TRACER
Κωδικός Έργου: 09ΣΥΝ-72-942

Ανάλυση Απαιτήσεων Πλατφόρμας TRACER
Παραδοτέο έργου

Ενότητα Εργασίας:	E1: State of the Art - Ανάλυση Απαιτήσεων
Αριθμός Παραδοτέου:	1.2
Συντονιστής:	SING
Συντελεστές:	SING, ΙΤΕ-ΙΠ, ΑΠΘ-ΤΠ, ΟΠΑ-ΔΕΤ
Ημερομηνία υποβολής:	31 Ιανουαρίου 2012
Ημερομηνία παράδοσης:	30 Μαρτίου 2012
Αναγνωριστικό εγγράφου:	TRACER_Π_1.2

ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΕΥΡΩΠΑΪΚΟ ΤΑΜΕΙΟ ΠΕΡΙΦΕΡΕΙΑΚΗΣ ΑΝΑΠΤΥΞΗΣ
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Υπουργείο Παιδείας
Διά Βίου Μάθησης και Θρησκευμάτων
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη

Περιεχόμενα

Λίστα Σχημάτων

Λίστα Πινάκων

1 Εισαγωγή

Στο παραδοτέο αυτό περιγράφονται τα αποτελέσματα της ανάλυσης απαιτήσεων που πραγματοποιήθηκε για την πλατφόρμα TRACER. Για την δομή του παραδοτέου και των επιμέρους ενοτήτων χρησιμοποιήσαμε τις προτεινόμενες πρακτικές που παρουσιάζονται στο «IEEE Recommended Practice for Software Requirements Specifications»[1].

1.1 Σκοπός του παραδοτέου

Ο σκοπός αυτού του παραδοτέου είναι να οριστούν οι απαιτήσεις των χρηστών, λειτουργικές και μη, καθώς οι περιπτώσεις χρήσης της πλατφόρμας TRACER. Ακόμη, σκοπός του παραδοτέου είναι να εντοπισθούν οι αλλαγές που θα γίνουν στις διαδικασίες που ακολουθούνται σήμερα στην διαδικασία ανάπτυξης ενός συστήματος λογισμικού με την εισαγωγή της πλατφόρμας TRACER καθώς και να διευκρινιστεί το εύρος των λειτουργιών της πλατφόρμας TRACER.

1.2 Σύνοψη

Η ασφάλεια ενός συστήματος λογισμικού είναι κάτι που επηρεάζει τόσο τους χρήστες του, όσο και την ομάδα ανάπτυξής του. Οι συνέπειες μιας τρωτότητας σημαίνουν κατ' ελάχιστο χαμένο χρόνο και σημαντικό κόστος για τους χρήστες του λογισμικού. Για την ομάδα ανάπτυξης, οι συνέπειες περιλαμβάνουν μεταξύ άλλων επιπλέον χρόνο και κόστος για την συντήρηση του λογισμικού με σκοπό την θωράκιση από τις τρωτότητες που ανακαλύπτονται, καθώς επίσης και την αποκατάσταση πιθανών βλαβών ή και την αποζημίωση σε ορισμένες περιπτώσεις των χρηστών του λογισμικού. Επιπλέον προβάλουν μια ασυνεπή εικόνα και δημιουργούν αρνητική φήμη για την ποιότητα ενός συστήματος λογισμικού, και κατ' επέκταση του οργανισμού που το αναπτύσσει.

Παρά το γεγονός ότι τα τελευταία χρόνια τα περιβάλλοντα ανάπτυξης λογισμικού έχουν καταφέρει να αποτρέψουν σε ένα βαθμό την εισαγωγή ευπαθειών κατά τη διάρκεια σχεδιασμού και ανάπτυξης ενός συστήματος λογισμικού, καθημερινά ανακαλύπτονται νέες τρωτότητες. Ένας συνηθισμένος λόγος είναι πως το κομμάτι της ασφάλειας συχνά δεν αποτελεί αντικείμενο μελέτης από τη φάση σχεδιασμού ενός συστήματος, αλλά προστίθεται εκ των υστέρων (retrofitted). Ειδικά σε συστήματα παλαιότερων γενεών (legacy systems) αλλά και σε ήδη εγκατεστημένα συστήματα τα οποία τυπικά χρησιμοποιούνται ήδη για μεγάλο διάστημα προτού ανακαλυφθούν οι αδυναμίες τους, η θωράκισή και η διαρκής ενημέρωση τους είναι μια κοστοβόρα και επίπονη διαδικασία.

Η διαδικασία ελέγχου του λογισμικού για αδυναμίες είναι ιδιαίτερα πολύπλοκη και πολλοί προγραμματιστές δεν είναι εξοικειωμένοι με βέλτιστες πρακτικές ανάπτυξης ασφαλούς λογισμικού, ενώ και η χρήση συστημάτων που εντοπίζουν συνηθισμένες ευπάθειες είναι δύσκολη και απαιτεί χρόνο που πολλές φορές δε διαθέτει μια ομάδα ανάπτυξης λόγω πιεστικών χρονοδιαγραμμάτων. Για τους παραπάνω λόγους, η – έστω και μερική – αυτοματοποίηση της διαδικασίας ελέγχου και θωράκισης του λογισμικού θα προσέφερε πολλαπλά οφέλη.

Βασικοί στόχοι της πλατφόρμας TRACER είναι ο έλεγχος συστημάτων παλαιών γενεών (legacy code) για πιθανές αδυναμίες και η εισήγηση πρακτικών λύσεων που θα θωρακίζουν το σύστημα αυτό από τις πλέον διαδεδομένες διαδικτυακές επιθέσεις. Ο έλεγχος θα γίνεται κατά βάση με την χρήση μεθόδων στατικής ανάλυσης. Συγκεκριμένα, η πλατφόρμα TRACER, θα ελέγχει τον κώδικα μιας εφαρμογής (είτε πηγαίο, είτε μεταγλωττισμένο) για αδυναμίες που μπορεί να οδηγήσουν σε ρήγματα ασφάλειας και θα ενημερώνει τους προγραμματιστές της για την ύπαρξή τους. Για να γίνει κάτι τέτοιο, η πλατφόρμα TRACER θα μπορεί να χρησιμοποιεί διάφορα υπάρχοντα εργαλεία ανάλογα με την περίπτωση της εφαρμογής που ελέγχει[2]. Επίσης, θα υλοποιηθεί με τέτοιο τρόπο ώστε να μπορεί να εκτελεσθεί και για παλαιότερες εκδόσεις της εφαρμογής. Με αυτό τον τρόπο θα δίνει μια συνολική εικόνα για το έργο και τις αδυναμίες του παράγοντας πληροφορίες που θα μπορούσαν να φανούν χρήσιμες για τους διαχειριστές μελλοντικών έργων. Έχοντας σαν βάση τις πιο διαδεδομένες επιθέσεις

---

[1] IEEE Recommended Practice for Software Requirements Specifications, The Institute of Electrical and Electronics Engineers, 1998, USA

[2] http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

στο διαδίκτυο[3], αλλά και τα αποτελέσματα της στατικής ανάλυσης, η πλατφόρμα TRACER θα προτείνει διάφορες υπάρχουσες λύσεις για την προστασία της εφαρμογής.

Εδώ θα πρέπει να σημειώσουμε πως ως legacy code εφαρμογή, δεν εννοούμε αναγκαία μια παλαιά εφαρμογή που έχει αναπτυχθεί σε μια παρωχημένη γλώσσα προγραμματισμού, αλλά σε μια εφαρμογή που τυπικά *δεν έχει υποστεί τον έλεγχο* που προσφέρει η πλατφόρμα μας.

Ιδανικά, την πλατφόρμα TRACER θα μπορούν να χρησιμοποιήσουν προγραμματιστές (programmers) και ελεγκτές (code reviewers) και κατά την διάρκεια της υλοποίησης ενός έργου.

1.3 Δομή του παραδοτέου

Στο παρόν παραδοτέο αναλύεται η μεθοδολογία που ακολουθήθηκε προκειμένου να συλλεχθούν οι απαιτήσεις των χρηστών και οι σύγχρονες μεθοδολογίες ανάπτυξης και ελέγχου λογισμικού. Στη συνέχεια, αναλύεται πως θα αλλάξουν οι διαδικασίες αυτές με την χρήση της πλατφόρμας TRACER και περιγράφονται εκτενώς οι περιπτώσεις χρήσης του συστήματος με τους δράστες σε αυτές. Στην επόμενη ενότητα αναλύονται οι λειτουργικές και μη απαιτήσεις των χρηστών και στην τελευταία ενότητα αντιστοιχίζονται οι λειτουργίες του συστήματος με της απαιτήσεις αυτές. Το παραδοτέο δομείται με αυτό τον τρόπο καθώς κάθε επόμενη ενότητα χρειάζεται τα στοιχεία που αναλύονται στις προηγούμενες ενότητες.

Συγκεκριμένα το παραδοτέο έχει την ακόλουθη δομή:

1. Η Μεθοδολογία Εξαγωγής Απαιτήσεων, που αναλύει τον τρόπο με τον οποίο οι πληροφορίες που μας παρείχαν οι χρήστες μετατράπηκαν σε συγκεκριμένες απαιτήσεις χρηστών (Ενότητα 2).
2. Η Παρούσα και Μελλοντική Κατάσταση, δηλαδή με ποιον τρόπο μια εταιρία πληροφορικής αναπτύσσει και ελέγχει λογισμικό σήμερα καθώς και πως θα αλλάξουν οι διαδικασίες αυτές με την χρήση της πλατφόρμας TRACER (Ενότητα 3).
3. Οι Περιπτώσεις Χρήσης και οι Δράστες σε αυτές (Ενότητα 4)
4. Οι Απαιτήσεις των Χρηστών που προέκυψαν από την προαναφερόμενη διαδικασία Εξαγωγής Απαιτήσεων (Ενότητα 5).
5. Η Περιγραφή των διεπαφών του συστήματος (Ενότητα 6).
6. Τα Συμπεράσματα του παραδοτέου (Ενότητα 7).
7. Το Παράρτημα Α που περιλαμβάνει όλες τις απαραίτητες πληροφορίες για την μεθοδολογία Rational unified process (RUP) και την Quality Function Deployment (QFD) προσέγγιση που χρησιμοποιούνται στο παραδοτέο.
8. Το Παράρτημα Β που περιλαμβάνει τις Ερωτήσεις που απαντήθηκαν προκειμένου να εξάγουμε τις απαιτήσεις των χρηστών.
9. Το Παράρτημα Γ που αποτελείται από το Γλωσσάρι, δηλαδή τον ορισμό διάφορων εννοιών που χρησιμοποιούνται στο παραδοτέο.
10. Το Παράρτημα Δ που περιλαμβάνει τη Βιβλιογραφία.

1.4 Σύνδεση με άλλα παραδοτέα

- Στο παρόν παραδοτέο συμβουλευτήκαμε το παραδοτέο Π1.1, προκειμένου να εντοπισθούν χρήσιμες πληροφορίες για τον τρόπο με τον οποίο αναπτύσσεται και ελέγχεται το λογισμικό σε οργανισμούς σήμερα καθώς και με ποιον τρόπο γίνεται ο έλεγχος και η θωράκιση συστημάτων παλαιών γενεών.
- Τα στοιχεία που προκύπτουν από τον παρόν παραδοτέο θα χρησιμοποιηθούν στα επόμενα στάδια της διαδικασίας και συγκεκριμένα στο παραδοτέο «Π2.1: Σχεδιασμός Πλατφόρμας TRACER».

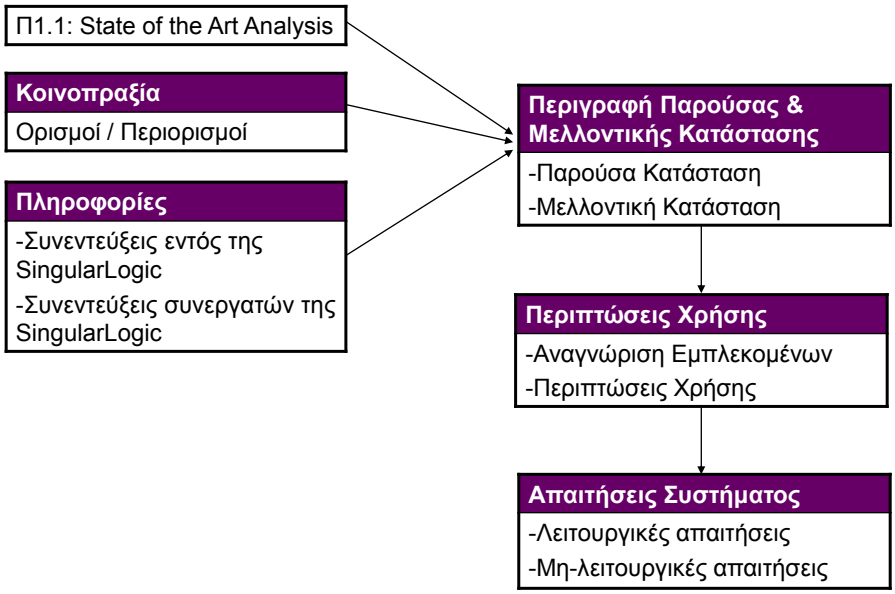
2 Μεθοδολογία Εξαγωγής Απαιτήσεων

Για την συλλογή και εξαγωγή των απαιτήσεων, εφαρμόστηκε η "Rational Unified Process" (RUP) ως μεθοδολογία ανάπτυξης λογισμικού σε συνδυασμό με την Quality Function Deployment

[3] https://www.owasp.org/index.php/Top_10_2010-Main

(QFD) προσέγγιση για τον ορισμό και την ανάπτυξη των περιπτώσεων χρήσης και των απαιτήσεων του συστήματος (Παράρτημα Α).

Η μεθοδολογία για τον ορισμό των απαιτήσεων ξεκινάει από την εύρεση των σεναρίων των χρηστών. Στη συνέχεια, για την οριστικοποίηση των απαιτήσεων από την πλατφόρμα TRACER ακολουθούνται τα βήματα που παρουσιάζονται στο Σχήμα 1.



Σχήμα 1: Βήματα Μεθοδολογίας Εξαγωγής Απαιτήσεων

Σε πρώτη φάση, μελετήθηκε το παραδοτέο Π1.1: State of the Art Analysis και λήφθηκαν υπόψη οι ορισμοί και οι περιορισμοί που έθεσε η κοινοπραξία. Στη συνέχεια οι πιθανοί χρήστες, έδωσαν τις απαραίτητες πληροφορίες προκείμενου να οριστούν τα σενάρια των χρηστών, δηλαδή οι περιγραφές της παρούσας και μελλοντικής κατάστασης (με την χρήση της πλατφόρμας TRACER). Οι τεχνικοί εταίροι του έργου ερμήνευσαν αυτά τα σενάρια σε περιπτώσεις χρήσης και απαιτήσεις της πλατφόρμας TRACER. Τέλος, βασισμένοι στις περιπτώσεις χρήσης και στις απαιτήσεις, ορίστηκαν οι μη-λειτουργικές απαιτήσεις του συστήματος.

2.1 Ορισμός Ενδιαφερόμενων / Επωφελούμενων και Εμπλεκομένων (stakeholders) στην πλατφόρμα TRACER

Η πλατφόρμα TRACER έχει ως στόχο την βελτιώση και την ενισχύση της ποιότητας και ασφάλειας των υπηρεσιών και προϊόντων που προσφέρονται από μια ευρεία γκάμα εταιριών. Οι εταιρίες που θα επωφεληθούν από την χρήση της πλατφόρμας TRACER θα είναι οι εταιρίες πληροφορικής που αναπτύσσουν εφαρμογές (διαδικτυακές και μη) και υπηρεσίες λογισμικού, καθώς χρειάζονται συνεχείς ελέγχους στον κώδικα που παράγεται και σε αυτό αναμένεται να παίξει σημαντικό ρόλο η πλατφόρμα TRACER.

Μέσα από την επιπλέον ασφάλεια που θα έχουν οι προσφερόμενες υπηρεσίες και τα προϊόντα θα επωφεληθούν και οι εταιρίες χρήστες, δηλαδή οι εταιρίες που χρησιμοποιούν αυτές τις υπηρεσίες και εφαρμογές, καθώς θα αποφεύγουν πιθανές κακόβουλες επιθέσεις. Για την πλατφόρμα TRACER θα ενδιαφερθούν και εταιρίες που ολοκληρώνουν συστήματα συνδυάζοντας λογισμικό και υλικό από διαφορετικές πηγές προέλευσης, καθώς θα αποφεύγουν πιθανά ρήγματα ασφαλείας κατά την ολοκλήρωση των επιμέρους μονάδων. Τέλος, θα επωφεληθούν όλοι οι οργανισμοί που χρησιμοποιούν συστήματα παλαιών γενεών (legacy systems) καθώς μπορεί τα συστήματα αυτά να παρουσιάζουν ρήγματα ασφαλείας.

Οι Εμπλεκόμενοι (stakeholders) σε ένα έργο είναι οι οντότητες εντός ή εκτός ενός οργανισμού που χρηματοδοτούν το έργο, που ενδιαφέρονται ή κερδίζουν από την επιτυχή ολοκλήρωση του έργου ή επηρεάζουν, θετικά ή αρνητικά, την ολοκλήρωση αυτού. Εμπλεκόμενοι στο έργο TRACER είναι τόσο οργανισμοί όπως η SingularLogic, το ΟΠΑ-ΔΕΤ, το ΑΠΘ-ΤΠ και το ΙΤΕ-ΙΠ.

