου εξάγονται τα αποτελέσματα των ελέγχων λογισμικού.



Εικόνα 30: Ο έλεγχος στη ροή του CI συστήματος.

Ο Executor έχει πολλούς ρόλους στη συνολική διαδικασία ελέγχου λογισμικού. Συνεπώς, για την βαθύτερη κατανόησή του είναι αναγκαία η ανάλυσή του σε απλούστερες λειτουργίες. Οι επιμέρους λειτουργίες που αποτελούν τον Executor είναι οι παρακάτω με τη αντίστοιχη σειρά εκτέλεσής τους:

1. Καθαρισμός χώρου εργασίας
2. SVN Checkout
3. Προετοιμασία περιβάλλοντος
4. Μεταγλώττιση πηγαίου κώδικα
5. Προετοιμασία εξομοιωτή
6. Μεταφορά εκτελέσιμου αρχείου στον εξομοιωτή
7. Έλεγχος Περιβάλλοντος εκτέλεσης

8. Τεστ
9. Συλλογή αρχείων καταγραφής
10. Αποθήκευση αποτελεσμάτων

Στη συνέχεια θα αναλυθούν με λεπτομέρεια οι παραπάνω λειτουργίες του Executor.

1. Καθαρισμός χώρου εργασίας: Ο κόμβος σκλάβος, στον οποίο εκτελείται ο Executor, κατά πάσα πιθανότητα, έχει υπολείμματα από προηγούμενες εκτελέσεις. Συνεπώς, είναι αναγκαίος ο «καθαρισμός» του χώρου εργασίας(workspace) πριν από οποιαδήποτε άλλη ενέργεια. Η συγκεκριμένη λειτουργία αποτελεί ένα προληπτικό βήμα για την αποφυγή σφαλμάτων αρχείων.

2. SVN Checkout: Αρκετές φορές, τα αρχεία που προωθούνται από τον Collector δεν είναι αρκετά για να πραγματοποιηθεί το εκτελέσιμο αρχείο. Συνεπώς, απαιτούνται επιπλέον αρχεία τα οποία βρίσκονται στο SVN repository. Η συγκεκριμένη λειτουργία είναι υπεύθυνη να αντλήσει από το SVN repository όλα τα επιπλέον αρχεία που απαιτούνται.

3. Προετοιμασία περιβάλλοντος: Κάθε έλεγχος λογισμικού είναι ξεχωριστός, ανάλογα με το κομμάτι λογισμικού που ελέγχεται. Συνεπώς, κάθε κομμάτι λογισμικού για να ελεγχθεί πλήρως χρειάζεται ένα εξομοιωμένο περιβάλλον που αντικατοπτρίζει τις πραγματικές συνθήκες. Ως παράδειγμα θα μπορούσε να θεωρηθεί ένα σύνολο διεργασιών οι οποίες έχουν το ρόλο να φέρουν φορτίο στο ελεγχόμενο σύστημα όπως στις πραγματικές συνθήκες. Η λειτουργία αυτή πρέπει να προετοιμάσει το περιβάλλον ελέγχου σύμφωνα με τις ανάγκες του υποψήφιου πρός έλεγχο λογισμικού.

4. Μεταγλώττιση πηγαίου κώδικα: Για να πραγματοποιηθεί ο έλεγχος λογισμικού, πρέπει να μετατραπεί ο πηγαίος κώδικας (το κομμάτι λογισμικού που υποβάλλεται στο CI σύστημα προς έλεγχο) σε ένα εκτελέσιμο αρχείο. Η διαδικασία, μέσα από την οποία θα προκύψει το εκτελέσιμο αρχείο ονομάζεται μεταγλώττιση πηγαίου κώδικα. Κατά την μεταγλώττιση υπάρχει το ενδεχόμενο, η διαδικασία αυτή, να αποτύχει. Συνεπώς, η συγκεκριμένη λειτουργία αποτελεί μία από τις πιο κρίσιμες λειτουργίες του Executor. Σε αυτήν την περίπτωση, ο προγραμματιστής που πραγματοποίησε το Commit κώδικα, είναι υποχρεωμένος να το αναιρέσει για να επανέλθει το CI σύστημα σε μη λανθάνουσα κατάσταση. Επιπλέον, το CI σύστημα ενημερώνει, με ηλεκτρονικό μήνυμα, τον προγραμματιστή για το συμβάν και του προωθεί όλα τα αρχεία που προέκυψαν κατά την λειτουργία αυτή (αρχεία καταγραφής - log files). Σε διαφορετική περίπτωση, δημιουργείται το εκτελέσιμο αρχείο.

5. Προετοιμασία εξομοιωτή: Ένας εξομοιωτής μπορεί να είναι μία πραγματική Atca ή ένα εικονικό περιβάλλον που προσομοιώνει μια Atca. Κατά την διάρκεια της συγκεκριμένης λειτουργίας, ο εξομοιωτής, στον οποίο επρόκειτο να πραγματοποιηθεί ο έλεγχος λογισμικού, τροποποιείται κατάλληλα για της ανάγκες του ελέγχου. Δηλαδή, διαμορφώνεται το περιβάλλον μέσα στο οποίο θα «τρέξει» το εκτελέσιμο αρχείο. Η τροποποίηση αφορά καθαρά το λογισμικό του προσομοιωτή και όχι το υλικό του. Πιο συγκεκριμένα, το εκτελέσιμο αρχείο έχει κάποια χαρακτηριστικά διασύνδεσης με το λογισμικό του εξομοιωτή, δηλαδή ορισμένους κανόνες που εγγυώνται την ελεγχόμενη λειτουργία του. Έτσι γίνεται μία αρχικοποίηση στον προσομοιωτή σύμφωνα με τους κανόνες του εκάστοτε εκτελέσιμου αρχείου. Κάθε τμήμα λογισμικού είναι μοναδικό,

συνεπώς, και το περιβάλλον εκτέλεσής του. Επομένως, η συγκεκριμένη λειτουργία είναι πολύ κρίσιμη για την έκβαση του ελέγχου.