Η SingularLogic χρηματοδοτεί προσωπικό που παίρνει μέρος στον σχεδιασμό της πλατφόρμας TRACER. Ακόμη, ενδιαφέρεται για την επιτυχή ολοκλήρωση της πλατφόρμας TRACER ώστε να την ενσωματώσει στη διαδικασία ανάπτυξης και συντήρησης των συστημάτων της και να προσφέρει ενισχυμένες υπηρεσίες στους πελάτες της. Το ΟΠΑ-ΔΕΤ, το ΑΠΘ-ΤΠ και το ΙΤΕ-ΙΠ θα σχεδιάσουν και θα υλοποιήσουν μέρος της πλατφόρμας TRACER, ενώ στη συνέχεια θα χρησιμοποιήσουν την πλατφόρμα TRACER για ερευνητικούς σκοπούς.

2.2 Συνεντεύξεις Εμπλεκομένων

Μέσω συνεντεύξεων των εμπλεκομένων, μπορεί να γίνουν κατανοητές οι επιχειρηματικές διαδικασίες και οι ανάγκες που προκύπτουν από αυτές μέσα από ένα ρεαλιστικό πρίσμα. Οι συνεντεύξεις έγιναν με την μορφή ημι-δομημένης συνέντευξης τόσο σε εργαζομένους τις SingularLogic όσο και σε εταιρείες από το δίκτυο συνεργατών της. Αυτό έδωσε την δυνατότητα να ερευνηθούν διάφορα θέματα που δεν είχαν προβλεφθεί στην αρχική λίστα των ερωτήσεων. Τα άτομα που ερωτήθηκαν σχετίζονται με την υλοποίηση και τον έλεγχο λογισμικού και κατέχουν διάφορους ρόλους. Ερωτήθηκαν Προγραμματιστές, Υπεύθυνοι Ποιότητας Λογισμικού, Υπεύθυνοι Ασφαλείας Λογισμικού και Συστημάτων και Υπεύθυνοι Πληροφοριακών Συστημάτων. Τα άτομα αυτά έχουν μακρόχρονη εμπειρία στην υλοποίηση έργων και προϊόντων πληροφορικής.

Σε συνδυασμό με τις γενικές ερωτήσεις, παρουσιάσαμε στους εμπλεκόμενους την ιδέα της πλατφόρμας TRACER και ζητήσαμε την γνώμη τους σχετικά με τις λειτουργίες που θα επιθυμούσαν να έχει η πλατφόρμα TRACER προκειμένου να καλύπτει τις ανάγκες τους.

Με την συλλογή των απαντήσεων συγκεντρώσαμε τα εξής αποτελέσματα:

- Τα συνηθέστερα περιβάλλοντα ανάπτυξης είναι τα:
  - .NET
  - Oracle Forms
  - Java SE και J2EE
- Τα περισσότερα συστήματα παλαιών γενεών που διαθέτει η SingularLogic είναι υλοποιημένα στις ακόλουθες γλώσσες προγραμματισμού:
  - C#
  - Delphi
  - Java
  - Visual Basic 6.0

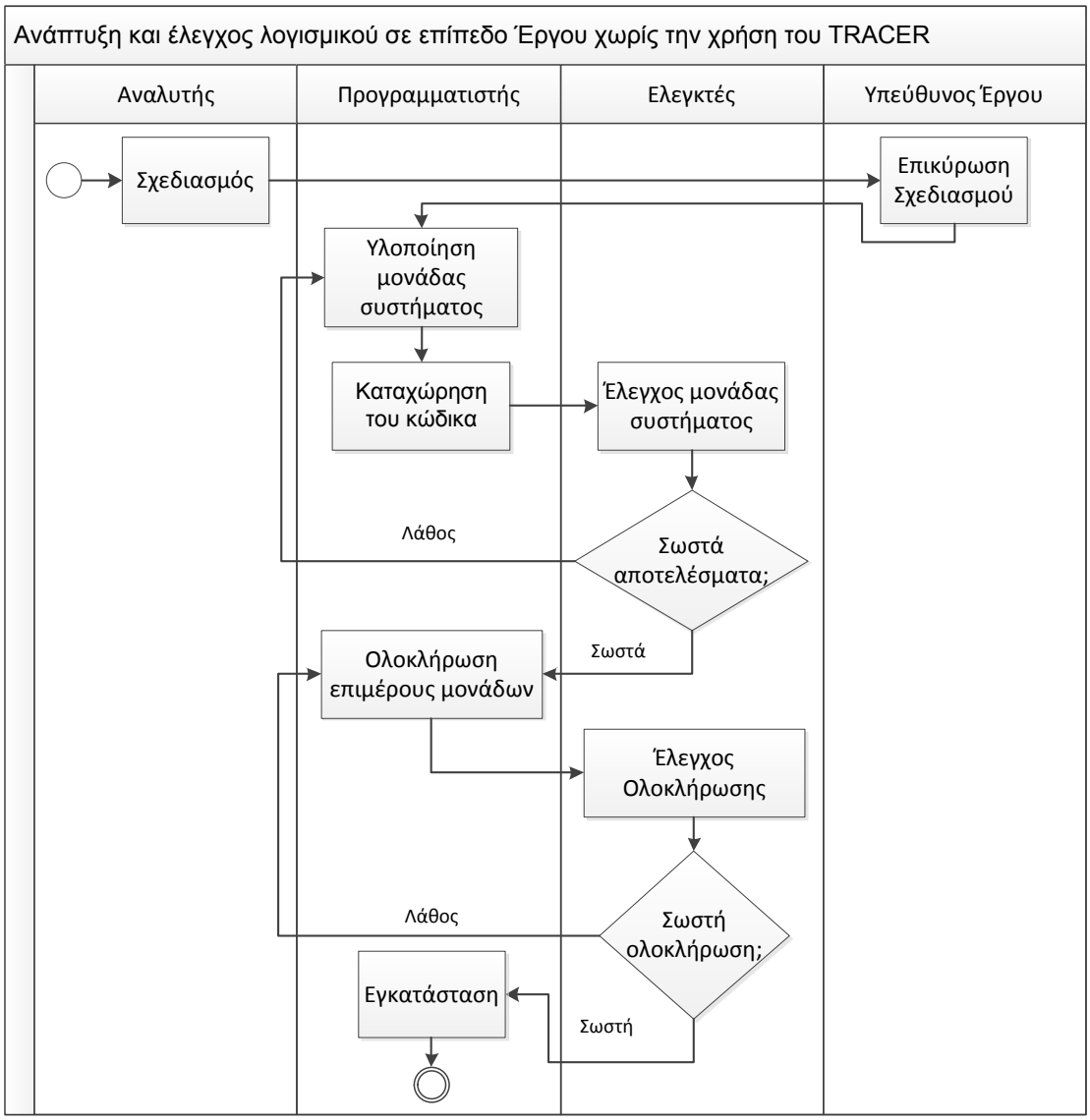
  Περισσότερες πληροφορίες για τον τρόπο αξιολόγησης καθώς και αναλυτικά αποτελέσματα μπορείτε να δείτε στο Παράρτημα Γ**.**
- Η πλειοψηφία των ερωτηθέντων ακολουθεί την μεθοδολογία RUP για το σχεδιασμό και την υλοποίηση των συστημάτων τους.
- Η μοναδική πιστοποίηση ασφάλειας που επικυρώνει την διαδικασία σχεδιασμού και υλοποίησης συστημάτων πληροφορικής είναι η ISO 27001. Τέτοια πιστοποίηση είχε μόνο μία από τις εταιρείες που συμμετείχαν στην έρευνα (SingularLogic).
- Οι έλεγχοι για θέματα ασφάλειας γίνονται στο στάδιο της υλοποίησης των μονάδων του συστήματος και της ολοκλήρωσης και γίνονται παράλληλα με τον έλεγχο λειτουργικότητας. Λεπτομερής παρουσίαση της όλης διαδικασίας βρίσκεται στην ενότητα 3.1.
- Σε επίπεδο κώδικα συστημάτων παλαιών γενεών δεν γίνονται έλεγχοι επιθέσεων διαδικτύου.
- Ο κώδικας δεν ελέγχεται είτε με στατικές είτε με δυναμικές μεθόδους τρωτότητες, ιδιαίτερα για cross-site scripting (XSS) επιθέσεις ενώ συγκεκριμένα υπομέρη του προς υλοποίηση συστήματος (που κρίνονται υψηλής κρισιμότητας) ελέγχονται αποσπασματικά για αδυναμίες που μπορούν να οδηγήσουν σε ρήγματα ασφάλειας.
- Σε επίπεδο υλοποίησης και λειτουργίας ενός προϊόντος για να εξασφαλιστεί η ασφάλεια ενός συστήματος παρακολουθούνται τα αρχεία καταγραφής του συστήματος. Δεδομένου του πλήθος των μονάδων ενός συστήματος είναι σημαντική η αυτοματοποίηση τόσο της

διαδικασίας παρακολούθησης όσο και της διαδικασίας ενημέρωσης σε περίπτωση πιθανού προβλήματος ασφάλειας.

- Τέλος, καταγράψαμε και τη χρήση σύγχρονων συστημάτων τείχους προστασίας που παρέχουν ισχυρούς μηχανισμούς αυθεντικοποίησης καθώς και καταγραφής όλων των πιθανών «διαδρομών» που εξουσιοδοτείται ο κάθε χρήστης να εκτελέσει. Αξίζει να σημειωθεί ότι το κόστος αγοράς και χρήσης τέτοιων συστημάτων είναι ιδιαίτερα υψηλό, γεγονός που τα καθιστά απαγορευτικά για μικρομεσαίες επιχειρήσεις.

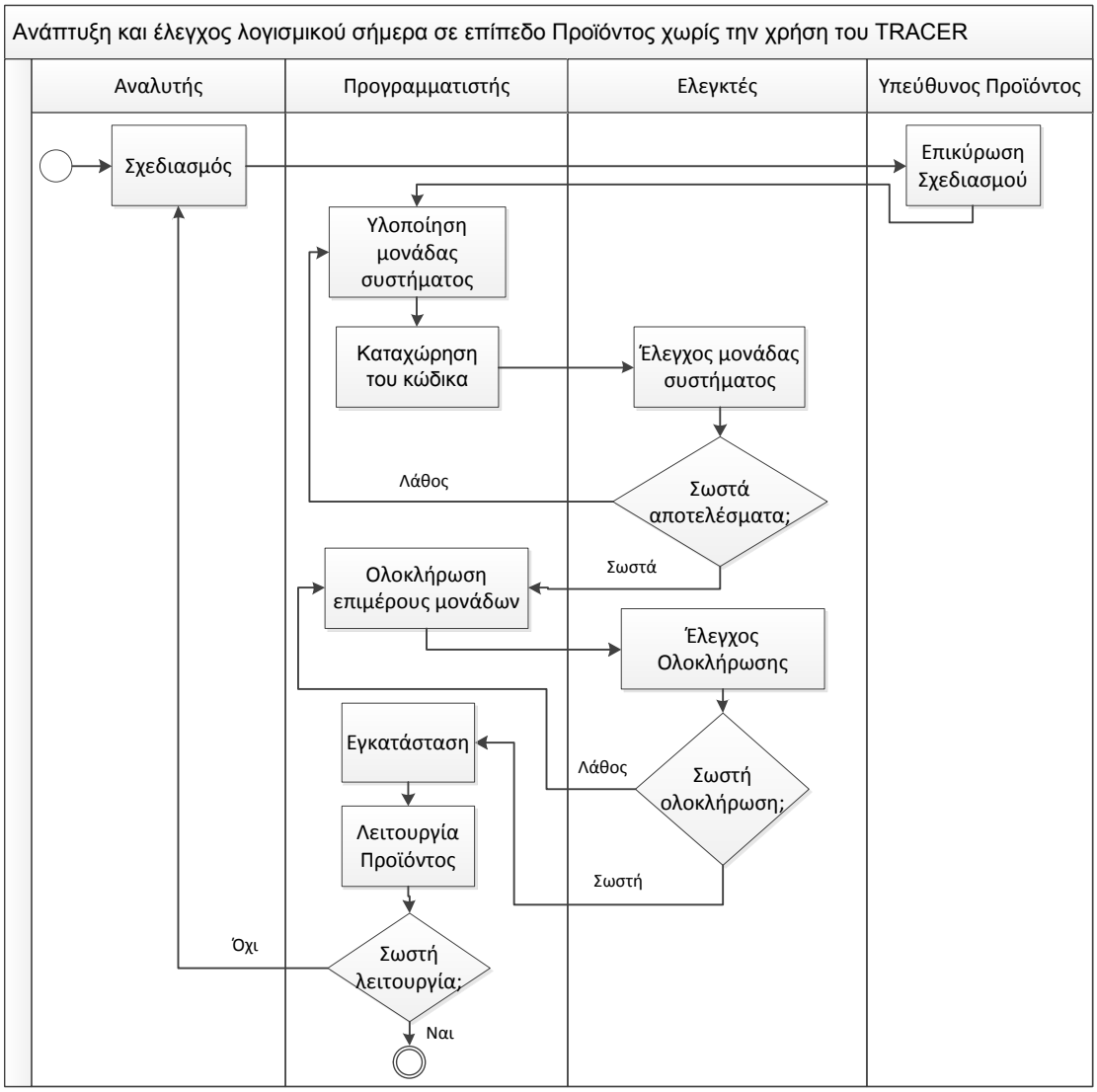
Μέσω της διαδικασίας των συνεντεύξεων και έπειτα από την συγκέντρωση και ανάλυση των αποτελεσμάτων τους, διαμορφώσαμε μία ξεκάθαρη εικόνα του τρόπου υλοποίησης λογισμικού, την παρούσα κατάσταση που περιγράφεται στην Ενότητα 3.1. Επιπλέον, εντοπίσαμε τα στάδια της διαδικασίας αυτής που παρουσιάζουν ελλείψεις και τον τρόπο που θα τα ενισχύσει η πλατφόρμα TRACER και μέσω αυτών εξάγαμε τις Περιπτώσεις Χρήσης της πλατφόρμας TRACER.

3 Περιγραφή παρούσας και μελλοντικής κατάστασης



Σχήμα 2: Διαδικασία Ανάπτυξης και Ελέγχου Λογισμικού σε επίπεδο Έργου

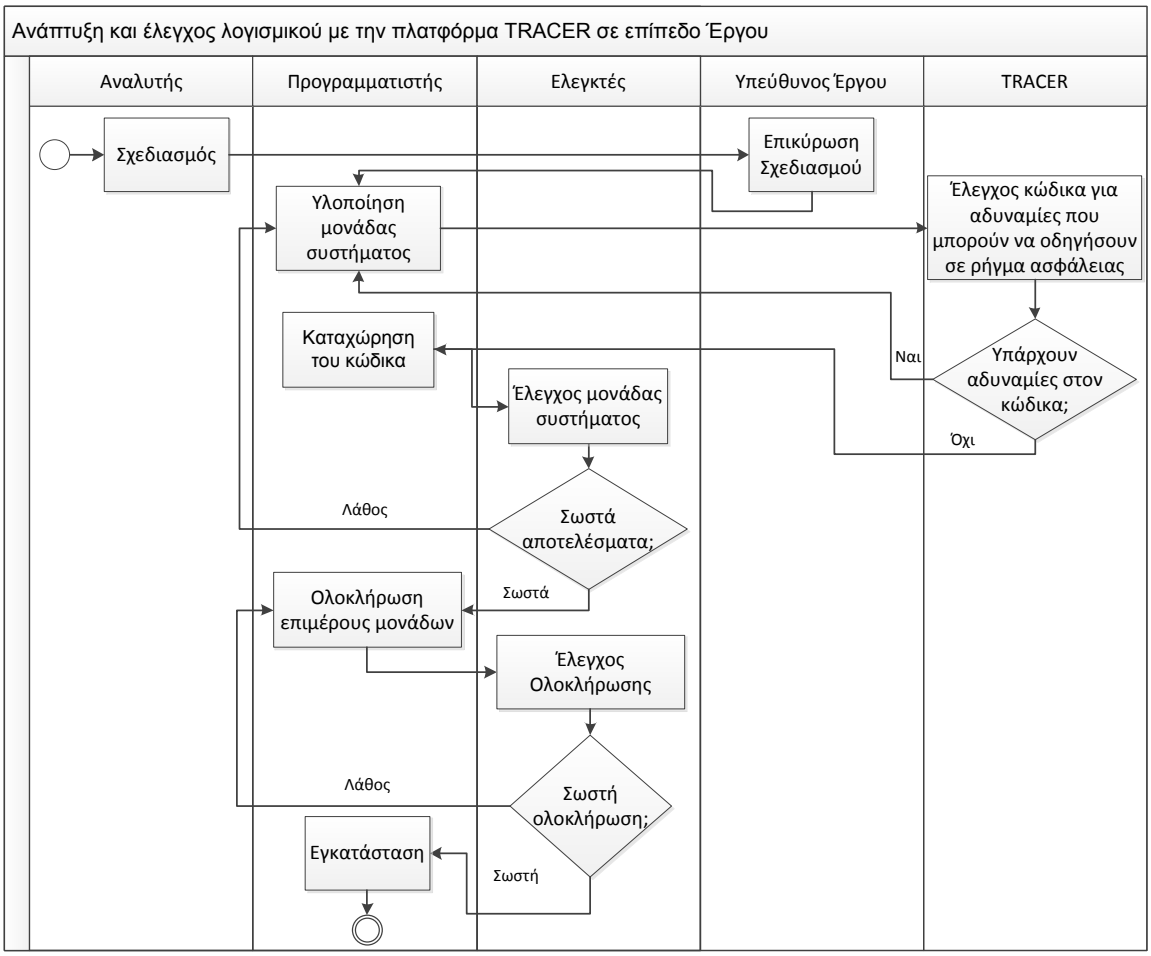
Σε επίπεδο Έργου, όταν πρόκειται για την υλοποίηση ενός πληροφοριακού συστήματος, τα βήματα που ακολουθούνται φαίνονται στο Σχήμα 2. Αρχικά αναλύονται οι απαιτήσεις των ενδιαφερομένων και σχεδιάζεται το πληροφοριακό σύστημα από τους αναλυτές. Στην συνέχεια ο Υπεύθυνος Έργου επικυρώνει τον σχεδιασμό και οι προγραμματιστές προχωρούν στην υλοποίηση των μονάδων του συστήματος και στην καταχώρηση του κώδικα (commit). Οι ελεγκτές πρέπει να ελέγξουν για το αν γίνονται οι απαραίτητες λειτουργίες και αντίστοιχα η διαδικασία προχωράει στο επόμενο βήμα ή οι προγραμματιστές αναλαμβάνουν να διορθώσουν ότι λάθη υπάρχουν. Εφόσον πραγματοποιείται επιτυχώς το προηγούμενο στάδιο, ολοκληρώνονται οι επιμέρους μονάδες και ελέγχεται αν η ολοκλήρωση έγινε επιτυχώς. Αν ναι, ολοκληρώνεται η διαδικασία με την εγκατάσταση του πληροφοριακού συστήματος.



Σχήμα 3: Διαδικασία Ανάπτυξης και Ελέγχου Λογισμικού σε επίπεδο Προϊόντος

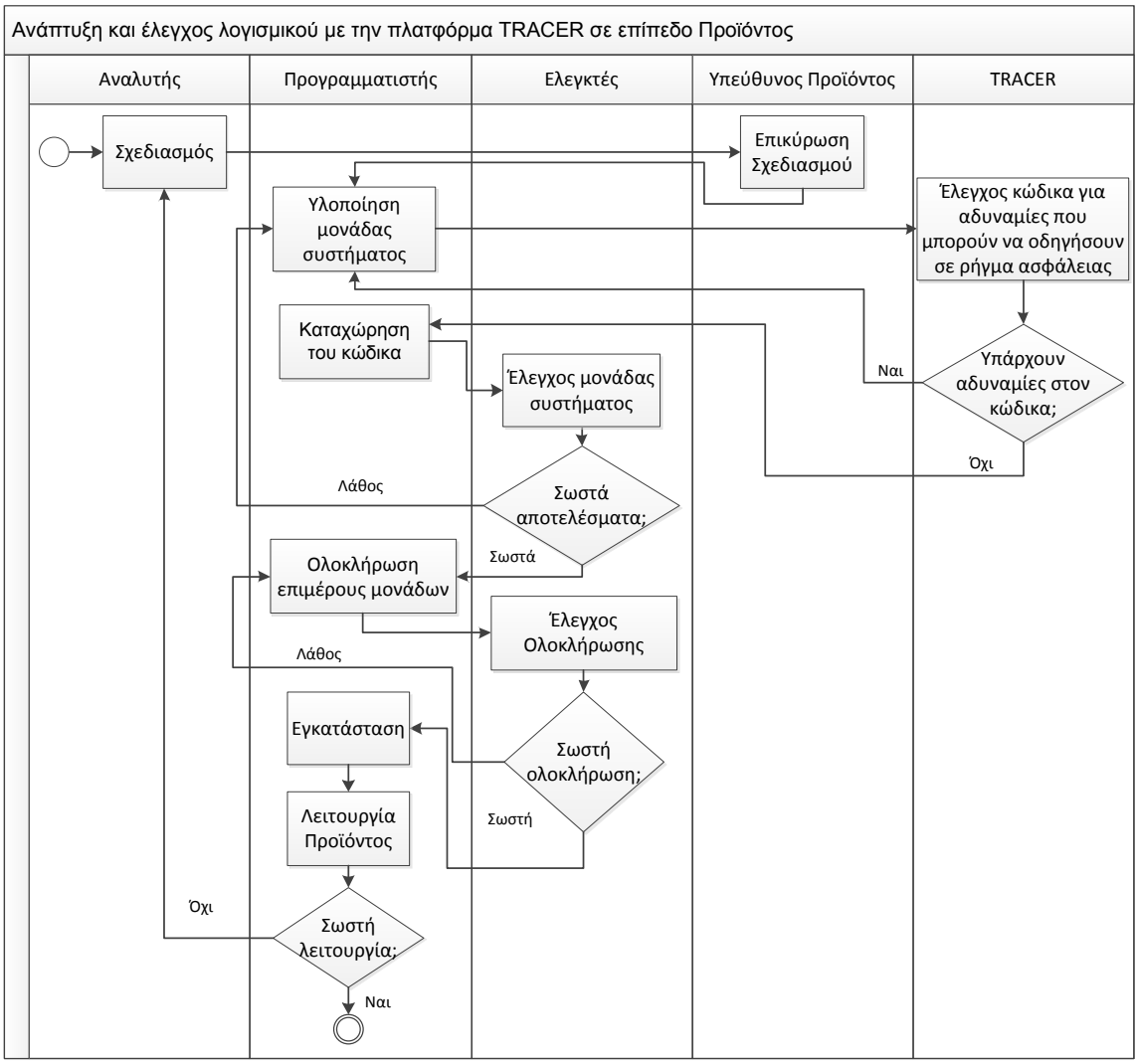
Σε επίπεδο Προϊόντος, όταν πρόκειται για την υλοποίηση ενός πληροφοριακού συστήματος, τα βήματα που ακολουθούνται φαίνονται στο Σχήμα 3. Ουσιαστικά η διαδικασία περιλαμβάνει τα ίδια βήματα με την υλοποίηση ενός πληροφοριακού συστήματος σε επίπεδο έργου με τη διαφορά ότι στο τέλος της διαδικασίας δεν αρκεί η εγκατάσταση του πληροφοριακού συστήματος αλλά ελέγχεται και η εύρυθμη λειτουργία του. Αν το προϊόν δεν λειτουργεί σωστά η διαδικασία ξεκινάει από την αρχή προκειμένου να βρεθεί το πρόβλημα είτε σε επίπεδο σχεδίασης είτε σε επίπεδο υλοποίησης και ενοποίησης του.

3.1 Ανάπτυξη Λογισμικού με την Χρήση της Πλατφόρμας TRACER



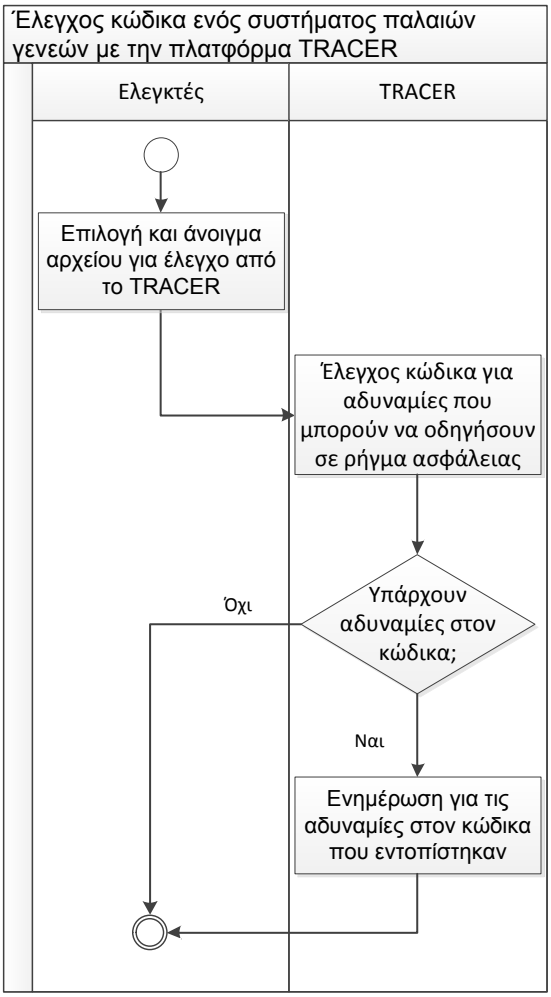
Σχήμα 4: Διαδικασία Ανάπτυξης και Ελέγχου Λογισμικού σε επίπεδο Έργου με την πλατφόρμα TRACER

Σε επίπεδο Έργου, με την χρήση της πλατφόρμας TRACER η διαδικασία για την υλοποίηση ενός πληροφοριακού συστήματος παραμένει σχεδόν ίδια εκτός ενός σημείου στο οποίο παρεμβαίνει η πλατφόρμα TRACER. Μετά την υλοποίηση των μονάδων συστήματος από τους προγραμματιστές και πριν την καταχώρηση του κώδικα, η πλατφόρμα TRACER ελέγχει τον κώδικα για αδυναμίες που μπορεί να οδηγήσουν σε ρήγματα ασφάλειας. Αν ο κώδικας βρεθεί ασφαλής πραγματοποιείται η καταχώρηση του και συνεχίζεται η διαδικασία, όπως και πριν. Αν ωστόσο βρεθούν λάθη ασφαλείας, επισημαίνεται στον προγραμματιστή το προβληματικό κομμάτι κώδικα και επαναλαμβάνεται η διαδικασία.



Σχήμα 5: Διαδικασία Ανάπτυξης και Ελέγχου Λογισμικού σε επίπεδο Προϊόντος με την πλατφόρμα TRACER

Σε επίπεδο Προϊόντος, με την χρήση της πλατφόρμας TRACER η διαδικασία για την υλοποίηση ενός πληροφοριακού συστήματος ακολουθεί τα ίδια βήματα με την χωρίς TRACER διαδικασία εκτός ενός σημείου στο οποίο παρεμβαίνει η πλατφόρμα TRACER. Ο κώδικας πριν καταχωρηθεί ελέγχεται για ρήγματα ασφαλείας και διορθώνεται εφόσον χρειαστεί. Διαφέρει επίσης από την διαδικασία που ακολουθείται σε επίπεδο έργου, καθώς στο τέλος ελέγχεται και η λειτουργία του προϊόντος και η όλη διαδικασία επαναλαμβάνεται αν χρειαστεί.



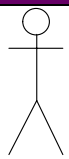
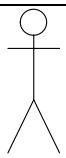
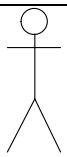
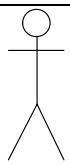
Σχήμα 6: Διαδικασία ελέγχου του κώδικα ενός συστήματος παλαιών γενεών με την πλατφόρμα TRACER

Με την πλατφόρμα TRACER δίνεται η δυνατότητα να ελέγχεται ο κώδικας παλαιών γενεών για αδυναμίες και η διαδικασία ακολουθεί τα βήματα που φαίνονται παραπάνω. Πριν την χρήση του TRACER ο κώδικας παλαιών γενεών δεν ελεγχόταν καθόλου ή ελεγχόταν μόνο από τον εκάστοτε προγραμματιστή με περιορισμό τις γνώσεις αυτού.

4 Ανάλυση Περιπτώσεων Χρήσης

Για την ανάλυση των περιπτώσεων χρήσης της πλατφόρμας TRACER και την ανάλυση των δραστών σε αυτήν χρησιμοποιούμε την γλώσσα UML και της σχηματικές απεικονίσεις, διαγράμματα, που αυτή περιλαμβάνει. Για περισσότερες πληροφορίες ανατρέξτε στο Παράρτημα Β.

4.1 Δράστες στο Σύστημα

Δράστης	Ρόλος	Περιγραφή
 Προγραμματιστής	Προγραμματιστής	Αναπτύσσει τις εφαρμογές και είναι υπεύθυνος για τον ορισμό των έργων που θα παρακολουθούνται, την θωράκιση του κώδικα εφόσον εντοπιστούν ρήγματα ασφαλείας και την δημιουργία λειτουργικών αρθρωμάτων.
 Υπεύθυνος Λειτουργικών Διαδικασιών	Υπεύθυνος Λειτουργικών Διαδικασιών	Είναι υπεύθυνος για τον ορισμό των έργων που θα παρακολουθούνται.
 Διαχειριστής του TRACER	Διαχειριστής της πλατφόρμας TRACER	Είναι υπεύθυνος για την δημιουργία των χρηστών και την διαχείριση των λειτουργικών αρθρωμάτων (plugins).
 Στοιχείο Συστήματος TRACER	Στοιχείο Πλατφόρμας TRACER (Component)	Το Στοιχείο Συστήματος θα εκτελεί όλες τις λειτουργίες που γίνονται αυτόματα χωρίς την συμβολή ανθρώπου, όπως τον έλεγχο του κώδικα για ρήγματα ασφαλείας, την καταγραφή των προβλημάτων που εντοπίζονται και την θωράκιση με τη χρήση βιβλιοθηκών ασφαλείας.

Πίνακας 1: Περιγραφή Δραστών στο Σύστημα

4.2 Κατάλογος Περιπτώσεων Χρήσης του Συστήματος

Για την αναγνώριση των περιπτώσεων χρήσης χρησιμοποιήθηκε η εξής κωδικοποίηση:

Κωδικός Περίπτωσης Χρήσης	ΠΧΑ/Α
ΠΧ	Περίπτωση Χρήσης
Α/Α	Αύξων αριθμός

Πίνακας 2: Κωδικοποίηση Περιπτώσεων Χρήσης

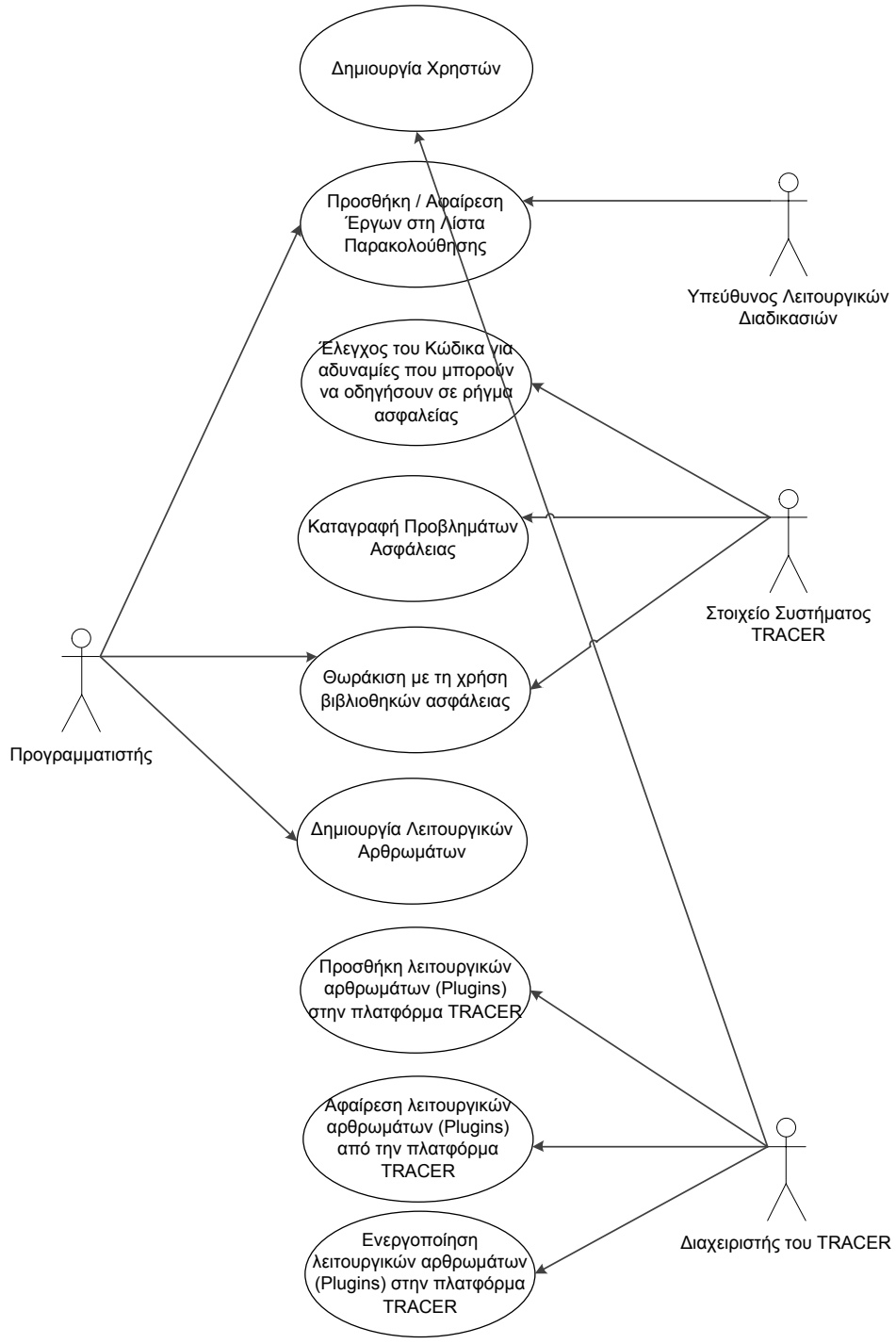
Ένας Κωδικός Περίπτωσης Χρήσης λοιπόν απαρτίζεται από 3 στοιχεία: το πρόθεμα ΠΧ, τον αριθμό της κατηγορίας, και τον αύξοντα αριθμό της περίπτωσης χρήσης εντός της κατηγορίας.