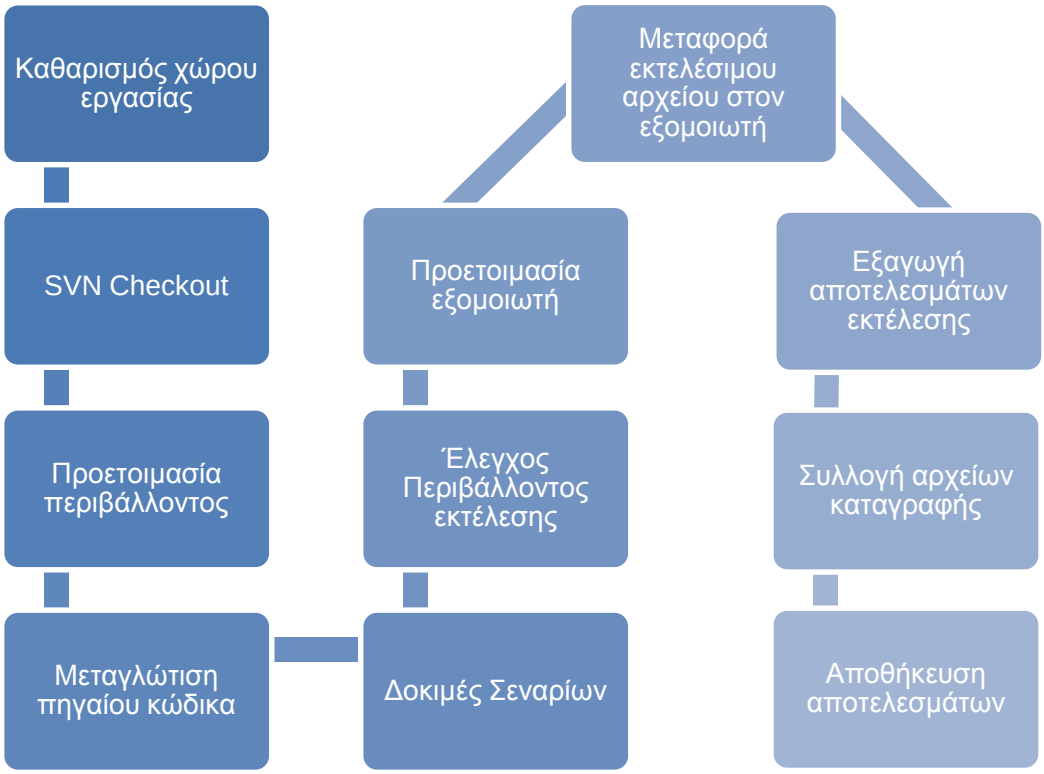
6. Μεταφορά εκτελέσιμου αρχείου στον εξομοιωτή: Μετά την πραγματοποίηση των παραπάνω δύο βημάτων, πρέπει να γίνει η μεταφορά του εκτελέσιμου αρχείου στον εξομοιωτή. Για να γίνει η παραπάνω μεταφορά πρέπει να είναι γνωστή η διεύθυνση του εξομοιωτή.

7. Έλεγχος Περιβάλλοντος εκτέλεσης: Αποτελεί έναν πολύ σύντομο έλεγχο που πραγματοποιείται ανάμεσα στον εξομοιωτή και το εκτελέσιμο αρχείο. Κατά αυτή τη διαδικασία ελέγχεται η συνδεσιμότητα του εξομοιωτή με το εκτελέσιμο αρχείο.

8. Τεστ: Στη συνέχεια, πραγματοποιούνται τα σενάρια ελέγχου σειριακά. Κατά την συγκεκριμένη λειτουργία δημιουργούνται αρχεία καταγραφής. Τα παραπάνω αρχεία αποθηκεύουν στοιχεία και δεδομένα που αφορούν την αλληλεπίδραση του εκτελέσιμου αρχείου με το περιβάλλον του.

9. Συλλογή αρχείων καταγραφής: Μετά το πέρας της εκτέλεσης, πραγματοποιείται συλλογή και μερική επεξεργασία των αρχείων καταγραφής. Πιο συγκεκριμένα, ο Executor συλλέγει όλα τα αρχεία καταγραφής με πρώτο στόχο την συγκομιδή αυτών των αρχείων. Στη συνέχεια, τα αρχεία αυτά θα υποστούν μερική επεξεργασία, η οποία περιορίζεται στην αφαίρεση διπλοτύπων, στην εισαγωγή πληροφοριών σχετικά με το Build και στην καταμέτρηση του πλήθους των λαθών.

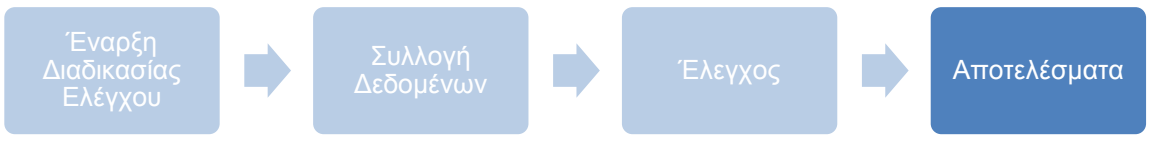
10. Αποθήκευση αποτελεσμάτων: Τέλος, ο Executor αποθηκεύει όλα τα δεδομένα που συλλέχθηκαν από το παραπάνω βήμα σε ένα φυσικό μέσο αποθήκευσης, όπως ένας σκληρός δίσκος. Η διεύθυνση του παραπάνω μέσου αποθήκευσης είναι γνωστή για το CI σύστημα.



Εικόνα 31: Οι λειτουργίες του Executor.

3.2.6 Αποτελέσματα ελέγχου

Τα αποτελέσματα ελέγχου είναι το επόμενο βήμα στην ιεραρχία της διαδικασίας ελέγχου. Στο συγκεκριμένο βήμα το CI σύστημα ενημερώνει τους άμεσα εμπλεκόμενους ανθρώπους με το συγκεκριμένο Build. Στη συνέχεια εισάγει όλα τα δεδομένα που έχουν εξορυχθεί από την διαδικασία ελέγχου σε βάση δεδομένων.



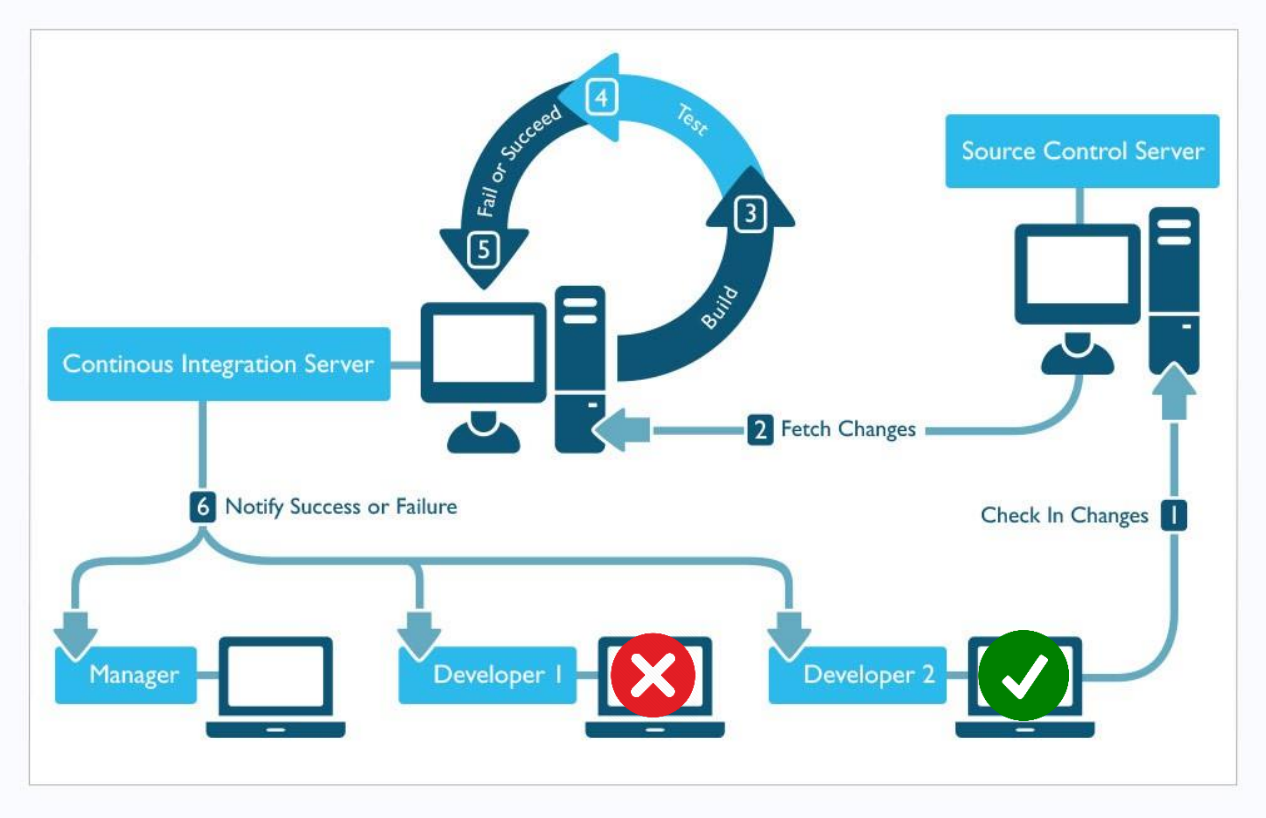
Εικόνα 32: Τα αποτελέσματα στη ροή του CI συστήματος.

3.2.6.1 Εμφάνιση αποτελεσμάτων

Μετά το πέρας της διεξαγωγής των σεναρίων ελέγχου, όπως αναφέρθηκαν παραπάνω, εξάγονται ορισμένα δεδομένα. Τα δεδομένα αυτά έχουν ιδιαίτερα αυξημένη σημασία διότι, με βάση αυτά θα αξιολογηθεί το αποτέλεσμα από την ανάπτυξη του συγκεκριμένου τμήματος λογισμικού. Πιο συγκεκριμένα, ο προγραμματιστής, ο μάνατζερ του και η ομάδα του ενημερώνονται για τα αποτελέσματα της εκτέλεσης. Ανάλογα με τα αποτελέσματα γίνονται και οι κατάλληλες ενέργειες.

Αν υπάρχουν λάθη στο τμήμα λογισμικού που υποβλήθηκε, γίνεται «πάγωμα» στις υποβολές λογισμικού και ο προγραμματιστής πρέπει να είναι σε θέση να διορθώσει το σφάλμα. Διαφορετικά, η υποβολή του κώδικα αναιρείται προκειμένου να είναι καθαρό το αποθετήριο προς πιθανή άλλη υποβολή τμήματος λογισμικού.

Σε διαφορετική περίπτωση, δηλαδή αν δεν υπάρχουν σφάλματα στο τμήμα λογισμικού, οι αλλαγές γίνονται μόνιμες στο αποθετήριο.



Εικόνα 33: Παράδειγμα αποτελεσμάτων επιτυχίας και αποτυχίας.

3.2.6.2 Ενημέρωση βάσης δεδομένων

Κλείνοντας την ιεραρχία της διαδικασίας ελέγχου αναλύεται το τελευταίο μέρος του CI συστήματος, η ενημέρωση της βάσης δεδομένων. Η ανάγκη για διατήρηση ιστορικού των αποτελεσμάτων δημιούργησε την ιδέα της δημιουργίας μίας βάσης δεδομένων. Η βάση αυτή περιέχει όλα τα δεδομένα που αφορούν παλαιότερες εκτελέσεις τμημάτων λογισμικού. Με

αυτόν τον τρόπο μπορεί να υπάρχει μία συγκεντρωτική εικόνα για την πορεία εξέλιξης ενός προϊόντος.

Κατά την διαδικασία αυτή, τα δεδομένα που προέρχονται από την τρέχουσα εκτέλεση αποθηκεύονται στην βάση δεδομένων. Οι πληροφορίες που αποθηκεύονται είναι ύψιστης σημασίας, διότι στο μέλλον μπορεί να αναπτυχθεί ένας μηχανισμός ο οποίος θα προβλέπει λάθη κατά την παραγωγή και την σχεδίαση λογισμικού.

4. ΑΠΟΤΕΛΕΣΜΑΤΑ - ΣΥΜΠΕΡΑΣΜΑΤΑ

Σε αυτό το κεφάλαιο θα γίνει η παρουσίαση των αποτελεσμάτων από την ανάπτυξη και υλοποίηση του CI συστήματος. Συγκεκριμένα τα αποτελέσματα παρατίθενται οπτικοποιημένα μέσω γραφημάτων. Έπειτα θα σχολιαστούν τα αποτελέσματα από την οπτική γωνία του Προγραμματιστή, του Line Manager καθώς και του Project Manager. Τέλος θα μελετηθεί το πώς συνδέονται τα αποτελέσματα από τις οπτικές γωνίες των διαφορετικών ρόλων.

4.1 Παρουσίαση Αποτελεσμάτων

Μετά την ανάπτυξη και την εφαρμογή του συστήματος CI στη διαδικασία παραγωγής λογισμικού του SGSN εξ ορύχθηκαν ορισμένα σημαντικά αποτελέσματα.

Προτού παρουσιαστούν τα αποτελέσματα, θα γίνει εκτενής αναφορά στην συλλογή των δεδομένων. Πρέπει να αναφερθεί ότι η ανάπτυξη και η εφαρμογή του CI συστήματος, είχε συνολική διάρκεια δώδεκα μηνών από την έναρξή του. Επιπλέον, τα αποτελέσματα βασίζονται σε δεδομένα που συλλέχθηκαν από το CI σύστημα που υλοποιήθηκε στο τρίτο κεφάλαιο με αντικείμενο ελέγχου το προϊόν Flexi NS της NOKIA. Πιο αναλυτικά, τα παρακάτω γραφήματα που ακολουθούν παρουσιάζουν με οπτικό τρόπο τα αποτελέσματα.