Κωδικός ΠΧ	Τίτλος ΠΧ	Σύντομη Περιγραφή	Βασικοί Δράστες	Σημαντικότητα
ΠΧ01	Δημιουργία Χρηστών	Δημιουργούνται οι χρήστες.	Διαχειριστής της πλατφόρμας TRACER	Υψηλή
ΠΧ02	Προσθήκη / Αφαίρεση Έργων στη Λίστα	Προσθέτονται / αφαιρούνται τα έργα που θα παρακολουθούνται στην λίστα	Υπεύθυνος Λειτουργικών	Υψηλή

	Παρακολούθησης	παρακολούθησης.	Διαδικασιών, Προγραμματιστής	
ΠΧ03	Έλεγχος του Κώδικα για αδυναμίες που μπορούν να οδηγήσουν σε ρήγμα ασφαλείας	Ελέγχεται ο κώδικας για αδυναμίες.	Στοιχείο Πλατφόρμας TRACER	Υψηλή
ΠΧ04	Καταγραφή Προβλημάτων Ασφάλειας	Καταγράφονται τα προβλήματα ασφαλείας που εντοπίστηκαν κατά τον έλεγχο του κώδικα.	Στοιχείο Πλατφόρμας TRACER	Υψηλή
ΠΧ05	Θωράκιση με τη χρήση βιβλιοθηκών ασφάλειας	Η πλατφόρμα προτείνει τη χρήση βιβλιοθηκών που παρέχουν αντίμετρα για τις συγκεκριμένες αδυναμίες.	Στοιχείο Πλατφόρμας TRACER, Προγραμματιστής	Μεσαία
ΠΧ06	Δημιουργία Λειτουργικών Αρθρωμάτων	Δημιουργείται κάποιο λειτουργικό άρθρωμα (Plugin) βασισμένο σε πρότυπο κώδικα που παρέχεται από την πλατφόρμα TRACER, με σκοπό την πραγματοποίηση ανίχνευσης νέων ειδών τρωτοτήτων ή την ενσωμάτωση εξωτερικών εργαλείων.	Προγραμματιστής	Υψηλή
ΠΧ07	Προσθήκη λειτουργικών αρθρωμάτων (Plugins) στην πλατφόρμα TRACER	Προστίθεται ένα νέο λειτουργικό άρθρωμα στην πλατφόρμα.	Διαχειριστής της πλατφόρμας TRACER	Υψηλή
ΠΧ08	Αφαίρεση λειτουργικών αρθρωμάτων (Plugins) από την πλατφόρμα TRACER	Αφαιρείται ένα λειτουργικό άρθρωμα από την πλατφόρμα.	Διαχειριστής της πλατφόρμας TRACER	Χαμηλή
ΠΧ09	Ενεργοποίηση λειτουργικών αρθρωμάτων (Plugins) στην πλατφόρμα TRACER	Καθορίζονται κανόνες με βάση τους οποίους θα ενεργοποιείται η εκτέλεση ενός λειτουργικού αρθρώματος.	Διαχειριστής της πλατφόρμας TRACER	Μεσαία

Πίνακας 3: Λίστα Περιπτώσεων Χρήσης

4.3 Περιγραφή Περιπτώσεων Χρήσης του Συστήματος



Σχήμα 7: Διάγραμμα Περιπτώσεων Χρήσης

Ονομασία Περίπτωσης Χρήσης	Δημιουργία Χρηστών
Κωδικός	ΠΧ01
Σύντομη Περιγραφή	Δημιουργούνται οι χρήστες.
Βασικοί Δράστες	Διαχειριστής της πλατφόρμας TRACER
Δευτερεύοντες Δράστες	-

Σημαντικότητα	Υψηλή
Προϋποθέσεις	-
Κύρια Ροή	• Ο Διαχειριστής της πλατφόρμας TRACER επιλέγει την καρτέλα Χρήστες στην Παραμετροποίηση του συστήματος. • Ο Διαχειριστής της πλατφόρμας TRACER επιλέγει την Δημιουργία Χρήστη. • Ο Διαχειριστής της πλατφόρμας TRACER δίνει τα στοιχεία του καινούριου χρήστη και το email του. • Ο Διαχειριστής της πλατφόρμας TRACER επιλέγει την Αποθήκευση Αλλαγών.
Μετασυνθήκες	Δημιουργείται ένας νέος Χρήστης αν όλα τα στοιχεία που εισήγαγε ο Διαχειριστής της πλατφόρμας TRACER είναι σωστά.
Εναλλακτικές Ροές	Να εισάγει λάθος στοιχεία, όνομα ή/ και email ο Διαχειριστής της πλατφόρμας TRACER και το σύστημα να εμφανίσει μήνυμα λάθους.

Ονομασία Περίπτωσης Χρήσης	Προσθήκη / Αφαίρεση Έργων στη Λίστα Παρακολούθησης
Κωδικός	ΠΧ02
Σύντομη Περιγραφή	Προσθέτονται / αφαιρούνται τα έργα που θα παρακολουθούνται στην λίστα παρακολούθησης.
Βασικοί Δράστες	Υπεύθυνος Λειτουργικών Διαδικασιών, Προγραμματιστής
Δευτερεύοντες Δράστες	-
Σημαντικότητα	Υψηλή
Προϋποθέσεις	-
Κύρια Ροή	• Ο Υπεύθυνος Λειτουργικών Διαδικασιών ή ο Προγραμματιστής επιλέγει την καρτέλα Λίστα Παρακολούθησης στην Παραμετροποίηση του συστήματος. • Ο Υπεύθυνος Λειτουργικών Διαδικασιών ή ο Προγραμματιστής επιλέγει την Προσθήκη / Αφαίρεση Έργων. • Ο Υπεύθυνος Λειτουργικών Διαδικασιών ή ο Προγραμματιστής επιλέγει το έργο που θα προστεθεί / αφαιρεθεί. • Ο Υπεύθυνος Λειτουργικών Διαδικασιών ή ο Προγραμματιστής επιλέγει την Προσθήκη / Αφαίρεση Αρχείων.
Μετασυνθήκες	Να προστεθεί / αφαιρεθεί το έργο στην λίστα παρακολούθησης αν όλες οι ενέργειες έγιναν σωστά.
Εναλλακτικές Ροές	-

Ονομασία Περίπτωσης Χρήσης	Έλεγχος του Κώδικα για αδυναμίες που μπορεί να οδηγήσουν σε ρήγματα ασφαλείας
Κωδικός	ΠΧ03
Σύντομη Περιγραφή	Ελέγχεται ο κώδικας για αδυναμίες.
Βασικοί Δράστες	Στοιχείο Πλατφόρμας TRACER
Δευτερεύοντες Δράστες	-
Σημαντικότητα	Υψηλή
Προϋποθέσεις	-
Κύρια Ροή	Το Στοιχείο Πλατφόρμας TRACER ελέγχει τον κώδικα για αδυναμίες.
Μετασυνθήκες	Ο κώδικας είναι ασφαλής ή δεν είναι ασφαλής και καταγράφεται το πρόβλημα που εντοπίστηκε.
Εναλλακτικές Ροές	-

Ονομασία Περίπτωσης Χρήσης	Καταγραφή Προβλημάτων Ασφάλειας

Κωδικός	ΠΧ04
Σύντομη Περιγραφή	Καταγράφονται τα προβλήματα ασφαλείας που εντοπίστηκαν κατά τον έλεγχο του κώδικα.
Βασικοί Δράστες	Στοιχείο Πλατφόρμας TRACER
Δευτερεύοντες Δράστες	-
Σημαντικότητα	Υψηλή
Προϋποθέσεις	• Να έχει γίνει έλεγχος ενός αρχείου για αδυναμίες (ΠΧ03). • Να έχει βρεθεί ένα κενό ασφαλείας.
Κύρια Ροή	1. Το Στοιχείο Πλατφόρμας TRACER καταγράφει το ή τα προβλήματα που εντοπίστηκαν. 2. Το Στοιχείο Πλατφόρμας TRACER εμφανίζει μήνυμα εύρεσης λάθους που περιέχει πληροφορίες σχετικά με τον τύπου του λάθους, το σημείο στο οποίο βρέθηκε και την σημαντικότητα του.
Μετασυνθήκες	Ο Προγραμματιστής επιλέγει να διορθώσει τα προβλήματα που καταγράφηκαν ή τα αγνοεί.
Εναλλακτικές Ροές	-

Ονομασία Περίπτωσης Χρήσης	Θωράκιση με security libraries
Κωδικός	ΠΧ05
Σύντομη Περιγραφή	Αν εντοπιστούν αδυναμίες ασφάλειας η πλατφόρμα προτείνει τη χρήση βιβλιοθηκών που παρέχουν αντίμετρα για τις συγκεκριμένες αδυναμίες.
Βασικοί Δράστες	Προγραμματιστής, Στοιχείο Πλατφόρμας TRACER
Δευτερεύοντες Δράστες	-
Σημαντικότητα	Μεσαία
Προϋποθέσεις	1. Να έχουν εντοπιστεί στον πηγαίο κώδικα της εφαρμογής αδυναμίες ασφάλειας. 2. Να έχει γίνει καταγραφή και συσχέτιση βιβλιοθηκών ασφάλειας με τις αδυναμίες από τις οποίες παρέχουν προστασία.
Κύρια Ροή	Το Στοιχείο Πλατφόρμας TRACER προτείνει τη χρήση συγκεκριμένων βιβλιοθηκών ασφάλειας ανάλογα με τις αδυναμίες που έχουν εντοπιστεί στον κώδικα της εφαρμογής
Μετασυνθήκες	Ο Προγραμματιστής ενσωματώνει στον κώδικα της εφαρμογής αναφορές και κατάλληλες κλήσεις για την χρήση των βιβλιοθηκών ασφάλειας που προτείνει η πλατφόρμα.
Εναλλακτικές Ροές	-

Ονομασία Περίπτωσης Χρήσης	Δημιουργία λειτουργικών αρθρωμάτων (Plugins)
Κωδικός	ΠΧ06
Σύντομη Περιγραφή	Δημιουργείται κάποιο λειτουργικό άρθρωμα (Plugin) βασισμένο σε πρότυπο κώδικα που παρέχεται από την πλατφόρμα TRACER, με σκοπό την πραγματοποίηση ανίχνευσης νέων ειδών τρωτοτήτων ή την ενσωμάτωση εξωτερικών εργαλείων.
Βασικοί Δράστες	Προγραμματιστής
Δευτερεύοντες Δράστες	-
Σημαντικότητα	Μεσαία
Προϋποθέσεις	• Η πλατφόρμα TRACER παρέχει πρότυπο κώδικα για τη δημιουργία ενός λειτουργικού αρθρώματος (Plugin)
Κύρια Ροή	1. Ο Προγραμματιστής δημιουργεί ένα αντίγραφο του προτύπου που παρέχει η πλατφόρμα TRACER για την δημιουργία ενός λειτουργικού αρθρώματος. 2. Ο Προγραμματιστής ενσωματώνει νέο κώδικα στο αντίγραφο του προτύπου, ώστε χρησιμοποιώντας τις κατάλληλες προγραμματιστικές

	διεπαφές που παρέχει η πλατφόρμα TRACER να δημιουργήσει ένα λειτουργικό άρθρωμα που προσθέτει νέα λειτουργικότητα στο σύστημα. 3. Ο Προγραμματιστής μεταγλωττίζει το νέο λειτουργικό άρθρωμα ώστε αυτό να μπορεί να χρησιμοποιηθεί από την πλατφόρμα TRACER.
Μετασυνθήκες	Δημιουργείται ένα νέο λειτουργικό άρθρωμα το οποίο επεκτείνει τη λειτουργικότητα της πλατφόρμας TRACER
Εναλλακτικές Ροές	-

Ονομασία Περίπτωσης Χρήσης	Προσθήκη λειτουργικών αρθρωμάτων (Plugins) στην πλατφόρμα TRACER
Κωδικός	ΠΧ07
Σύντομη Περιγραφή	Προστίθεται ένα νέο λειτουργικό άρθρωμα στην πλατφόρμα TRACER.
Βασικοί Δράστες	Διαχειριστής της πλατφόρμας TRACER
Δευτερεύοντες Δράστες	-
Σημαντικότητα	Μεσαία
Προϋποθέσεις	-
Κύρια Ροή	1. Ο Διαχειριστής της πλατφόρμας TRACER επιλέγει την καρτέλα Λειτουργικά Αρθρώματα στην Παραμετροποίηση του συστήματος. 2. Ο Διαχειριστής της πλατφόρμας TRACER επιλέγει ένα ή περισσότερα αρχεία τα οποία απαρτίζουν το λειτουργικό άρθρωμα. 3. Ο Διαχειριστής της πλατφόρμας TRACER επιλέγει την ενέργεια Προσθήκη
Μετασυνθήκες	Προστίθεται ένα νέο λειτουργικό άρθρωμα στην πλατφόρμα TRACER
Εναλλακτικές Ροές	1. Ο Διαχειριστής της πλατφόρμας TRACER να επιλέξει ένα αρχείο που δεν υπάρχει στο σύστημα ή δεν περιέχει κάποιο λειτουργικό άρθρωμα και το σύστημα να εμφανίσει μήνυμα λάθους. 2. Ο Διαχειριστής της πλατφόρμας TRACER να μην επιλέξει κανένα αρχείο, να επιλέξει Προσθήκη και το σύστημα να εμφανίσει μήνυμα λάθους.

Ονομασία Περίπτωσης Χρήσης	Αφαίρεση λειτουργικών αρθρωμάτων (Plugins) από την πλατφόρμα TRACER
Κωδικός	ΠΧ08
Σύντομη Περιγραφή	Αφαιρείται ένα υπάρχον λειτουργικό άρθρωμα από την πλατφόρμα TRACER
Βασικοί Δράστες	Διαχειριστής της πλατφόρμας TRACER
Δευτερεύοντες Δράστες	-
Σημαντικότητα	Χαμηλή
Προϋποθέσεις	Να έχει προστεθεί κάποιο λειτουργικό άρθρωμα στην πλατφόρμα TRACER (ΠΧ08)
Κύρια Ροή	1. Ο Διαχειριστής της πλατφόρμας TRACER επιλέγει την καρτέλα Λειτουργικά Αρθρώματα στην Παραμετροποίηση του συστήματος. 2. Ο Διαχειριστής της πλατφόρμας TRACER επιλέγει ένα ή περισσότερα από τα λειτουργικά αρθρώματα που έχουν προστεθεί στην πλατφόρμα TRACER. 3. Ο Διαχειριστής της πλατφόρμας TRACER επιλέγει την ενέργεια Αφαίρεση.
Μετασυνθήκες	Αφαιρείται ένα ή περισσότερα από τα λειτουργικά αρθρώματα που έχουν προστεθεί στην πλατφόρμα TRACER
Εναλλακτικές Ροές	2α. Ο Διαχειριστής της πλατφόρμας TRACER να μην επιλέξει κανένα λειτουργικό άρθρωμα, να επιλέξει Αφαίρεση και το σύστημα να εμφανίσει μήνυμα λάθους.

Ονομασία Περίπτωσης Χρήσης	Ενεργοποίηση λειτουργικών αρθρωμάτων (Plugins)
Κωδικός	ΠΧ09
Σύντομη Περιγραφή	Καθορίζονται κανόνες με βάση τους οποίους θα ενεργοποιείται η εκτέλεση ενός λειτουργικού αρθρώματος.
Βασικοί Δράστες	Διαχειριστής της πλατφόρμας TRACER
Δευτερεύοντες Δράστες	-
Σημαντικότητα	Υψηλή
Προϋποθέσεις	1. Να έχουν προστεθεί λειτουργικά αρθρώματα (Plugins) στην πλατφόρμα TRACER (ΠΧ07) 2. Ή να έχουν δημιουργηθεί λειτουργικά αρθρώματα (Plugins) (ΠΧ06)
Κύρια Ροή	1. Ο Προγραμματιστής ή ο Διαχειριστής της πλατφόρμας TRACER επιλέγει την καρτέλα Λειτουργικά Αρθρώματα στην Παραμετροποίηση του συστήματος. 2. Ο Προγραμματιστής ή ο Διαχειριστής της πλατφόρμας επιλέγει την Ενεργοποίηση Λειτουργικών Αρθρωμάτων. 3. Ο Προγραμματιστής ή ο Διαχειριστής της πλατφόρμας επιλέγει ένα ή περισσότερα από τα λειτουργικά αρθρώματα που έχουν προστεθεί στην πλατφόρμα TRACER για ενεργοποίηση. 4. Ο Προγραμματιστής ή ο Διαχειριστής της πλατφόρμας επιλέγει την Ενεργοποίηση Επιλεγμένων.
Μετασυνθήκες	Ενεργοποιούνται τα λειτουργικά αρθρώματα που επέλεξε ο Προγραμματιστής ή ο Διαχειριστής της πλατφόρμας.
Εναλλακτικές Ροές	3α.Ο Προγραμματιστής ή ο Διαχειριστής της πλατφόρμας δεν επιλέγει κανένα λειτουργικό άρθρωμα και εμφανίζεται αντίστοιχο μήνυμα λάθους.

5 Ανάλυση Απαιτήσεων Πλατφόρμας TRACER

5.1 Περιγραφή των Λειτουργικών Απαιτήσεων

Οι λειτουργικές απαιτήσεις της πλατφόρμας TRACER που περιγράφονται στις ακόλουθες ενότητες, έχουν οργανωθεί με βάση τη λειτουργικότητα που θα παρέχει. Στις περιπτώσεις που μια λειτουργία μπορεί να αφορά περισσότερες από μια κατηγορίες χρηστών για τις οποίες υπάρχει διαφοροποίηση στην τρόπο εκτέλεσης, αυτές ορίζονται ανάλογα με την κατηγορία. Οι περιγραφές και οι απαιτήσεις για τις διεπαφές που θα δημιουργηθούν ή χρησιμοποιηθούν από την πλατφόρμα αναλύονται στην ενότητα 6.

5.1.1 Απαιτήσεις σχετικά με τους Χρηστών

Οι λειτουργικές απαιτήσεις της συγκεκριμένης κατηγορίας σχετίζονται με τη διαχείριση των δικαιωμάτων των χρηστών του συστήματος, τη δημιουργία και τη διαχείριση των ρόλων των χρηστών στο σύστημα, καθώς και με τη δημιουργία και τη διαχείριση των χρηστών του συστήματος.

5.1.1.1 Δημιουργία Χρηστών (Λειτουργική Απαίτηση 1.1)

Η λειτουργία 1.1 αφορά στη δημιουργία χρηστών, δηλαδή στην ένα προς ένα αντιστοίχηση φυσικών προσώπων με τους δυνατούς χειριστές του συστήματος. Κατά την εκτέλεση της συγκεκριμένης λειτουργίας, καταγράφεται στο σύστημα ένας εν δυνάμει χειριστής του (διατηρώντας στοιχεία όπως το ονοματεπώνυμο και το email του) με αρχικά κενό σύνολο ρόλων και κατά συνέπεια και δικαιωμάτων σε αυτό.