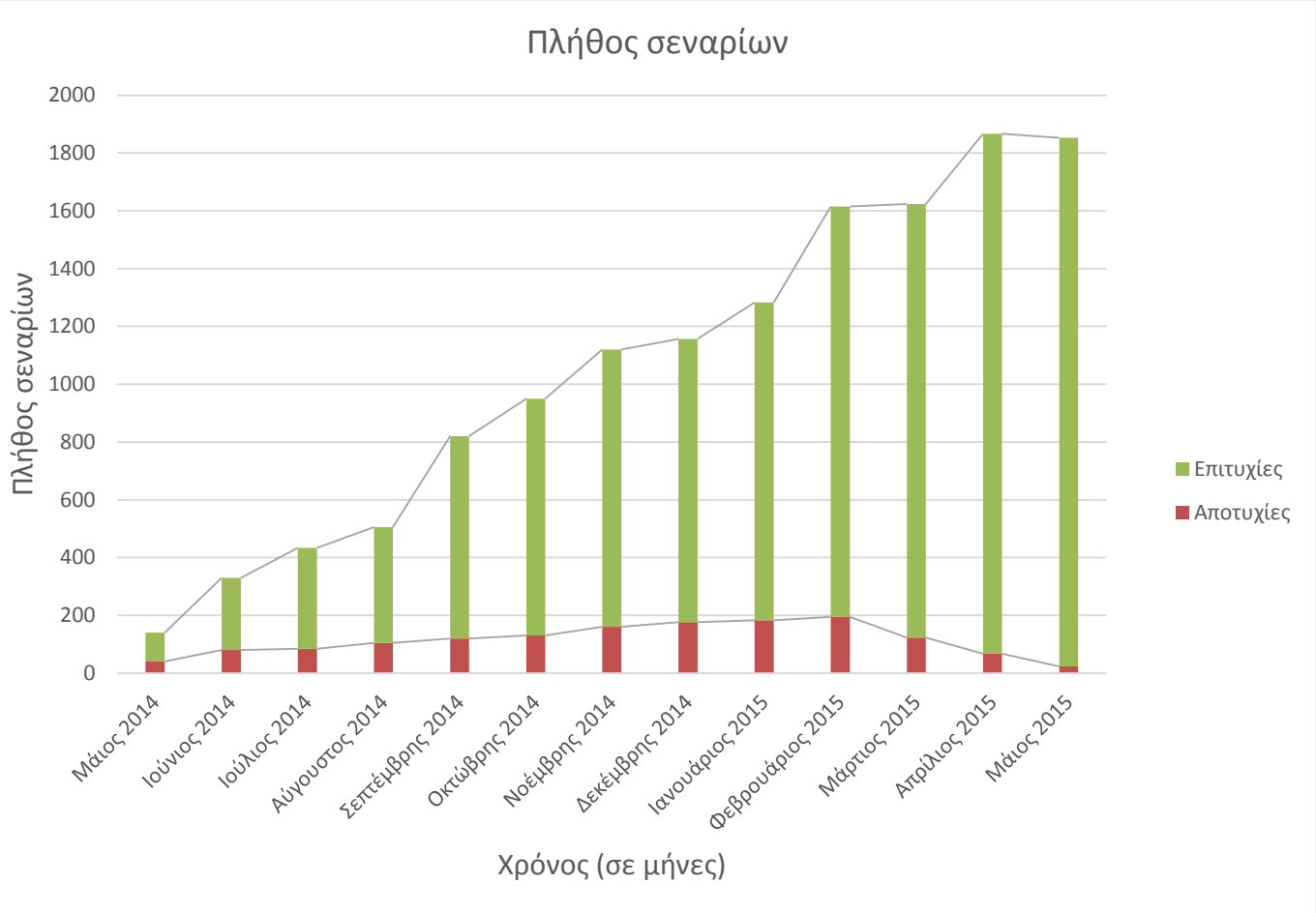
Το παρακάτω γράφημα παρουσιάζει την πορεία εξέλιξης των εκτελεσμένων, με επιτυχία, σεναρίων σε χρονικό διάστημα ενός έτους.

Εικόνα 34: Διάγραμμα πορείας των επιτυχών σεναρίων.

Ο οριζόντιος άξονας αντικατοπτρίζει τον χρόνο που διήρκησε η εξόρυξη δεδομένων. Πιο συγκεκριμένα, κάθε διάστημα χρόνου στο παραπάνω διάγραμμα, που παρουσιάζεται με μία κουκίδα, έχει διάρκεια έναν μήνα. Επιπρόσθετα, ο κάθετος άξονας αναπαριστά το πλήθος σεναρίων (Test Cases) για το οποίο το λογισμικό έχει εκτελεστεί επιτυχώς.

Στο παραπάνω διάγραμμα, παρατηρείται μη σταθερή συμπεριφορά του πλήθους των επιτυχών σεναρίων σε χρονικό διάστημα μερικών μηνών. Πιο συγκεκριμένα, το πλήθος των επιτυχών σεναρίων αυξάνεται γνησίως με μη γραμμικό τρόπο για το συνολικό διάστημα 12 μηνών. Επομένως, πρακτικά το προϊόν δοκιμάζεται σε μεγαλύτερο πλήθος σεναρίων και σημειώνει επιτυχία στο πλήθος αυτών.

Στη συνέχεια, αναλύεται ένα δεύτερο γράφημα όπου παρουσιάζει το πλήθος των επιτυχιών και των αποτυχιών, σε δοκιμές, σεναρίων σε διάστημα ενός έτους.



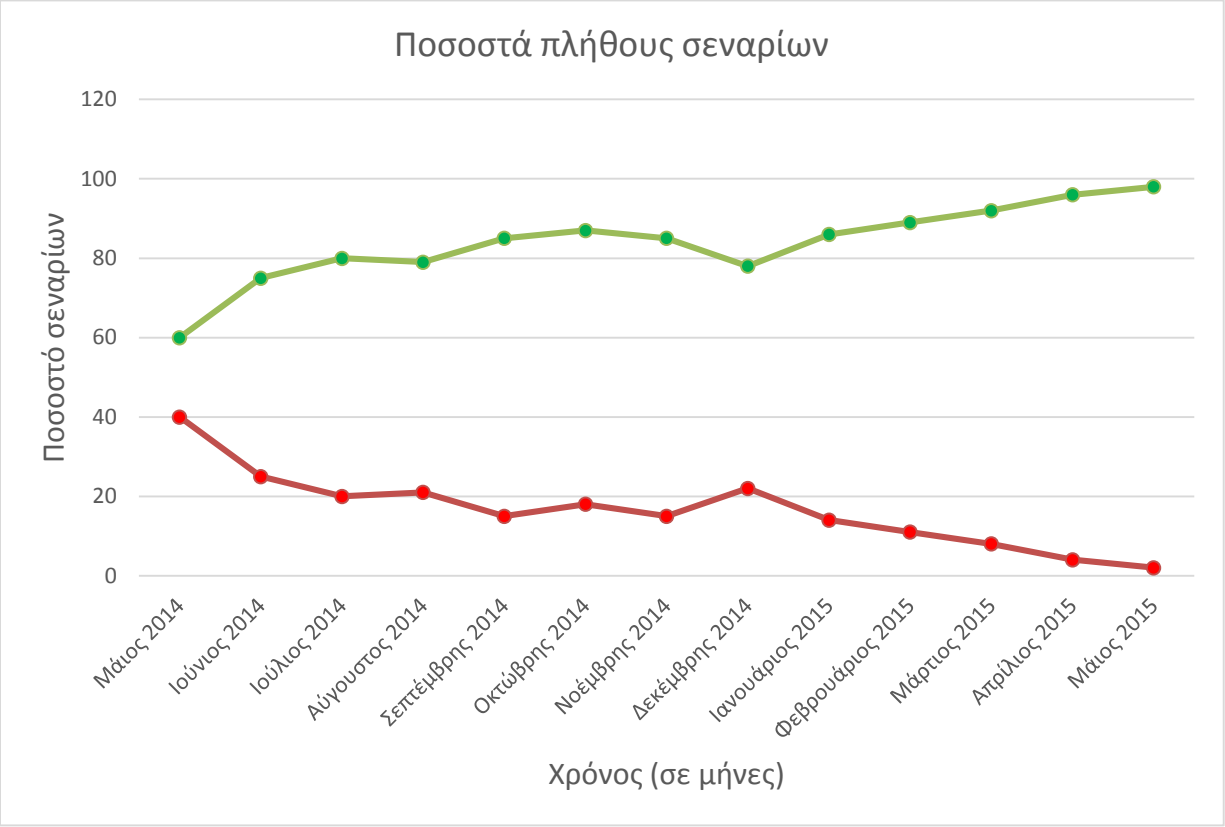
Εικόνα 35: Διάγραμμα συνολικών αποτελεσμάτων.

Ο οριζόντιος άξονας αντικατοπτρίζει τον χρόνο που διήρκησε η εξόρυξη δεδομένων. Πιο συγκεκριμένα, κάθε διάστημα χρόνου στο παραπάνω διάγραμμα, που παρουσιάζεται με μία κάθετη στοίβα, έχει διάρκεια έναν μήνα. Επιπρόσθετα, ο κάθετος άξονας αντικατοπτρίζει το πλήθος σεναρίων (Test Cases) για το οποίο το λογισμικό έχει υποβληθεί σε έλεγχο, περιλαμβάνοντας τις επιτυχίες και τις αποτυχίες.

Στο παραπάνω διάγραμμα, παρατηρείται μη σταθερή συμπεριφορά του πλήθους των σεναρίων σε χρονικό διάστημα μερικών μηνών. Πιο συγκεκριμένα, το πλήθος των σεναρίων αυξάνεται γνησίως με μη γραμμικό τρόπο για το συνολικό διάστημα 12 μηνών. Επομένως, πρακτικά το προϊόν δοκιμάζεται σε μεγαλύτερο πλήθος σεναρίων. Η αύξηση του πλήθους των σεναρίων οφείλεται σε δύο λόγους. Ο πρώτος λόγος είναι ότι στους πρώτους μήνες το

προϊόν βρίσκεται σε αρχικό στάδιο της ανάπτυξής του και κατά συνέπεια, ο έλεγχος που πραγματοποιούν τα σενάρια στο λογισμικό περιορίζεται σε βασικές λειτουργίες του. Ο δεύτερος λόγος είναι ότι από ένα σημείο και έπειτα, από τον Νοέμβρη και μετά κυρίως, το CI σύστημα έχει επιτύχει έναν από τους βασικούς στόχους του, να βοηθήσει τους προγραμματιστές να διορθώσουν τα κύρια σφάλματα στον κώδικα. Εξαιτίας αυτού, το ποσοστό επιτυχιών για τον μήνα Σεπτέμβρη αυξήθηκε σημαντικά, με αποτέλεσμα να προστεθούν νέα σενάρια στο σύστημα. Αυτή η νέα προσθήκη σεναρίων στο σύστημα, αυξάνει τον λεπτομερή έλεγχο του προϊόντος και κατά συνέπεια την ποιότητά του. Επιπλέον, παρατηρείται βελτίωση της συνολικής εικόνας του προϊόντος, αφού αρχικά το προϊόν δοκιμαζόταν σε πολύ μικρότερο πλήθος σεναρίων απ' ότι στο τέλος (Μάιος 2015).

Στη συνέχεια παρατίθεται ένα διάγραμμα που παρουσιάζει το ποσοστό επιτυχίας ανά μήνα για χρονικό διάστημα ενός έτους, όπως και παραπάνω στα υπόλοιπα διαγράμματα.



Εικόνα 36: Διάγραμμα ποσοστών επιτυχιών και αποτυχιών σεναρίων.

Ο οριζόντιος άξονας αντικατοπτρίζει τον χρόνο που διήρκησε η εξόρυξη δεδομένων. Πιο συγκεκριμένα, κάθε διάστημα χρόνου στο παραπάνω διάγραμμα, που παρουσιάζεται με μία κουκίδα, έχει διάρκεια έναν μήνα. Επιπρόσθετα, ο κάθετος άξονας αντικατοπτρίζει το συνολικό πλήθος σεναρίων (Test Cases) για το οποίο το λογισμικό έχει υποβληθεί σε έλεγχο, περιλαμβάνοντας τις επιτυχίες και τις αποτυχίες.

Στο παραπάνω διάγραμμα, παρατηρείται ότι ενώ το ποσοστό των επιτυχιών αυξάνεται γνησίως, το ποσοστό των αποτυχιών τείνει να εξαλειφθεί εντελώς. Αυτό είναι φυσιολογικό

αν ληφθεί υπόψιν ότι προστίθενται νέα σενάρια μόνο όταν αυξάνονται οι επιτυχίες και κατά συνέπεια μειώνονται τα σφάλματα.

Συνοπτικά, από τα παραπάνω δεδομένα , δημιουργείται το συμπέρασμα ότι με την εισαγωγή του CI συστήματος στη διαδικασία ανάπτυξης του προϊόντος, επιτυγχάνεται η βελτίωσή του. Πιο συγκεκριμένα:

- Αυξάνεται το πλήθος σεναρίων που δοκιμάζεται το προϊόν.
- Αυξάνεται το ποσοστό της επιτυχίας συνολικά για το προϊόν.

Κατά συνέπεια των παραπάνω, αφενός μεν αυξάνεται η ποιότητα του προϊόντος και αφετέρου δε μειώνεται ραγδαία το συνολικό κόστος αφαίρεσης λαθών σε επόμενα στάδια της παραγωγής του, αφού τα σφάλματα του λογισμικού τείνουν να εξαλειφθούν από τα πρώτα στάδια της ανάπτυξής του.

Συνοψίζοντας με βάση τα παραπάνω, μπορεί να εξ ορυχθεί ένα έμμεσο αποτέλεσμα. Πιο συγκεκριμένα, βελτιώνοντας την ποιότητα ενός προϊόντος όπως το Flexi NS, βελτιώνεται και το ίδιο το SGSN ως οντότητα. Από τα προηγούμενα κεφάλαια, έχει γίνει κατανοητός ο ζωτικής σημασίας ρόλος του SGSN για ολόκληρο τον πυρήνα του δικτύου. Συνεπώς, βελτιώνοντάς αυτό, επιτυγχάνεται συνολική βελτίωση της ποιότητας του δικτύου.