5.1.2 Απαιτήσεις σχετικά με τη Διαχείριση Απειλών

Οι λειτουργικές απαιτήσεις της συγκεκριμένης κατηγορίας σχετίζονται με τη διαχείριση των αρχείων και των προτύπων που θα ελέγχονται. Με τον όρο λίστα παρακολούθησης αναφερόμαστε σε μια λίστα από αρχεία τα οποία ελέγχονται από την πλατφόρμα TRACER. Αντίστοιχα, με τον όρο Προφίλ Ασφαλείας αναφερόμαστε σε μια λίστα από ελέγχους που εκτελούνται σε συγκεκριμένα αρχεία. Σκοπό της διαδικασίας αποτελεί η ομαδοποίηση στόχων ασφαλείας, με αποτέλεσμα την ευκολότερη διαχείριση της πλατφόρμας.

5.1.2.1 Προσθήκη / Αφαίρεση 'Εργων στη Λίστα Παρακολούθησης (Λειτουργική Απαίτηση 2.1)

Η λειτουργία 2.1 αφορά στην προσθήκη αρχείων σε μια ήδη δημιουργημένη λίστα παρακολούθησης. Κατά την εκτέλεση της συγκεκριμένης λειτουργίας, το σύνολο των αρχείων που υπάρχουν σε μία λίστα παρακολούθησης αυξάνεται ως προς ένα ή περισσότερα αρχεία.

5.1.3 Απαιτήσεις σχετικά με την Ανάλυση Τρωτοτήτων

Οι λειτουργικές απαιτήσεις της συγκεκριμένης κατηγορίας αφορούν στη διενέργεια και στο χειρισμό των αποτελεσμάτων των ελέγχων συγκεκριμένων αρχείων. Ο έλεγχος των αρχείων μπορεί να πραγματοποιηθεί είτε αυτοματοποιημένα και συνεχώς, είτε μετά από αίτηση ενός χρήστη.

5.1.3.1 Έλεγχος του Κώδικα για αδυναμίες που μπορεί να οδηγήσουν σε ρήγματα ασφάλειας (Λειτουργική Απαίτηση 3.1)

Η λειτουργία 3.3 αφορά στον έλεγχο του κώδικα ενός συγκεκριμένου αρχείου για αδυναμίες σε επίπεδο λογισμικού, που μπορεί να οδηγήσουν σε ρήγματα ασφαλείας τα οποία μπορεί να εκμεταλλευθεί ένας κακόβουλος χρήστης. Αποτέλεσμα της συγκεκριμένης διαδικασίας αποτελεί η ανίχνευση ή όχι της ύπαρξης των προτύπων που σχετίζονται με διάφορες ευπάθειες.

5.1.3.2 Καταγραφή Προβλημάτων Ασφάλειας (Λειτουργική Απαίτηση 3.2)

Η λειτουργία 3.2 αφορά στη καταγραφή των κενών ασφαλείας που έχουν εντοπιστεί μέσω της εκτέλεσης της λειτουργίας 3.3. Κατά την εκτέλεση της λειτουργίας το / τα προβλήματα καταγράφονται και αποστέλλονται μέσω προειδοποιητικού email στον υπεύθυνο ασφαλείας για το συγκεκριμένο αρχείο.

5.1.4 Απαιτήσεις για την επέκταση της πλατφόρμας με λειτουργικά αρθρώματα (Plug-ins)

Οι λειτουργικές απαιτήσεις της κατηγορίας αυτής αφορούν το πως η πλατφόρμα θα μπορεί να επεκτείνεται είτε προσθέτοντας εξωτερικά εργαλεία που θα ανιχνεύουν αδυναμίες που μπορούν να οδηγήσουν σε ρήγματα ασφάλειας, είτε νέα προγράμματα που θα έχουν γραφτεί από τους προγραμματιστές και θα έχουν σαν στόχο να ανιχνεύσουν συγκεκριμένες αδυναμίες σε σχέση με το έργο. Και στις δυο περιπτώσεις, η ενσωμάτωση θα γίνεται με την μορφή ενός plugin.

5.1.4.1 Δημιουργία υποδομής για την υποστήριξη λειτουργικών αρθρωμάτων (Plugins) (Λειτουργική Απαίτηση 4.1)

Ένα λειτουργικό άρθρωμα είναι ένα στοιχείο λογισμικού (component) το οποίο θα υλοποιεί κάποια διεπαφή την οποία γνωρίζει η πλατφόρμα. Η πλατφόρμα θα μπορεί επομένως να ενεργοποιήσει το Plugin προκειμένου αυτή να ενεργοποιηθεί και να μπορεί να εκτελεστεί αυτόνομα ή μέσω άλλων Plugins. Η πλατφόρμα θα πρέπει να ορίζει μια τέτοια προγραμματιστική διεπαφή (Application Programming Interface, API), που θα επιτρέπει βασικές λειτουργίες όπως η αυτόματη ενεργοποίηση κατά την πραγματοποίηση ενός αυτοματοποιημένου ή χειροκίνητου ελέγχου για τρωτότητες. Η διεπαφή αυτή θα πρέπει να τεκμηριωθεί με τη μορφή εξωτερικής τεκμηρίωσης που να επιτρέπει σε τρίτους να δημιουργούν δικά τους plugins τα οποία θα μπορούν να εκτελεστούν στην πλατφόρμα.

Περισσότερες λεπτομέρειες σχετικά με τη διεπαφή αυτή μπορούν να βρεθούν στην ενότητα Διεπαφές συστήματος 6.1.

5.1.4.2 Ενσωμάτωση εξωτερικών εργαλείων (Λειτουργική Απαίτηση 4.2)

Υπάρχει μια πληθώρα εργαλείων που αναλύουν στατικά κώδικα για να εντοπίσουν αδυναμίες [4]. Η ανάλυση μπορεί να γίνεται είτε σε επίπεδο πηγαίου κώδικα (source code) είτε σε επίπεδο μεταγλωττισμένου κώδικα (compiled code). Σκοπός μας είναι να μπορεί ένας προγραμματιστής να ενσωματώσει ένα τέτοιο εργαλείο στην πλατφόρμα μας εύκολα και αποδοτικά. Για να γίνει κάτι τέτοιο θα πρέπει να δημιουργηθεί κάποιο πρότυπο (template) το οποίο θα πρέπει να ακολουθεί ο προγραμματιστής ώστε να κάνει την ενσωμάτωση. Ακολουθώντας το πρότυπο αυτό, ο προγραμματιστής θα δημιουργεί ένα plugin το οποίο αρχικά θα καλεί το εργαλείο ενώ μόλις αυτό το ολοκληρώσει τον έλεγχο του θα αναλύει τα αποτελέσματά του επιτρέποντας στον προγραμματιστή να τα απεικονίσει με όποιον τρόπο επιθυμεί.

5.1.4.3 Ενσωμάτωση προσαρμοσμένων προγραμμάτων (Λειτουργική Απαίτηση 4.3)

Υπάρχουν περιπτώσεις όπου ένας προγραμματιστής θα θέλει να διαπιστώσει συγκεκριμένα πράγματα σε σχέση με την ασφάλεια της εφαρμογής του. Η πλατφόρμα TRACER θα του παρέχει την ευκαιρία να δημιουργεί προσαρμοσμένα (custom) plugins που θα κάνουν έλεγχους προσαρμοσμένους στην φύση της εφαρμογής που αναπτύχθηκε (π.χ. Την γλώσσα προγραμματισμού που χρησιμοποιήθηκε, εάν είναι web εφαρμογή κ.α.). Αυτό θα γίνεται πάλι με πρότυπα όπως αναφέρθηκε και στην προηγούμενη περίπτωση.

5.1.5 Απαιτήσεις σχετικά με την ενίσχυση της ασφαλείας της εφαρμογής

Εκτός από την ανεύρεση αδυναμιών, ένας άλλος βασικός στόχος της πλατφόρμας TRACER είναι να προτείνει αντίμετρα που να μπορούν να χρησιμοποιήσουν οι προγραμματιστές της εφαρμογής ώστε να την θωρακίσουν. Οι προτάσεις αυτές θα εξαρτώνται άμεσα από τα αποτελέσματα της στατικής ανάλυση που θα έχει προηγηθεί.

5.1.5.1 Θωράκιση μέσω βιβλιοθηκών ασφαλείας (Λειτουργική Απαίτηση 5.1)

Ανάλογα με τα αποτελέσματα του στατικού ελέγχου η πλατφόρμα TRACER θα προτείνει για την προστασία της εφαρμογής την χρησιμοποίηση βιβλιοθηκών ασφάλειας. Συγκεκριμένα, τα αποτελέσματα του στατικού ελέγχου θα καταγράφονται. Έπειτα, η πλατφόρμα θα τα ελέγχει και θα προτείνει βιβλιοθήκες που θα μπορούν να καταπολεμήσουν τις καταγεγραμμένες αδυναμίες. Ο όρος "βιβλιοθήκη ασφάλειας" μπορεί να περιλαμβάνει είτε υπάρχουσες βιβλιοθήκες, είτε νέες που μπορούν να αναπτυχθούν για να καλύψουν συγκεκριμένες ανάγκες.

5.2 Περιγραφή Μη-λειτουργικών Απαιτήσεων

5.2.1 Φυσικές απαιτήσεις

Ο υλικός εξοπλισμός που θα χρησιμοποιηθεί για την εκτέλεση της πλατφόρμας TRACER είναι επιθυμητό να βρίσκεται σε φυσική τοποθεσία με τα ακόλουθα χαρακτηριστικά:

- Περιορισμένη φυσική πρόσβαση μέσω χρήσης μηχανικής ή ηλεκτρονικής κλειδαριάς.
- Συστήματα κλιματισμού και εξαερισμού με ανεξάρτητη παροχή ισχύος, προκειμένου ο εξοπλισμός να λειτουργεί εντός των προδιαγραφών θερμοκρασίας και υγρασίας που προτείνεται από τους εκάστοτε κατασκευαστές.
- Συστήματα αυτόματης ανίχνευσης, σήμανσης συναγερμού και κατάσβεσης πυρκαγιάς. Ο συναγερμός είναι επιθυμητό να διαθέτει και δυνατότητα χειροκίνητης σήμανσης, ενώ το σύστημα πυρόσβεσης θα πρέπει να είναι κατάλληλος για πυρκαγιές που προκαλούνται από ηλεκτρικό εξοπλισμό. Η ανίχνευση είναι επιθυμητό να πραγματοποιείται με τη χρήση αισθητήρων θερμότητας, καπνού ή οπτικούς, ή συνδυασμούς αυτών.
- Ψευδοροφή ή/και υπερυψωμένο πάτωμα, προκειμένου να διευκολύνεται ο κλιματισμός, η οργάνωση της καλωδίωσης και η προστασία από πλημμύρες.

- Συστήματα αδιάλειπτης παροχής ισχύος (Uninterruptible Power Supply, UPS) των οποίων η ικανότητα παροχής ισχύος πρέπει να υπερβαίνει τη μέγιστη κατανάλωση του εξοπλισμού κατά 20% κατ' ελάχιστον. Τα συστήματα αυτά θα πρέπει να διαθέτουν κυκλώματα προστασίας από υπερτάσεις.
- Καλωδίωση δικτύου συμβατή με τα πρότυπα ANSI/TIA/EIA-568-B.1-2001 και ANSI/TIA/EIA-568-B.2-2001, EN50173 ή ISO/IEC 11801. Η υποδομή καλωδίωσης θα πρέπει να είναι συμβατή με την Ευρωπαϊκή Οδηγία για την Ηλεκτρομαγνητική Συμβατότητα 2004/108/EEC και να υποστηρίζει μετάδοση δεδομένων με ρυθμούς ως 1000Mbps/s στα 100m (1000BaseT).
- Όλος ο ενεργός ηλεκτρονικός εξοπλισμός (εξυπηρετητές, ενεργός δικτυακός εξοπλισμός, κλπ) θα πρέπει να είναι εγκατεστημένος σε standard ερμάρια (racks) πλάτους 19 ιντσών, ελάχιστου βάθους 60 εκατοστών και ελάχιστου ύψους τέτοιου που να υπερβαίνει το ύψος του εγκατεστημένου εξοπλισμού κατά 20%. Τα racks θα πρέπει να επιτρέπουν διευθέτηση της καλωδίωσης, να διαθέτουν αφαιρούμενα πλαϊνά και κλειδαριά στο εμπρόσθιο τμήμα.

5.2.2 Περιβάλλον λειτουργίας και προσαρμοστικότητα

Σε επίπεδο υποδομών που χρησιμοποιούνται για την λειτουργία της πλατφόρμας, είναι σημαντικό αυτές να μπορούν να ενισχυθούν / επεκταθούν προκειμένου να είναι διαθέσιμη η απαραίτητη επεξεργαστική ισχύς και η συνδεσιμότητα μέσω δικτύου που απαιτείται για την ομαλή λειτουργία της πλατφόρμας εντός των απαιτήσεων που ορίζονται για την απόδοσή της. Αυτό σημαίνει πως πρέπει να είναι δυνατή η εγκατάσταση επιπλέον εξοπλισμού επεξεργασίας, συνδεσιμότητας (δικτύων) και παροχής ενέργειας αντίστοιχα προκειμένου να μπορούν να εξυπηρετηθούν μελλοντικές ανάγκες.

5.2.3 Απαιτήσεις απόδοσης συστήματος

Η αξιολόγηση της απόδοσης ενός συστήματος εντοπισμού προβλημάτων ασφάλειας σε κώδικα μπορεί να στηριχτεί σε δύο εναλλακτικές προσεγγίσεις:

1. Αξιολόγηση ως προς μία κατασκευασμένη σουίτα δοκιμαστικών προγραμμάτων που σκοπό έχει να καλύψει μεγάλο φάσμα προβλημάτων ασφάλειας σε όλες τις πιθανές εκδοχές [3].
2. Αξιολόγηση ως προς την απόδοση του συστήματος σε λογισμικό και benchmarks με ήδη καταγεγραμμένα προβλήματα ασφαλείας. Στην περίπτωση αυτή ελέγχεται η επάρκεια αποκλειστικά και μόνο στα προβλήματα που εμφανίζει το λογισμικό που χρησιμοποιείται για το σκοπό αυτό. Αν δεν καταστεί δυνατή η διάθεση τέτοιου λογισμικού από το χρήστη, τότε θα αναζητηθούν εφαρμογές web ανοικτού λογισμικού.

Η πλατφόρμα TRACER θα χρησιμοποιείται για την ανάλυση σημαντικής ποσότητας δεδομένων (πηγαίου κώδικα εφαρμογών) από διάφορα συστήματα λογισμικού, όπως περιγράφεται στην ενότητα 4.3. Καθώς πολλοί χρήστες θα χρησιμοποιούν την πλατφόρμα ταυτόχρονα, είναι σημαντικό να επιτευχθεί ένα σύνολο στόχων απόδοσης που θα επιτρέπει την απροβλημάτιστη εξυπηρέτηση των αναγκών των χρηστών αυτών. Αυτές οι απαιτήσεις περιγράφονται στις παραγράφους που ακολουθούν.

5.2.3.1 Απόδοση λειτουργιών δημιουργίας χρηστών

Περιγραφή	Οι λειτουργίες που αφορούν σε δημιουργία χρηστών πρέπει να ολοκληρώνονται σε χρονικό διάστημα μικρότερο του ενός λεπτού στο 95% των περιπτώσεων.
Σχετικές Π.Χ.	ΠΧ01
Σχόλια	Οι ενέργειες που αφορούν στη δημιουργία χρηστών μετά την αρχική παραμετροποίηση της πλατφόρμας δεν θα λαμβάνουν χώρα με ιδιαίτερη συχνότητα, οπότε η απόδοσή τους δεν είναι κρίσιμη. Ωστόσο, η πραγματοποίησή τους δεν πρέπει να είναι ιδιαίτερα χρονοβόρα προκειμένου να εξασφαλισθεί η καλή εμπειρία του χρήστη (στη συγκεκριμένη περίπτωση του διαχειριστή της πλατφόρμας). Το χρονικό περιθώριο δεν αφορά στο χρόνο που απαιτείται για την εισαγωγή δεδομένων αλλά για την πραγματοποίηση της λειτουργίας.

5.2.3.2 Απόδοση λειτουργιών διαχείρισης Έργων

Περιγραφή	Οι λειτουργίες που αφορούν σε διαχείριση έργων πρέπει να ολοκληρώνονται σε χρονικό διάστημα μικρότερο του ενός λεπτού στο 95% των περιπτώσεων.
Σχετικές Π.Χ.	ΠΧ02
Σχόλια	Οι ενέργειες που αφορούν στη διαχείριση έργων θα λαμβάνουν χώρα με ιδιαίτερη συχνότητα κατά τη διάρκεια της λειτουργίας του συστήματος, οπότε η απόδοσή τους είναι κρίσιμη. Η ολοκλήρωσή τους δεν πρέπει να είναι ιδιαίτερα χρονοβόρα προκειμένου να εξασφαλισθεί η καλή εμπειρία του χρήστη. Το χρονικό περιθώριο δεν αφορά στο χρόνο που απαιτείται για την εισαγωγή δεδομένων αλλά για την πραγματοποίηση της λειτουργίας.