4.2 Τα συμπεράσματα από διάφορες οπτικές γωνίες

Στη συνέχεια, θα γίνει μια μικρή αναφορά στον τρόμο με τον οποίο αντιλαμβάνονται, ορισμένοι σημαντικοί παράγοντες για την ανάπτυξη του προϊόντος, την συμβολή του CI στην καθημερινότητά τους όπως έχει διαμορφωθεί μετά την εισαγωγή του.

4.2.1 Από την οπτική του Προγραμματιστή

Η οπτική «γωνία» του προγραμματιστή αποτελεί ίσως την πιο σημαντική από όλες τις οπτικές, διότι ο προγραμματιστής είναι ο άμεσα εμπλεκόμενος με την ανάπτυξη του λογισμικού. Είναι ο παράγοντας που θα αναπτύξει τον κώδικα και θα διορθώσει σφάλματα, που πιθανώς να έχει ξεχάσει στον κώδικά του. Επόμενος, το CI σύστημα για αυτόν είναι ο πιο στενός του «φίλος» που αφενός, θα τον αφυπνίσει για πιθανά σφάλματα που υπάρχουν, υπάρχουν, αφετέρου θα του προσδώσει σιγουριά και ασφάλεια για την δουλειά του.

4.2.2 Από την οπτική του Line Manager

Ο Line Manager έχει τον ρόλο του συντονιστή μιας ομάδας που αποτελείται από ένα σύνολο προγραμματιστών. Ο ρόλος του είναι κυρίως οργανωτικός και ορισμένες φορές αποτελεί μέρος της ομάδας των προγραμματιστών αφού ασχολείται και με την συγγραφή κώδικα. Το CI σύστημα μπορεί να του φανεί ιδιαίτερα χρήσιμο σε δύο περιπτώσεις. Στην πρώτη περίπτωση, το CI τον βοηθά στο να έχει μια σφαιρική άποψη για το υπό-έργο λογισμικού που έχει αναλάβει η ομάδα του. Στην δεύτερη περίπτωση, το CI του επιτρέπει να κάνει ορισμένες υποθέσεις και προβλέψεις για τον χρόνο περάτωσης του υπό-έργου που έχει αναλάβει η ομάδα του.

4.2.3 Από την οπτική του Project Manager

Ο Project Manager είναι υπεύθυνος για την περάτωση του έργου λογισμικού στο σύνολό του. Το CI σύστημα μπορεί να του προσφέρει σημαντικές πληροφορίες για την πορεία του έργου λογισμικού και επιπλέον μπορεί να του δώσει ποσοδείκτες που αφορούν την απόδοση των υφιστάμενων. Έτσι, το CI αντικατοπτρίζει την παρούσα εικόνα του προϊόντος.

Συνοψίζοντας από τα παραπάνω, το CI σύστημα μπορεί να χρησιμοποιηθεί με πολλούς τρόπους ανάλογα με το ρόλο του εργαζόμενου. Κατ' αυτόν τον τρόπο μπορεί να βοηθήσει στην καλύτερη οργάνωση ενός έργου λογισμικού και κατ' επέκταση, την ίδια την επιχείρηση που το χρησιμοποιεί. Συνεπώς, η αναγκαιότητα για ενσωμάτωση και χρήση τέτοιων συστημάτων μπορεί να αλλάξει την εικόνα των προϊόντων και την στάση των πελατών απέναντι τους, αλλά και κατ' επέκταση απέναντι στην εταιρία-παραγωγό γενικότερα.

5. ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

Σε αυτό το κεφάλαιο θα ασχοληθούμε με μελλοντική εργασία που προκύπτει από δημιουργία και τα συμπεράσματα της παρούσας εργασίας. Αρχικά θα παρουσιαστεί η επαναχρησιμοποίηση πόρων ως μελλοντική εργασία. Στη συνέχεια, ο επανασχεδιασμός του CI συστήματος με έμφαση στην απόδοση. Τέλος θα παρουσιαστεί πώς ο συγκερασμός του CI και των Big Data θα βελτιώσει το ίδιο το προϊόν.

5.1 Μετρικές Λογισμικού

Μετρικές λογισμικού (CodeMetrics) είναι ένα σύνολο μέτρων λογισμικού που παρέχει στους προγραμματιστές την καλύτερη κατανόηση του κώδικα που αναπτύσσουν. Εκμεταλλευόμενοι μετρήσεις κώδικα, οι προγραμματιστές μπορούν να προσδιορίσουν τους τύπους ή / και τις μεθόδους που πρέπει να επαναδιατυπωθούν ή να ελεγχθούν περισσότερο διεξοδικά. Οι ομάδες ανάπτυξης λογισμικού μπορούν να εντοπίσουν πιθανούς κινδύνους, να κατανοήσουν την τρέχουσα κατάσταση ενός έργου, καθώς και την παρακολούθηση της προόδου, κατά τη διάρκεια της ανάπτυξης του λογισμικού. [46] [48]

Μερικές μετρικές λογισμικού είναι:

- Διπλότυπος κώδικας: Ελέγχει αν υπάρχουν κοινά κομμάτια κώδικα.
- Πολυπλοκότητα κώδικα: Ελέγχει την πολυπλοκότητα του κώδικα με προσπέλαση από κάθε μέρος του κώδικα.
- Έλεγχος Ροής Δεδομένων: τεχνική πλέγματος, συλλέγει δεδομένα από τις πιθανές τιμές μεταβλητών.
- Γραμμές κώδικα: Ελέγχει πόσες γραμμές κώδικα προτέθηκαν.

Συνεπώς, με την εισαγωγή μετρικών λογισμικού στο CI σύστημα, θα μπορούν να αντληθούν πολλές χρήσιμες πληροφορίες για την βελτίωση του προϊόντος λογισμικού. [47] [49]

5.2 Big Data – Μηχανική Μάθηση

Η δοκιμή λογισμικού είναι η διαδικασία της εύρεσης σφαλμάτων στον κώδικα ενώ βρίσκεται σε λειτουργία. Τα αποτελέσματα των δοκιμών μπορούν να χρησιμοποιηθούν για να διορθώσουν τα σφάλματα. Η πρόβλεψη ελαττωματικού λογισμικού μπορεί να χρησιμοποιηθεί για την βελτίωση των δοκιμών και τελικά την βελτίωση της ποιότητάς του λογισμικού. Η μηχανική μάθηση, σε ότι αφορά τα προγράμματα ηλεκτρονικών υπολογιστών μάθησης από δεδομένα, μπορεί να χρησιμοποιηθεί για την κατασκευή μοντέλων πρόβλεψης, τα οποία χρησιμεύουν στην ταξινόμηση των δεδομένων. [50]

Συνεπώς, η εισαγωγή της μηχανικής μάθησης για την πρόβλεψη λαθών στον κώδικα μπορεί να βελτιώσει σε μεγάλο βαθμό το λογισμικό και επιπλέον να μειώσει σημαντικά το κόστος, αφού ο παραγόμενος κώδικας θα έχει λιγότερα λάθη.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
1G	Πρώτη γενιά
2G	Δεύτερη γενιά
2.5G	Ενδιάμεση δεύτερη γενιά
2G++	Ενδιάμεση δεύτερη γενιά
2.75G	Τρίτη γενιά
3G	Τρίτη γενιά
4G	Τέταρτη γενιά
Cloud	Τεχνολογία νέφους
Collector	Συλλογέας
Executor	Δοκιμαστής Τεστ
Listener	Ακροατής
Hardware	Υλικός εξοπλισμός
Microwave	Μικροκυματικό
Release	Έκδοση
Software	Λογισμικό
Software layer	Επίπεδο Λογισμικού
Internet	Διαδίκτυο
Multimedia	Πολυμέσα
Video	Βίντεο
Timeslot	Χρονοθυρίδα
Video streaming	Δρομολόγηση βίντεο
Internet browsing	Αναζήτηση στο διαδίκτυο
Uplink	Ανοδική ζεύξη
Downlink	Καθοδική ζεύξη
Gateway	Πύλη
Process	Διαδικασία
Development	Ανάπτυξη
Continuous integration	Συνεχής ενσωμάτωση
Releasing	Δημιουργία έκδοσης
Patches	Διορθωτικά τμήματα λογισμικού
SW Build	Παραγόμενο λογισμικό
Throughput	Διεκπεραίωση
Bug	Σφάλμα λογισμικού
Repository	Αποθετήριο
Code	Κώδικας λογισμικού
Testing	Δοκιμές
Slave	Σκλάβος
Master	Κύριος
Node	Κόμβος
Hosted	Φιλοξενούμενος

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

MSC	Mobile Switching Center
AMPS	Advanced Mobile Phone System
C-450	Radio Telephone Network C
NMT	Nordic Mobile Telephony
TACS	Total Access Communication System
ΗΠΑ	Ηνωμένες πολιτείες Αμερικής
FMDA	Frequency Division Multiple Access
GSM	Global System for Mobile Communication
ITU	International Telecommunication Union
TDMA	Time Division Multiple Access
PDC	Primary Domain Controller
SMS	Short Message Service
NADC	North American Digital Cellular
CDMA	Code Division Multiple Access
GPRS	General Packet Radio Service
HSCSD	High Speed Circuit Switched Data
ISDN	Integrated Services Digital Network
WAP	Wireless Application Protocol
WCDMA	Wideband Code Division Multiple Access
UMTS	Universal Mobile Telecommunications System
EDGE	Enhanced Data rates for GSM Evolution
ETSI	European Telecommunications Standards Institute
ITU-R	International Telecommunications Union-Radio sector
WiMAX	Worldwide Interoperability for Microwave Access
3GPP	3rd Generation Partnership Project
DSL	Digital Subscriber Line
FDD	Feature-driven development
TDD	Test-driven development
IP	Internet Protocol
EPC	Evolved Packet Core
UE	User Equipment
RNS	Radio Network Subsystem
UTRAN	UMTS Radio Network Access
BSS	Base Station Subsystem
CN	Core Network
NSS	Network Switching Subsystem
RF	Radio frequency
Li-ion	lithium-ion
IMSI	International Mobile Subscriber Identity
MSISDN	Mobile Station International Subscriber Directory Number

RNC	Radio Network Controller
GMSC	Gateway MSC
SGSN	Serving GPRS Support Node
GGSN	Gateway GPRS Support Node
SNDCP	Sub-Network-Dependent Convergence Protocol
LLC	Logical Link Control
MM	Mobility Management
SM	Session Management
CDRs	Call Detail Records
CGF	Charging Gateway Function
SDLC	Software Development Life Cycle
SRS	Software Requirement Specification
DDS	Design Document Specification
UAT	User Acceptance Testing
WM	Waterfall Model
DSDM	Dynamic Systems Development Method
CI	Continuous Integration
RAD	Rapid Application Development
R&D	Research and Development
PICMG	PCI Industrial Computer Manufacturers Group
ATCA	Advanced Telecommunications Computing Architecture
VoIP	Voice over IP
SW Build	Software Build
NASA	National Aeronautics and Space Administration
CVCS	Centralized Version Control System
DVCS	Distributed/Decentralized Version Control System
SVN	Subversion
SCM	Chain Management Software
XP	Extreme Programming

ΑΝΑΦΟΡΕΣ

[1] Wikipedia, the free encyclopedia, "*Advanced Mobile Phone System*" https://en.wikipedia.org/wiki/Advanced_Mobile_Phone_System, October 2010. [Προσπελάστηκε 20/10/2015]
[2] A. Bhawani, "*Idea 3G Tariff Plans – Prepaid & Postpaid Offerings*", http://www.amitbhawani.com/blog/idea-3g-tariff-plans-prepaid-postpaid-offerings/, March 2011. [Προσπελάστηκε 30/1/2016]
[3] R. Gondane, "*What is 4G LTE?*", http://www.coolpctips.com/2012/12/what-is-4g-lte/, Desember 2012. [Προσπελάστηκε 9/12/2015]
[4] Wikipedia, the free encyclopedia, "*LTE (Telecommunication)*", https://en.wikipedia.org/wiki/LTE_telecommunication, January 2016. [Προσπελάστηκε 4/2/2016]
[5] Tektronix, "*K1297-G20 Protocol Tester*", http://www.tek.com/datasheet/umts-software, July 2006. [Προσπελάστηκε 20/1/2016]
[6] I. Poole, " *UMTS UTRA / UTRAN: terrestrial radio access network*", Resources and analysis for electronics, http://www.radio-electronics.com/info/cellulartelecomms/umts/utra-utran-umts-radio-access-network.php. [Προσπελάστηκε 30/12/2015]
[7] Telecom ABC, "*GGSN*", http://www.telecomabc.com/g/ggsn.html, 2005. [Προσπελάστηκε 9/12/2015]
[8] I. Poole, "3G UMTS / WCDMA Network Architecture", Resource and analysis for electronics, http://www.radio-electronics.com/info/cellulartelecomms/umts/umts-wcdma-network-architecture.php. [Προσπελάστηκε 25/9/2015]
[9] Telecom ABC, "*SGSN*", http://www.telecomabc.com/s/sgsn.html, 2005. [Προσπελάστηκε 13/8/2015]
[10] RCR Wireless News, *"Master LTE with the Help of an LTE Network Diagram*", Intelligence on all things wireless, http://www.rcrwireless.com/20140509/wireless/lte-network-diagram, May 2014. [Προσπελάστηκε 5/8/2015]
[11] Standards Coordinating Committee of the IEEE Computer Society, "*IEEE Standard 610.12-1990*", http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=159342, September 1990. [Προσπελάστηκε 23/7/2015]
[12] Wikipedia, the free encyclopedia, "*ISO/IEC 12207 , ISO/IEC 12207*", https://en.wikipedia.org/wiki/ISO/IEC_12207_, ISO/IEC_12207. [Προσπελάστηκε 9/12/2015]
[13] D. Pilone & R. Miles, "*A Brain Friendly Guide: Head First Software Development"*, O'Reilly Media, first Edition, January 2008.
[14] G. D. Everett & R. McLeod Jr., "*Software Testing: Testing Across the Entire Software Development Life Cycle*", Wiley-IEEE Computer Society Pr, first Edition, July 2007.
[15] J. Lewis, "*SDLC 100 Success Secrets - Software Development Life Cycle (SDLC) 100 Most Asked Questions, SDLC Methodologies, Tools, Process and Business Models*", Emereo Publishing, July 2008.
[16] R. Murch, "*The Software Development Lifecycle - A Complete Guide*", Amazon Digital Services LLC, Kindle Edition, April 2012.
[17] A. Seffah & J. Guilliksen & M. C. Desmarais, Springer, "*Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle*