5.2.3.3 Απόδοση λειτουργιών ελέγχου Έργων

Απόδοση Ελέγχου του Κώδικα για αδυναμίες που μπορεί να οδηγήσουν σε ρήγματα ασφάλειας

Περιγραφή	Η πλατφόρμα TRACER πρέπει να εκτελεί τον έλεγχο ενός αρχείου για κάθε κατηγορία προβλημάτων ασφάλειας σε λιγότερο από 1 λεπτό ανά τύπο στο 95% των περιπτώσεων. Σε περίπτωση που χρειάζεται να εξεταστούν αυτόματα περισσότερα αρχεία λόγω εξαρτήσεων μεταξύ τους ο συνολικός χρόνος για την ανάλυση δεν πρέπει να ξεπερνά το 1 λεπτό ανά αρχείο αντίστοιχα.
Σχετικές Π.Χ.	ΠΧ03
Σχόλια	Προκειμένου να αποφευχθεί η παρείσφρηση τρωτοτήτων είναι σημαντικό η ανάλυση ενός αρχείου να πραγματοποιείται σε σύντομο χρονικό διάστημα, έτσι ώστε να επιτευχθεί καλή εμπειρία χρήστη. Δεδομένου ότι η πλατφόρμα TRACER τυπικά θα εξετάζει μεγάλο αριθμό αρχείων, η καλή απόδοση των ελέγχων για τους προαναφερθέντες τύπους τρωτοτήτων θα εξασφαλίσει την καλή απόδοση της πλατφόρμας συνολικά. Το χρονικό περιθώριο που ορίζεται αφορά στον έλεγχο για όλες τις γνωστές πιθανές τρωτότητες που εμπίπτουν στην κάθε κατηγορία αντίστοιχα.

Απόδοση Καταγραφής Προβλημάτων Ασφάλειας

Περιγραφή	Η πλατφόρμα TRACER πρέπει να εκτελεί την καταγραφή των προβλημάτων ασφάλειας που έχουν εντοπιστεί σε ένα σύστημα λογισμικού εντός ενός λεπτού στο 95% των περιπτώσεων.
Σχετικές Π.Χ.	ΠΧ04
Σχόλια	Δεδομένου ότι η καταγραφή των τρωτοτήτων είναι μια από τις πλέον κρίσιμες λειτουργίες της πλατφόρμας, είναι σημαντικό να χαρακτηρίζεται από καλή απόδοση. Το χρονικό περιθώριο που ορίζεται κρίνεται επαρκές για να λάβει ο χρήστης της πλατφόρμας έγκαιρη ενημέρωση σχετικά με τρωτότητες που έχουν ανιχνευτεί.

5.2.3.4 Απόδοση αποθήκευσης και ανάκτησης πληροφοριών

Περιγραφή	Ο χρόνος που απαιτείται για την αποθήκευση και ανάκτηση ενός διακριτού στοιχείου πληροφορίας σχετικά με τη λειτουργία και την παραμετροποίηση της πλατφόρμας TRACER (πχ ρόλος, χρήστης, αρχείο προς παρακολούθηση, κλπ.) πρέπει να εκτελείται εντός ενός δευτερολέπτου στο 95% των περιπτώσεων, ανεξάρτητα από την υποκείμενη αναπαράσταση και μορφή αποθήκευσης των δεδομένων που χρησιμοποιείται.
Σχετικές Π.Χ.	όλες
Σχόλια	Δεδομένου ότι η πλατφόρμα θα καταγράφει πληροφορίες για ένα μεγάλο αριθμό οντοτήτων (χρηστών, ρόλων, προφίλ ασφάλειας κλπ) και αρχείων υπό παρακολούθηση, η απόδοση του μηχανισμού αποθήκευσης και ανάκτησης των δεδομένων αυτών κρίνεται ως ιδιαίτερα σημαντική για την καλή απόδοση της

	πλατφόρμας συνολικά. Η απόδοση πρέπει να παραμείνει μέσα στο παραπάνω όριο ανεξαρτήτως του αριθμού των οντοτήτων που διαχειρίζεται η πλατφόρμα.

5.2.3.5 Απόδοση διεπαφής χρήστη για τοπικές ενέργειες

Περιγραφή	Ο χρόνος που απαιτείται για την πραγματοποίηση ενεργειών μέσω της διεπαφής χρήστη που περιγράφεται στην ενότητα XX, οι οποίες δεν απαιτούν ανάκτηση στοιχείων ή μεταφορά δεδομένων από/προς τον εξυπηρετητή στον οποίο εκτελείται η πλατφόρμα TRACER πρέπει να πραγματοποιούνται εντός ενός δευτερολέπτου στο 99% των περιπτώσεων.
Σχετικές Π.Χ.	όλες
Σχόλια	Η πλατφόρμα TRACER θα προσφέρει κάποιου είδους γραφική διεπαφή (GUI) που θα επιτρέπει στους χρήστες να αλληλεπιδρούν μαζί της για την πραγματοποίηση των εργασιών διαχείρισης ρόλων, προφίλ ασφάλειας, αρχείων παρακολούθησης κλπ. Η πραγματοποίηση των ενεργειών αυτών απαιτεί την εκτέλεση μιας σειράς τοπικών ενεργειών, για τις οποίες δεν είναι απαραίτητη η επικοινωνία μέσω δικτύου με την πλατφόρμα. Προκειμένου να επιτευχθεί υψηλή χρηστικότητα της διεπαφής, είναι κρίσιμο οι ενέργειες αυτές να πραγματοποιούνται σε σύντομο χρονικό διάστημα. Σύμφωνα με μελέτες σχετικά με την Αλληλεπίδραση Ανθρώπου-Υπολογιστή, προκειμένου μια διεπαφή χρήστη να προσφέρει αίσθηση καλής ταχύτητας ανταπόκρισης πρέπει να εκτελεί τις ενέργειες ή να προσφέρει απαντήσεις (feedback) εντός ενός δευτερολέπτου.

5.2.3.6 Απόδοση διεπαφής χρήστη για απομακρυσμένες ενέργειες

Περιγραφή	Ο χρόνος που απαιτείται για την πραγματοποίηση ενεργειών μέσω της διεπαφής χρήστη που περιγράφεται στην ενότητα XX, οι οποίες απαιτούν ανάκτηση στοιχείων ή μεταφορά δεδομένων από/προς τον εξυπηρετητή στον οποίο εκτελείται η πλατφόρμα TRACER πρέπει να πραγματοποιούνται εντός 30 δευτερολέπτων στο 99% των περιπτώσεων.
Σχετικές Π.Χ.	όλες
Σχόλια	Η πλατφόρμα TRACER θα προσφέρει κάποιου είδους γραφική διεπαφή (GUI) που θα επιτρέπει στους χρήστες να αλληλεπιδρούν μαζί της για την πραγματοποίηση των εργασιών διαχείρισης ρόλων, προφίλ ασφάλειας, αρχείων παρακολούθησης κλπ. Η πραγματοποίηση των ενεργειών αυτών απαιτεί την εκτέλεση μιας σειράς τοπικών ενεργειών, για τις οποίες δεν είναι απαραίτητη η επικοινωνία μέσω δικτύου με την πλατφόρμα, και μια σειρά από ενέργειες οι οποίες προκαλούν την εκτέλεση ενεργειών στον εξυπηρετητή της πλατφόρμας. Τα μέγιστα επιτρεπτά χρονικά διαστήματα για τα οποία μπορεί να περιμένει ο χρήστης προτού μια ενέργεια θεωρηθεί ότι απέτυχε τυπικά καθορίζονται από το πρωτόκολλο επικοινωνίας που χρησιμοποιείται ανάμεσα στη διεπαφή χρήστη και τον εξυπηρετητή της πλατφόρμας. Τα 30 δευτερόλεπτα είναι μια τυπική τιμή χρονικού περιθωρίου σε αρκετά πρωτόκολλα.

5.2.3.7 Ικανότητα Ταυτόχρονης Χρήσης Πλατφόρμας

Περιγραφή	Η πλατφόρμα πρέπει να μπορεί να χρησιμοποιηθεί ταυτόχρονα μέσω διαφορετικών στιγμιότυπων των παρεχόμενων διεπαφών χρήστη από τουλάχιστον 5 χρήστες προτού ο χρόνος απόκρισης της πλατφόρμας διπλασιαστεί, υποθέτοντας ότι κάθε χρήστης εκτελεί μια ενέργεια ανά 30 δευτερόλεπτα κατά μέσο όρο.
Σχετικές Π.Χ.	όλες
Σχόλια	Η πλατφόρμα TRACER θα χρησιμοποιείται ταυτόχρονα από χρήστες κάθε κατηγορίας. Δεδομένου ότι οι χρήστες αυτοί θα εκτελούν διαφορετικές εργασίες ανά πάσα στιγμή, η πλατφόρμα πρέπει να μπορεί να εξυπηρετήσει τις ενέργειες που απαιτούνται από όλους τους χρήστες σε εύλογο χρονικό διάστημα.

5.2.4 Ορθότητα και ακρίβεια αποτελεσμάτων

Οι δείκτες αξιολόγησης της επάρκειας στην ανίχνευση προβλημάτων ασφάλειας σε κώδικα είναι γνωστοί από τη σχετική βιβλιογραφία [3]. Σε ότι αφορά την αποτελεσματικότητα της ανάλυσης (λάθος θετικά/false positives και λάθος αρνητικά/false negatives) αυτά είναι συνάρτηση της ευαισθησίας (sensitivity) της ανάλυσης εντοπισμού των προβλημάτων ασφάλειας. Ενδεικτικές μετρικές της αποτελεσματικότητας [3] είναι οι:

- αξιοπιστία (accuracy)
- ακρίβεια (precision)
- ανάκληση ή αναλογία σωστών θετικών (recall or true positive rate)
- προσδιοριστικότητα ή αναλογία σωστών αρνητικών (specificity or true negative rate)
- μετρική F

5.2.4.1 Αποδεκτά κατώφλια για εσφαλμένες θετικές / αρνητικές αποκρίσεις

Η πλατφόρμα TRACER θα αξιοποιεί εξωτερικές εφαρμογές για την ανάλυση του πηγαίου κώδικα και την ανίχνευση τρωτοτήτων. Η ακρίβεια και ο βαθμός ανάκλησης των τρωτοτήτων που ανιχνεύονται (precision & recall) εξαρτάται από τις δυνατότητες που επιδεικνύουν οι εξωτερικές αυτές εφαρμογές. Επομένως δε μπορούν να τεθούν αυστηρά κατώφλια αποδεκτών ποσοστών για εσφαλμένες θετικές / αρνητικές αποκρίσεις (false positives / false negatives).

5.2.5 Ασφάλεια συστήματος

Η πλατφόρμα TRACER δεν χειρίζεται προσωπικά δεδομένα ή ευαίσθητες πληροφορίες που να απαιτούν προστασία της εμπιστευτικότητας, ειδικά δεδομένου ότι θα χρησιμοποιείται στο πλαίσιο της διαδικασίας ανάπτυξης λογισμικού σε έναν οργανισμό. Επομένως, στο πλαίσιο αυτό είναι πιο σημαντικές άλλες όψεις της ασφάλειας, όπως η ακεραιότητα και η διαθεσιμότητα.

5.2.5.1 Διαθεσιμότητα συστήματος

Περιγραφή	Η πλατφόρμα TRACER πρέπει να είναι διαθέσιμη κατά το 95% του χρόνου σε περίοδο ενός έτους, και κατ' ελάχιστο 75% του χρόνου κατά τις εργάσιμες μέρες και ώρες.
Σχετικές Π.Χ.	Όλες
Σχόλια	Στο πλαίσιο των πληροφοριακών και τηλεπικοινωνιακών συστημάτων η διαθεσιμότητα αναφέρεται στο βαθμό κατά τον οποίο ένα σύστημα είναι λειτουργικό και μπορεί να παρέχει υπηρεσίες σε μια τυχαία χρονικά στιγμή. Με άλλα λόγια, είναι το ποσοστό του χρόνου κατά τον οποίο το σύστημα είναι λειτουργικό σε μια δεδομένη χρονική περίοδο. Δεδομένου ότι η μη λειτουργία της πλατφόρμας TRACER δεν παρεμποδίζει όλη τη διαδικασία της ανάπτυξης ενός συστήματος λογισμικού αλλά μόνο τις φάσεις ελέγχου του, οι απαιτήσεις διαθεσιμότητας της πλατφόρμας TRACER είναι αρκετά ελαστικές.

5.2.5.2 Αξιοπιστία συστήματος

Με τον όρο αξιοπιστία συστήματος μπορεί να αναφερθεί κανείς στην πιθανότητα που έχει το σύστημα να είναι σε πλήρως λειτουργική κατάσταση σε μια δεδομένη στιγμή. Με λίγα λόγια ορίζεται με βάση τη διαθεσιμότητα, στην οποία έγινε αναφορά στην προηγούμενη παράγραφο. Η αξιοπιστία ενός συστήματος λοιπόν μειώνεται όταν προκύπτουν σφάλματα. Προκειμένου να οριστούν απαιτήσεις για την αξιοπιστία της πλατφόρμας TRACER είναι απαραίτητο να γίνει πρώτα μια κατηγοριοποίηση των σφαλμάτων που μπορεί να προκύψουν κατά τη λειτουργία της. Η κατηγοριοποίηση αυτή περιλαμβάνει τα παρακάτω είδη σφαλμάτων:

- Κρίσιμα σφάλματα: Σφάλματα κατά τα οποία το σύστημα δεν είναι λειτουργικό και δε μπορεί να επανέλθει αυτόματα σε κατάσταση πλήρους λειτουργίας χωρίς ανθρώπινη επέμβαση. Παραδείγματα τέτοιων αποτυχιών στην περίπτωση της πλατφόρμας TRACER είναι και τα ακόλουθα:

1. Φυσική καταστροφή του ηλεκτρονικού και υπολογιστικού εξοπλισμού που χρησιμοποιείται για τη λειτουργία της πλατφόρμας, λόγω φυσικής καταστροφής, πυρκαγιάς, μη εξουσιοδοτημένης φυσικής πρόσβασης και καταστροφής εξοπλισμού.
2. Καταστροφή / πλήρης αποτυχία της υποδομής στην οποία αποθηκεύονται τα στοιχεία της παραμετροποίησης του συστήματος

- Σημαντικά σφάλματα: Σφάλματα κατά τα οποία η πλατφόρμα δεν είναι διαθέσιμη για κάποιο περιορισμένο χρονικό διάστημα, αλλά από τα οποία μπορεί να ανακάμψει αυτόματα με επανεκκίνηση εξοπλισμού και λογισμικού. Χαρακτηριστικό παράδειγμα τέτοιου σφάλματος είναι η διακοπή της συνδεσιμότητας δικτύου με τον εξυπηρετητή στον οποίο εκτελείται η πλατφόρμα TRACER.
- Μη σημαντικά σφάλματα: Σφάλματα που μπορεί να οδηγήσουν στην εξαγωγή λανθασμένων αποτελεσμάτων σχετικά με την ανάλυση του λογισμικού για τρωτότητες, ή αποτυχία για απόκριση των διεπαφών χρήστη ή εκτέλεση των ζητούμενων ενεργειών εντός του πλαισίου απόδοσης που έχει αναφερθεί στην ενότητα 5.2.3.

Με βάση τα παραπάνω, οι απαιτήσεις αξιοπιστίας της πλατφόρμας TRACER ορίζονται ως εξής:

- Μέσος χρόνος μεταξύ σφαλμάτων (Mean Time Between Failures, MTBF): Προκειμένου να επιτευχθούν τα επιθυμητά επίπεδα διαθεσιμότητας, τα διαστήματα μεταξύ σφαλμάτων θα πρέπει να υπερβαίνουν:
  1. Για Κρίσιμα Σφάλματα: > 24 ημέρες
  2. Για Σημαντικά Σφάλματα: > 2 ημέρες
  3. Για Μη Σημαντικά Σφάλματα: > 20 λεπτά
- Μέσος χρόνος ανάκαμψης από σφάλματα (Mean Time To Repair, MTTR): Προκειμένου να επιτευχθούν τα επιθυμητά επίπεδα διαθεσιμότητας, τα χρονικά διαστήματα ανάκαμψης πρέπει να μην υπερβαίνουν:
  1. Για Κρίσιμα Σφάλματα: 8 ώρες (2 ημέρες σε περίπτωση φυσικής καταστροφής)
  2. Για Σημαντικά Σφάλματα: 2 ώρες
  3. Για Μη Σημαντικά Σφάλματα: Δεν υπάρχει περιορισμός

5.2.5.3 Ακεραιότητα συστήματος

Για την εξασφάλιση της ακεραιότητας της πλατφόρμας TRACER θα πρέπει να ληφθούν τα παρακάτω μέτρα:

- Έλεγχος εξουσιοδότησης: Ένας μηχανισμός ελέγχου εξουσιοδότησης θα πρέπει να υλοποιηθεί προκειμένου να υπάρχει έλεγχος των χρηστών που μπορούν να αλληλεπιδράσουν με την πλατφόρμα και των ενεργειών που μπορούν να πραγματοποιήσουν. Η εξουσιοδότηση των χρηστών θα σχετίζεται με τους ρόλους που αυτοί έχουν, όπως περιγράφονται στην ενότητα 4.1.
- Έλεγχος αυθεντικοποίησης: Ένας μηχανισμός αυθεντικοποίησης θα πρέπει να υλοποιηθεί προκειμένου οι χρήστες να αποκτήσουν πρόσβαση στις ενέργειες για τις οποίες είναι εξουσιοδοτημένοι. Η αυθεντικοποίηση δεν αφορά τις αυτοματοποιημένες διαδικασίες ελέγχου, αλλά μόνο αυτές στις οποίες υπάρχει απευθείας αλληλεπίδραση του χρήστη με κάποια διεπαφή της πλατφόρμας.
- Καταγραφή ενεργειών: Όλες οι ενέργειες που αφορούν παραμετροποίηση της πλατφόρμας TRACER (περιπτώσεις χρήσης 01-14) θα πρέπει να καταγράφονται αυτόματα σε κατάλληλο αρχείο ιστορικού συμβάντων. Οι ενέργειες που πραγματοποιούνται αυτόματα με τακτική περιοδικότητα θα πρέπει επίσης να καταγράφονται. Σε κάθε συμβάν που καταγράφεται θα πρέπει να υπάρχει χρονική σήμανση, και αν αυτό δεν προκύπτει από κάποια αυτοματοποιημένη διαδικασία αλλά από ενέργεια κάποιου χρήστη, να αναφέρεται και κάποιο αναγνωριστικό του χρήστη αυτού.
- Αντίγραφα ασφάλειας: Τα δεδομένα που αφορούν την παραμετροποίηση της πλατφόρμας (πχ στοιχεία χρηστών, προφίλ ασφάλειας, λίστες αρχείων προς παρακολούθηση, κλπ) θα πρέπει να υπόκεινται σε κάποια πολιτική τήρησης αντιγράφων ασφαλείας, η οποία θα πρέπει να διαθέτει τα παρακάτω χαρακτηριστικά:
  1. Θα τηρείται ένα πλήρες αντίγραφο ασφάλειας των δεδομένων της πλατφόρμας κάθε εβδομάδα, και κάθε ημέρα θα δημιουργούνται συμπληρωματικά διαφορικά (differential) αντίγραφα.