(Human-Computer Interaction Series)", Softcover reprint of hardcover 1st ed. 2005 Edition, January 2011.
[18] J. Boyde, "A Down-To-Earth Guide to SDLC Project Management", CreateSpace Independent Publishing Platform, October 2012.
[19] Wikipedia, the free encyclopedia, "*Agile software development*", https://en.wikipedia.org/wiki/Agile_software_development, December 2015. [Προσπελάστηκε 24/12/2015]
[20] Wikipedia, the free encyclopedia, "*Systems development life cycle*", https://en.wikipedia.org/wiki/Systems_development_life_cycle, December 2010. [Προσπελάστηκε 14/11/2015]
[21] "*Manifesto for Agile Software Development*", http://agilemanifesto.org/, 2001. [Προσπελάστηκε 14/11/2015]
[22] PCM – SIERRA, "*ATCA Design Considerations for Telecommunication Platforms*", PMC-2081221, August 2008.
[23] PICMG, *"AdvancedTCA® Overview"*, Open Modular Computing Standards, https://www.picmg.org/openstandards/advancedtca/, 2014. [Προσπελάστηκε 12/12/2015]
[24] Rajan D., "*Building IP networks using Advanced Telecom Computing Architecture.",* Nortel, Research Triangle Park, North Carolina, USA.19 - 25 June 2007.
[25] J. Ruohonen Nokia Siemens Networks (NSN), "*How one TEM leverages ATCA Hardware Platform Management*", http://picmg.mil-embedded.com/articles/how-hardware-platform-management/, May 2012. [Προσπελάστηκε 14/11/2015]
[26] Nokia Solutions and Networks 2016, "Flexi Network Server", http://networks.nokia.com/portfolio/products/evolved-packet-core/flexi-network-server, 2016. [Προσπελάστηκε 2/2/2016]
[27] V. Tarkiainen & R. Pandey, "Nokia Flexi Network Server Release 15", Executive Presentation, April 2015.
[28] M. Fowler, "Continuous Integration (original version)", http://www.martinfowler.com/articles/originalContinuousIntegration.html, September 2000. [Προσπελάστηκε 13/4/2015]
[29] B. Taylor, "*Rails Deployment and Automation with Shadow Puppet and Capistrano*", Rails machine, https://railsmachine.com/articles/2009/02/10/rails-deployment-and-automation-with-shadowpuppet-and-capistrano/, October 2009. [Προσπελάστηκε 7/10/2015]
[30] D. Brauneis, "*Possible new Working Group – Automation*", open-services.net Community (Mailing list), January 2010. [Προσπελάστηκε 14/11/2015]
[31] E. Rise, "*Continuous deployment in 5 easy steps*", http://radar.oreilly.com/2009/03/continuous-deployment-5-eas.html, March 2009. [Προσπελάστηκε 17/8/2015]
[32] T. Fitz, "*Continuous Deployment at IMVU: Doing the impossible fifty times a day*", http://timothyfitz.com/2009/02/10/continuous-deployment-at-imvu-doing-the-impossible-fifty-times-a-day/, February 2009. [Προσπελάστηκε 14/11/2015]
[33] Wikipedia, the free encyclopedia, "*Continuous integration*", https://en.wikipedia.org/wiki/Continuous_integration, May 2015. [Προσπελάστηκε 28/5/2015]
[34] Git, "*Getting Started - About Version Control*", Software Freedom Conservancy , http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control, . [Προσπελάστηκε 25/5/2015]

[35] G. Lionetti, “What is Version Control: Centralized vs. DVCS”, Atlassian Blog, http://blogs.atlassian.com/2012/02/version-control-centralized-dvcs/, February 2012. [Προσπελάστηκε 5/5/2015]
[36] M. Mason, “*Pragmatic Guide to Subversion (Pragmatic Programmers)*”, first Edition, Pragmatic Bookshelf, December 2010.
[37] C. M. Pilato & B. Collins-Sussman & B. W. Fitzpatrick, “*Version Control with Subversion*”, September 2008.
[38] G. Rooney & D. Berlin, “*Practical Subversion (Expert's Voice in Open Source)*”, second ed.2007 Edition, Apress, November 2006.
[39] J. Loeliger & M. McCullough, “*Version Control with Git: Powerful tools and techniques for collaborative software development*”, second Edition, O'Reilly Media, August 2012.
[40] U. B. Meding, “*Getting Started With Continuous Integration*”, Uwe Meding blog, http://uwemeding.com/getting-started-with-continuous-integration/, February 2015. [Προσπελάστηκε 14/11/2015]
[41] T. Roomer et al., “*Why developers love CI: A guide to loving Continuous Integration*”, tech. rep., ZeroTurnaround, January 2013.
[42] J. F. Smart, “*Jenkins: The Definitive Guide*”, O'Reilly Media, July 2011.
[43] K. Kawaguchi & D. Beck, “*Distributed builds*”, https://wiki.jenkins-ci.org/display/JENKINS/Distributed+builds/, December 2015. [Προσπελάστηκε 20/9/2015]
[44] Atlassian web site, “*Build, Test, Deploy*”, http://www.atlassian.com/software/bamboo/overview/, 2016. [Προσπελάστηκε 28/2/2016]
[45] Source of Acquisition NASA Johnson Space Center, “*Error Cost Escalation Through the Project Life Cycle*”, 2010.
[46] Microsoft MSDN, “*Code Metrics Values*”, https://msdn.microsoft.com/en-us/library/bb385914.aspx, . [Προσπελάστηκε 14/11/2015]
[47] Wikipedia, the free encyclopedia, “*Software metric*”, https://en.wikipedia.org/wiki/Software_metric, January 2016. [Προσπελάστηκε 27/2/2016]
[48] Wikipedia, the free encyclopedia, “*Static program analysis*”, https://en.wikipedia.org/wiki/Static_program_analysis, March 2016. [Προσπελάστηκε 6/3/2016]
[49] D. Kafura & G. R. Reddy, “*The Use of Software Complexity Metrics in Software Maintenance*”, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-13, NO. Three, March 1987.
[50] M. Liljeson & A. Mohlin, “*Faculty of Computing: Software defect prediction using machine learning on test and source code metrics*”, Blekinge Institute of Technology SE–371 79 Karlskrona Sweden, June 2014.