2. Πρέπει να υπάρχουν πλήρη αντίγραφα ασφαλείας που να καλύπτουν μια περίοδο τουλάχιστον δύο εβδομάδων.
3. Τα αντίγραφα ασφαλείας θα διατηρούνται σε μαγνητικά ή άλλα μέσα αποθήκευσης, τα οποία θα βρίσκονται σε απομακρυσμένη τοποθεσία.
4. Τα αντίγραφα ασφάλειας και τα μέσα αποθήκευσής τους θα πρέπει να ελέγχονται για σφάλματα τουλάχιστον 1 φορά κάθε μήνα.

5.2.6 Συντηρησιμότητα Συστήματος

Το λογισμικό που θα αναπτυχθεί για τη λειτουργία της πλατφόρμας TRACER είναι σημαντικό να χαρακτηρίζεται από εύκολη συντήρηση, δεδομένου ότι θα χρειαστεί να συντηρείται τακτικά καθώς προκύπτουν καινοτόμες τεχνικές ανίχνευσης τρωτοτήτων οι οποίες θα ήταν επιθυμητό να μπορούν να αξιοποιηθούν με μικρό κόπο, Επίσης είναι προφανές πως θα χρειαστεί συντήρηση με την έννοια των τροποποιήσεων προκειμένου να προσαρμοστεί στο λειτουργικό περιβάλλον της, όπως επίσης και των διορθώσεων που είναι απαραίτητες για την εξάλειψη προγραμματιστικών σφαλμάτων. Στο πλαίσιο λοιπόν του έργου, οι αναφορές στην συντηρησιμότητα αφορούν τα παρακάτω χαρακτηριστικά τα οποία αποτελούν τις διαφορετικές παραμέτρους της συντηρησιμότητας οποιουδήποτε συστήματος λογισμικού:

- Αναλυσιμότητα (analyzability): Αφορά στη δυσκολία ανίχνευσης του τμήματος του κώδικα το οποίο περιέχει κάποιο σφάλμα ή το οποίο πρέπει να τροποποιηθεί ώστε το σύστημα να παρέχει νέα λειτουργικότητα. Με άλλα λόγια, αφορά στην ευκολία με την οποία μπορεί να γίνει κατανοητή η αρχιτεκτονική του συστήματος.
- Εναλλαξιμότητα (changeability): Αφορά στον κόπο που απαιτείται για την πραγματοποίηση αλλαγών στο σύστημα.
- Σταθερότητα (stability): Αφορά στο βαθμό και την έκταση κατά τα οποία πρέπει να υποστεί τροποποιήσεις το σύστημα προκειμένου να διορθωθεί μια ατέλεια ή να πραγματοποιηθεί μια αλλαγή σε κάποιο σημείο του.
- Ελεγξιμότητα (testability): Αφορά στη δυνατότητα επαλήθευσης της σωστής λειτουργίας του συστήματος μετά την εφαρμογή τροποποιήσεων σε αυτό.

Οι απαιτήσεις που περιγράφονται στις επόμενες παραγράφους καλύπτουν τα παραπάνω χαρακτηριστικά, έτσι ώστε να επιτευχθεί ο επιθυμητός στόχος.

5.2.6.1 Συμβατότητα με προγραμματιστικά πρότυπα

Το λογισμικό που θα αναπτυχθεί για την πλατφόρμα TRACER θα πρέπει να τηρεί τις γενικά αποδεκτές συμβάσεις (coding standards) που αφορούν την γλώσσα προγραμματισμού στην οποία έχει αναπτυχθεί το κάθε τμήμα του αντίστοιχα. Αυτό μπορεί να επιτευχθεί με τη χρήση υπαρχόντων εργαλείων που εξετάζουν κατά πόσο ένα τμήμα κώδικα ακολουθεί τα πρότυπα αυτά.
Επίσης ενθαρρύνεται η χρήση έτοιμων βιβλιοθηκών κώδικα (code libraries) και πλαισίων εφαρμογών (application frameworks) ανοιχτού λογισμικού (open source) τα οποία απολαμβάνουν γενικής αποδοχής από την προγραμματιστική κοινότητα, και των οποίων η άδεια χρήσης είναι συμβατή με αυτή του λογισμικού που αναπτύσσεται.
Όσον αφορά την αναλυσιμότητα και την εναλλαξιμότητα, σημαντικό ρόλο παίζει η εύκολη κατανόηση του πηγαίου κώδικα, η οποία εξαρτάται κυρίως από το κατά πόσο το στυλ συγγραφής του είναι συνεπές. Η συνέπεια αφορά στην ομοιομορφία του προγραμματιστικού στυλ που χρησιμοποιείται για δηλώσεις μεταβλητών, μεθόδων, κλάσεων και εντολών. Το προγραμματιστικό στυλ επιβάλλει συγκεκριμένο τρόπο ονοματοδοσίας για τα αναγνωριστικά μεταβλητών, μεθόδων, κλάσεων, πακέτων κλπ, συγκεκριμένους κανόνες μορφοποίησης του πηγαίου κώδικα, και συγκεκριμένους κανόνες για την παράθεση εσωτερικών σχολίων στον πηγαίο κώδικα. Η συνεπής χρήση ενός συγκεκριμένου προγραμματιστικού στυλ επιτρέπει την υψηλή αναγνωσιμότητα και κατ' επέκταση την παραγωγικότητα της ομάδας ανάπτυξης.

5.2.6.2 Σενάρια Ελέγχου (Unit Tests)

Όλος ο κώδικας που θα αναπτυχθεί για την πλατφόρμα TRACER θα πρέπει να συνοδεύεται από σενάρια ελέγχου (Unit Tests) τα οποία μπορούν να χρησιμοποιηθούν για την επαλήθευση της σωστής λειτουργίας του. Τα σενάρια αυτά θα πρέπει να μπορούν να εκτελεστούν αυτόματα σε τακτικά χρονικά διαστήματα όταν προκύπτουν αλλαγές στον πηγαίο κώδικα του λογισμικού της πλατφόρμας. Κατά

προτίμηση για την εκτέλεση των σεναρίων θα χρησιμοποιείται η οικογένεια εργαλείων ελέγχου *Unit ανάλογα με τη γλώσσα του πηγαίου κώδικα της εφαρμογής (NUnit για γλώσσες της οικογένειας Microsoft.NET, JUnit για Java, CppUnit για C++).

5.2.6.3 Πρωτόκολλα επικοινωνίας

Η επικοινωνία μεταξύ διακριτών τμημάτων της πλατφόρμας που εκτελείται σε διαφορετικούς υπολογιστές (πχ η γραφική διεπαφή χρήστη και το κυρίως τμήμα της πλατφόρμας) θα πρέπει να γίνεται με τη χρήση εδραιωμένων πρωτοκόλλων που στηρίζονται στο HTTP ή/και HTTPS. Η επιλογή των επιμέρους πρωτοκόλλων που θα χρησιμοποιηθούν είναι στη διακριτική ευχέρεια της ομάδας ανάπτυξης.

5.2.6.4 Κωδικοποίηση δεδομένων

Όλα τα έγγραφα τα οποία προκύπτουν κατά την ανάπτυξη της πλατφόρμας TRACER, είτε αυτά περιέχουν πηγαίο κώδικα, είτε εσωτερική ή εξωτερική τεκμηρίωση, θα πρέπει να είναι κωδικοποιημένα σε μορφή Unicode UTF-8. Η συγκεκριμένη κωδικοποίηση είναι η προτιμητέα και για τη μεταφορά κειμένου κατά τη διάρκεια ανταλλαγής δεδομένων μέσω δικτύου.
Για τη μεταφορά και αποθήκευση δομημένων ή ημιδομημένων δεδομένων προτιμάται η χρήση της γλώσσας XML (eXtensible Markup Language) ή η γλώσσα JSON, που τείνει να καθιερωθεί ως de facto πρότυπο.

5.2.7 Τεκμηρίωση συστήματος

Η τεκμηρίωση ενός συστήματος λογισμικού περιλαμβάνει το σύνολο των πληροφοριών που περιγράφουν τον τρόπο με τον οποίο λειτουργεί και τον τρόπο με τον οποίο μπορεί να χρησιμοποιηθεί. Οι γενικές κατηγορίες στις οποίες διακρίνεται είναι η εσωτερική και η εξωτερική τεκμηρίωση.
Με τον όρο εσωτερική τεκμηρίωση περιγράφεται το σύνολο των πληροφοριών σχετικά με το σχεδιασμό και την υλοποίησή του λογισμικού και τις αποφάσεις που λήφθηκαν προκειμένου να καταλήξει στο σχεδιασμό αυτό η ομάδα ανάπτυξης. Η ύπαρξη και σωστή οργάνωση των πληροφοριών αυτών μπορεί να βοηθήσει σημαντικά στην αύξηση της παραγωγικότητας της ομάδας ανάπτυξης, αφού προσφέρει εύκολη και γρήγορη κατανόηση της αρχιτεκτονικής του.
Η εξωτερική τεκμηρίωση περιλαμβάνει το σύνολο εκείνο των πληροφοριών που περιγράφουν τον τρόπο λειτουργίας και χρήσης του λογισμικού, και συμπεριλαμβάνει σενάρια και παραδείγματα χρήσης. Τυπικά η εξωτερική τεκμηρίωση έχει διαφορετικές μορφές ανάλογα με την κατηγορία χρηστών στην οποία απευθύνεται, δεδομένου ότι τα σενάρια χρήσης είναι διαφορετικά για την καθεμιά.

5.2.7.1 Εσωτερική τεκμηρίωση

Στο πλαίσιο της πλατφόρμας TRACER η εσωτερική τεκμηρίωση θα πρέπει να περιλαμβάνει κατ' ελάχιστον τα παρακάτω:

1. Σχόλια που συνοδεύουν τον πηγαίο κώδικα του λογισμικού που θα αναπτυχθεί.
2. Διαγράμματα UML που περιγράφουν σε πιο υψηλό επίπεδο τις οντότητες από τις οποίες απαρτίζεται το λογισμικό.
3. Πιθανώς άλλα συνοδευτικά έγγραφα όπως διαγράμματα που συνοδεύονται από σχετικά σχόλια αναφορικά με αποφάσεις σχεδιασμού που λήφθηκαν κατά τις διάφορες φάσεις της ανάπτυξής του.

Η τεκμηρίωση αυτή θα πρέπει να ενσωματώνεται στα κατάλληλα σημεία του πηγαίου κώδικα, με τρόπο συνεπή σε σχέση με τα προγραμματιστικά πρότυπα που ακολουθεί η ομάδα ανάπτυξης, και πιθανώς να περιέχει αναφορές σε άλλα έγγραφα και τεχνουργήματα (artifacts) όπως τα διαγράμματα UML τα οποία δε μπορούν να ενσωματωθούν στον πηγαίο κώδικα. Τα σχόλια και τα υπόλοιπα είδη εσωτερικής τεκμηρίωσης πρέπει να δημιουργούνται και να εξελίσσονται παράλληλα και όχι να προστεθούν εκ των υστέρων, προκειμένου να διευκολυνθεί η διαδικασία ανάπτυξης.
Η λεπτομέρεια της τεκμηρίωσης πρέπει να είναι τέτοια που να καλύπτει όλες τις βασικές οντότητες από τις οποίες απαρτίζεται το λογισμικό, όπως κλάσεις, μεθόδους, κλπ. Η τεκμηρίωση των οντοτήτων που θα είναι δημόσια διαθέσιμες σε όποιον επιθυμεί να δημιουργήσει εφαρμογές και συστήματα που να συνεργάζονται με την πλατφόρμα πρέπει να ολοκληρωθεί κατά προτεραιότητα.

5.2.7.2 Εξωτερική τεκμηρίωση

Η εξωτερική τεκμηρίωση της πλατφόρμας TRACER θα πρέπει να είναι διαθέσιμη ξεχωριστά και ανά κατηγορία χρήστη. Το επίπεδο της λεπτομέρειας που πρέπει να χαρακτηρίζει το εκάστοτε έγγραφο είναι ανάλογο με την κάθε κατηγορία. Κάθε έγγραφο πρέπει να περιλαμβάνει τα παρακάτω:

1. Σύντομη περιγραφή των λειτουργιών που μπορεί να εκτελέσει η αντίστοιχη ομάδα χρηστών
2. Αναλυτική περιγραφή της καθεμιάς, που θα συνοδεύεται από σενάρια χρήσης, εικόνες (screenshots) από τις διεπαφές που θα παρέχει η πλατφόρμα, και εκτίμηση της δυσκολίας και του χρόνου που απαιτείται για την ολοκλήρωση της καθεμίας.
3. Ευρετήριο των περιεχομένων ανά τύπο λειτουργίας και ανά κατηγορία χρηστών, αν το ίδιο έγγραφο αφορά σε περισσότερες από μία κατηγορίες.

5.2.8 Πολιτικές και Ρυθμιστικό Πλαίσιο

Η ενότητα αυτή περιλαμβάνει απαιτήσεις σχετικές με νομικά θέματα και πολιτικές που σχετίζονται με τη φύση της πλατφόρμας TRACER.

5.2.8.1 Διεθνοποίηση Διεπαφών

Όλη η τεκμηρίωση του συστήματος θα πρέπει να γράφεται στην αγγλική γλώσσα αρχικά, αλλά θα πρέπει να επιτρέπει την μετάφραση σε άλλες γλώσσες, με εξαίρεση ορολογία και καθιερωμένα ακρώνυμα. Τα σχόλια στον πηγαίο κώδικα και η ονοματοδοσία των δομικών στοιχείων που απαρτίζουν το λογισμικό και των δημόσια διαθέσιμων προγραμματιστικών διεπαφών (API) θα πρέπει να είναι στην αγγλική γλώσσα.

5.2.8.2 Πολιτική κατά των Διακρίσεων

Η πλατφόρμα TRACER στο σύνολό της θα πρέπει να ακολουθεί μια πολιτική κατά των διακρίσεων σχετικά με τη φυλή, την καταγωγή, τη γλώσσα, την ηλικία, το φύλο και την απασχόληση διαφορετικών χρηστών. Όλα τα έγγραφα και τα συνοδευτικά τεχνουργήματα (πχ διαγράμματα) πρέπει να μην περιέχουν κείμενο, εικόνες ή αναφορές που είναι προσβλητικά έναντι κάποιας κοινωνικής ομάδας.

6 Περιγραφή διεπαφών συστήματος

6.1 Διεπαφές συστήματος

Η πλατφόρμα θα παρέχει ένα μικρό σύνολο διεπαφών που αφορούν την επικοινωνία με τη βάση δεδομένων στην οποία θα αποθηκεύονται οι χρήστες, τα προφίλ ασφάλειας, κλπ. καθώς και τις προγραμματιστικές διεπαφές (API) που θα πρέπει να παρέχει η πλατφόρμα για την επικοινωνία με τα plugins.

1. Διεπαφή με χώρο αποθήκευσης για καταγραφή ιστορικού συμβάντων (logs). Η πλατφόρμα TRACER θα χρησιμοποιεί την διεπαφή του λειτουργικού συστήματος για την επικοινωνία με το μέσο αποθήκευσης.
2. Διεπαφή με χώρο αποθήκευσης για καταγραφή αντιγράφων ασφαλείας.
    a. Η πλατφόρμα TRACER θα πρέπει να αλληλεπιδρά τόσο με τοπικά (στο ίδιο μηχάνημα) όσο και με απομακρυσμένα (πάνω από το δίκτυο) μέσα αποθήκευσης για την αξιόπιστη καταγραφή και ανάκτηση των αντιγράφων ασφαλείας.
    b. Θα πρέπει να υποστηρίζεται τόσο ολική ανάκληση όλων των δεδομένων της πλατφόρμας όσο και η μερική ανάκτηση συγκεκριμένων αρχείων.
    c. Λόγω της ύπαρξης πολλαπλών αντιγράφων από διαφορετικές ημερομηνίες θα πρέπει να υποστηρίζεται η γρήγορη επισκόπηση τον διαθέσιμων αντιγράφων για κάθε αρχείο και να δύνεται η επιλογή για ανάκτηση από συγκεκριμένη ημερομηνία.
3. Η πλατφόρμα TRACER θα επικοινωνεί με βάση δεδομένων για την αποθήκευση των παρακάτων πληροφοριών.
    a. Δεδομένα για τους χρήστες της πλατφόρμας. Για κάθε χρήστη θα κρατούνται κατ' ελάχιστο οι εξής πληροφορίες: Όνομα, επώνυμο, αναγνωριστικό (username), κωδικός

πρόσβασης στην πλατφόρμα μετά την εφαρμογή hash function, ρόλος στην πλατφόρμα όπως ορίζεται στην ενότητα 4.1, διεύθυνση ηλεκτρονικού ταχυδρομείου.

b. Πληροφορίες για τα αρχεία που παρακολουθούνται. Για κάθε αρχείο θα κρατούνται κατ' ελάχιστο οι εξής πληροφορίες. Όνομα αρχείου, γλώσσα προγραμματισμού την οποία περιέχει, ημερομηνία τελευταίας τροποποίησης, ημερομηνία τελευταίου ελέγχου για τρωτότητες, , λειτουργικά αρθρώματα (plugins) τα οποία είναι συμβατά ή/και ενεργά με αυτό το αρχείο, αναφορά τρωτοτήτων που έχουν εντοπιστεί στο αρχείο.
c. Πληροφορίες για τα λειτουργικά αρθρώματα. Για κάθε λειτουργικό άρθρωμα θα κρατούνται κατ' ελάχιστο οι εξής πληροφορίες: Όνομα λειτουργικού αρθρώματος, όνομα συγγραφέα, περιγραφή λειτουργίας του, τρωτότητες της οποίες ανιχνεύει.
d. Πληροφορίες για προβλήματα ασφαλείας που εντοπίστηκαν κατά τον έλεγχο του κώδικα. Για κάθε πρόβλημα ασφαλείας θα αποθηκεύονται κατ' ελάχιστο οι εξής πληροφορίες. Το πρόβλημα που εντοπίστηκε, το αρχείο στο οποίο εντοπίστηκε το πρόβλημα, αν το πρόβλημα διορθώθηκε αυτόματα ή όχι (ΠΧ3.04), το όνομα του λειτουργικού αρθρώματος που εντόπισε το πρόβλημα, η ημερομηνία που εντοπίστηκε το πρόβλημα.

4. Η επικοινωνία μεταξύ διακριτών τμημάτων της πλατφόρμας που εκτελείται σε διαφορετικούς υπολογιστές θα πρέπει να γίνεται με τη χρήση εδραιωμένων πρωτοκόλλων που στηρίζονται στο Internet Protocol.
5. Όλα τα έγγραφα τα οποία προκύπτουν κατά την ανάπτυξη της πλατφόρμας TRACER, είτε αυτά περιέχουν πηγαίο κώδικα, είτε εσωτερική ή εξωτερική τεκμηρίωση, θα πρέπει να είναι κωδικοποιημένα σε μορφή Unicode UTF-8.
6. Για τη μεταφορά και αποθήκευση δομημένων ή ημιδομημένων δεδομένων προτιμάται η χρήση της γλώσσας XML ή της JSON.
7. Η πλατφόρμα TRACER θα πρέπει να ορίζει μια προγραμματιστική διεπαφή μέσω της οποίας θα αλληλεπιδρά με λειτουργικά αρθρώματα (plugins). Η διεπαφή θα πρέπει να υποστηρίζει τουλάχιστον την εκκίνηση των αρθρωμάτων και την συλλογή τον αποτελεσμάτων τους.

6.2 Διεπαφές χρήστη

- Οι χρήστες θα επικοινωνούν με την πλατφόρμα TRACER μέσω διαδικτυακής γραφικής διεπαφής (Web-based user interface), η οποία είναι υποκατηγορία τον γραφικών διεπαφών (GUI). Η διεπαφή θα αποτελείται από ιστοσελίδες οι οποίες θα προβάλλονται στον φυλλομετρητή (browser) του χρήστη. Έτσι εξασφαλίζεται τόσο η τοπική (από το ίδιο μηχάνημα) όσο και η απομακρυσμένη (μέσω του Διαδικτύου) πρόσβαση στην πλατφόρμα.
- Κάθε ιστοσελίδα θα ακολουθείτε από συνοδευτικό *κείμενο που θα περιγράφει τον* τρόπο χρήσης της.
- Η διεπαφή θα πρέπει να είναι σύμφωνη με της απαιτήσεις απόδοσης 5.2.5.3 και 5.2.3.6.
- Όλοι οι χρήστες θα πρέπει να περνάνε από έλεγχο αυθεντικοποιήσης πριν αποκτήσουν πρόσβαση στην διεπαφή. Για την αυθεντικοποίηση οι χρήστες θα πρέπει να παρέχουν το αναγνωριστικό (username) και τον κωδικό τους, ώστε να ταυτοποιούνται με τα στοιχεία που είναι αποθηκευμένα στην βάση.
-

6.2.1 Διεπαφή διαχειριστή της πλατφόρμας TRACER

Η διεπαφή του διαχειριστή θα πρέπει να υποστηρίζει τις περιπτώσεις ΠΧ01, ΠΧ07-ΠΧ09. Η διεπαφή θα περιλαμβάνει μία καρτέλα Χρήστες με τις εξής επιλογές:

1. Προσθήκη Χρήστη: Ο διαχειριστής θα συμπληρώνει τα στοιχεία του χρήστη που αναφέρονται στην ενότητα 5.1.1.1 και επιλέγει Αποθήκευση Αλλαγών
2. Λειτουργικά Αρθρώματα με τις παρακάτω επιλογές:
   - Προσθήκη. Όπου ο διαχειριστής επιλέγει ένα η περισσότερα αρχεία που απαρτίζουν ένα λειτουργικό άρθρωμα για να προστεθεί στην πλατφόρμα.
   - Αφαίρεση. Όπου ο διαχειριστής επιλέγει ένα η περισσότερα λειτουργικά αρθρώματα για να τα αφαιρέσει στην πλατφόρμα.
   - Ενεργοποίηση. Όπου ο διαχειριστής επιλέγει ένα ή περισσότερα από τα λειτουργικά αρθρώματα που έχουν προστεθεί στην πλατφόρμα TRACER για ενεργοποίηση και επιλέγει την Ενεργοποίηση Επιλεγμένων

6.2.2 Διεπαφή υπεύθυνου λειτουργικών συστημάτων

Η διεπαφή του διαχειριστή συστήματος πρέπει να υποστηρίζει την περίπτωση ΠΧ02. Η διεπαφή θα περιλαμβάνει τις μια καρτέλα που θα αφορά την προσθήκη και την αφαίρεση Έργων Ο υπεύθυνος επιλέγει ένα ή περισσότερα έργα για να τα προσθέσει / αφαιρέσει από την πλατφόρμα.

6.2.3 Διεπαφή προγραμματιστή.

Η διεπαφή του προγραμματιστή θα πρέπει να υποστηρίζει της περιπτώσεις ΠΧ02, ΠΧ05 και ΠΧ06. Η διεπαφή του προγραμματιστή περιλαμβάνει την καρτέλα 6.2.2 1. 1. Προσθήκη / Αφαίρεση Έργων που περιγράφηκε προηγουμένως. Επιπλέον, *περιλαμβάνει τις εξής καρτέλες:*

1. Θωράκιση με τη χρήση βιβλιοθηκών ασφάλειας: Όπου ο προγραμματιστής ενσωματώνει στον κώδικα της εφαρμογής αναφορές και κατάλληλες κλήσεις για την χρήση των βιβλιοθηκών ασφάλειας που έχει προτείνει η πλατφόρμα.
2. Λειτουργικά Αρθρώματα με την παρακάτω επιλογή:
    - Δημιουργία Λειτουργικών Αρθρωμάτων. Όπου ο προγραμματιστής δημιουργεί κάποιο λειτουργικό άρθρωμα (Plugin) βασισμένο σε πρότυπο κώδικα που παρέχεται από την πλατφόρμα TRACER.

7 Συμπεράσματα

Ένας από τους βασικούς στόχους που καθορίζουν το σχεδιασμό της πλατφόρμας TRACER είναι να αποτελέσει ένα εργαλείο εύχρηστο και επεκτάσιμο, που να μπορεί να χρησιμοποιηθεί αποδοτικά από οργανισμούς που είτε θέλουν να θωρακίσουν παλαιότερα συστήματά τους, είτε να αναπτύξουν νέα τα οποία δε θα πάσχουν από συνηθισμένα προβλήματα ασφάλειας. Για την ανάπτυξη μιας τέτοιας πλατφόρμας, η ανάλυση απαιτήσεων παίζει πολύ σημαντικό ρόλο. Για τον λόγο αυτό η ανάλυση των απαιτήσεων που περιέχεται στο έγγραφο αυτό ακολουθεί σε μεγάλο βαθμό τις προτεινόμενες πρακτικές που παρουσιάζονται στο πρότυπο IEEE-830 “Recommended Practice for Software Requirements Specifications”. Όσον αφορά την ανάπτυξη της πλατφόρμας, μετά την εξέταση διάφορων εναλλακτικών προσεγγίσεων, αποφασίστηκε η χρήση μιας από τις πλέον διαδεδομένες μεθοδολογίες, η “Rational Unified Process” (RUP).

Για την ανάλυση των απαιτήσεων πραγματοποιήθηκαν αρχικά ημιδομημένες συνεντεύξεις των εμπλεκομένων και ενός αριθμού δυνητικών χρηστών της πλατφόρμας, λαμβάνοντας υπόψη και τα συμπεράσματα που προέκυψαν από το παραδοτέο 1.1 (Επισκόπηση του State of the Art). Από τις πληροφορίες που αντλήθηκαν από αυτές τις συνεντεύξεις και σε συνδυασμό με την δόμηση συγκεκριμένων περιπτώσεων χρήσης διαμορφώθηκαν συγκεκριμένες λειτουργικές και μη-λειτουργικές απαιτήσεις. Οι απαιτήσεις αυτές θα αποτελέσουν βασικό οδηγό τόσο για το λεπτομερή σχεδιασμό και την ανάπτυξη της πλατφόρμας, όσο και για την μελλοντική αξιολόγηση του κατά πόσον οι παραπάνω στόχοι επιτεύχθηκαν.

A. Παράρτημα Α – RUP/QFD

A.1 The rational unified process (RUP) - Επαναληπτική Ενοποιημένη Διαδικασία

Η RUP είναι μια μέθοδος διαχείρισης, σχεδιασμού και ανάπτυξης έργων λογισμικού μεγάλης κλίμακας. Θα μπορούσε να χαρακτηριστεί ως ένα ολοκληρωμένο πλαίσιο χειρισμού της ανάπτυξης λογισμικού το οποίο προβλέπει:

1. Επαναληπτική ανάπτυξη (τα επιμέρους τμήματα λογισμικού που αναπτύσσονται επανελέγχονται και συμπληρώνονται συνεχώς καθώς όλο και μεγαλύτερο εύρος του έργου υλοποιείται)

2. Την δομημένη διαχείριση των απαιτήσεων του έργου. Οι απαιτήσεις οργανώνονται σε θεμελιώδη αυτόνομα τμήματα λογισμικού με καλά καθορισμένη διεπαφή με το σύνολο του έργου.

3. Αρχιτεκτονικό σχεδιασμό που βασίζεται σε Components (ολοκληρωμένες ανεξάρτητες δομικές μονάδες λογισμικού)

4. Διαχείριση ποιότητας (Quality control and Management). Στοιχείου απαραίτητου στην μοντέρνα φιλοσοφία ανάπτυξης εφαρμογών

5. Δομημένο έλεγχο των αλλαγών και των επεκτάσεων του λογισμικού. Στοιχείο απαραίτητο γιατί κανένα λογισμικό δεν έχει ολοκληρωμένο το σύνολο των απαιτήσεων πριν και ίσως κατά τη διάρκεια της ανάπτυξης

6. Καλά ορισμένο πλαίσιο διαχείρισης των κειμένων που θα πρέπει να συνοδεύουν την ανάπτυξη λογισμικού. Για τα κείμενα της RUP υπάρχουν templates τα οποία συμπληρώνονται από τα επιφορτισμένα πρόσωπα και δημιουργούν ένα σύμπλεγμα τεκμηρίωσης που καθιστά εφικτή την διόρθωση, συμπλήρωση και τον έλεγχο ενός προϊόντος λογισμικού από τον οποιοδήποτε είναι «μυημένος» στη RUP ακόμα και αν έχουν παρέλθει έτη από την αποπεράτωση του έργου. Επίσης η τήρηση σωστής τεκμηρίωσης βοηθά την εύκολη εισαγωγή νέων προσώπων στις ομάδες ανάπτυξης και διαχείρισης ενός έργου.

7. Γραφικό τρόπο αναπαράστασης του σχεδιασμού με τη βοήθεια εμπορικών λογισμικών και UML γλώσσας προγραμματισμού και αναπαράστασης (πχ Rational Rose)

A.1.1 Γιατί να υιοθετήσουμε μια επαναληπτική διαδικασία ανάπτυξης

Μια επαναληπτική διαδικασία ανάπτυξης…:

- Αναγνωρίζει και αντιμετωπίζει ρεαλιστικά την καθημερινή πραγματικά της δυναμικής ανάγκης αλλαγής των απαιτήσεων που παρατηρείται σε μεγάλης κλίμακας έργα. Η εμπειρία δείχνει ότι το 30-50% των πραγματικών απαιτήσεων εμφανίζεται μετά τη φάση της ανάλυσης και αφού έχει ξεκινήσει η διαδικασία υλοποίησης του έργου.

- Παρέχει τη δυνατότητα ελέγχου και έγκαιρου εντοπισμού και περιορισμού του κινδύνου (risk management) διατέμνοντας το έργο σε επιμέρους υπο-έργα και αφήνοντας είτε όλη την ομάδα είτε μέρος αυτής να ασχοληθεί πρώτα και επισταμένα με τα τμήματα που ενέχουν μεγαλύτερο ρίσκο αποτυχίας.

- Επιτρέπει τη βήμα-βήμα ανάπτυξη του έργου κάνοντας συνδυασμένη σταδιακή (λίγο-λίγοο) ανάλυση-σχεδιασμό-υλοποίηση.

- Από την αρχή του έργου και μέχρι το τέλος δίνει τη δυνατότητα σε όλες της ομάδες (ελέγχου, ανάπτυξης, ανάλυσης) να συμμετέχουν στο έργο και να συνεισφέρουν την εμπειρία τους.

- Η ύπαρξη πολλών επαναλήψεων δημιουργεί τη δυνατότητα της διόρθωσης και του

- επαναπροσδιορισμού των επιλογών και χτίζει εμπειρία στο σύνολο των ομάδων που συνεργάζονται.

- Εκ των πραγμάτων έχει τη φιλοσοφία της βήμα προς βήμα ανάλυσης και ανάπτυξης και άρα έχει σαν βασικό συστατικό την ανάπτυξη θεμελιωδών τμημάτων κώδικα (component). Έτσι αποφεύγει τις «τελικές λύσεις» τύπου Big-Band που έχουν μεγάλο κίνδυνο αποτυχίας ενώ ταυτόχρονα εξασφαλίζει το ποθητό modularity.

Α.1.2 Οι βασικοί στόχοι κάθε επαναληπτικής διαδικασίας

Ο βασικός στόχος κάθε επαναληπτικής διαδικασίας είναι να μειώνει σιγά σιγά τον κίνδυνο αποτυχίας του έργου αντιμετωπίζοντας σταδιακά:

1. Τους κινδύνους απόδοσης (αργές DB, προβληματικά κανάλια και πρωτόκολλα μετάδοσης δεδομένων, διαδικασίες που καταναλώνουν χρόνο δυσανάλογο προς το αποτέλεσμα)

2. Τους κινδύνους ολοκλήρωσης (υιοθέτηση διαφορετικών τεχνολογιών, επιλογή πλατφόρμας, γλώσσες προγραμματισμού)

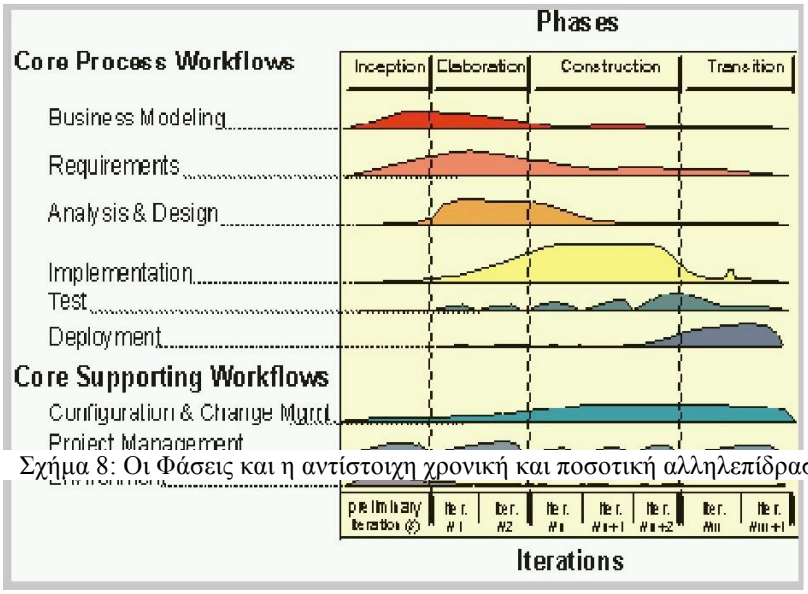
3. Τους κινδύνους βασικής φιλοσοφίας. Κινδύνους που προκύπτουν από βασικές αρχιτεκτονικές επιλογές πχ. Client – Server, intranet, όπως επίσης και κινδύνους από επιλογές δόμησης των ομάδων που συμμετέχουν ή ακόμα και εταιριών που αναλαμβάνουν τα υπο-έργα

4. Η κάθε επαναληπτική διαδικασία δομείται σαν ένα μικρό έργο «waterfall» το οποίο ολοκληρώνεται με την παράδοση ενός λογισμικού διαθέσιμου για περαιτέρω ανάλυση και διόρθωση από τις εμπλεκόμενες ομάδες.

5. Η κάθε επανάληψη είναι καθοδηγούμενη από τον κίνδυνο γι' αυτό και θα πρέπει να αποτελείται από εκείνα τα επιμέρους τμήματα λογισμικού τα οποία αναδεικνύουν τους πιθανούς κινδύνους και τις περιπτώσεις αποτυχίας.

6. Τα αποτελέσματα τις κάθε επανάληψης θα πρέπει να προωθεί το έργο προσθέτοντας συνεχώς καινούργια στοιχεία (πχ ανανεωμένες Beta Versions)

A.1.3 Διαχείριση κινδύνου



Σχήμα 8: Οι Φάσεις και η αντίστοιχη χρονική και ποσοτική αλληλεπίδραση

Μέσα από τη RUP η διαχείριση κινδύνου γίνεται από την αναγνώριση των πηγών και επιλογών αυξημένου κινδύνου. Την σταδιακή και συνεχώς αυξανόμενη ανάπτυξη του έργου. Από την ανάπτυξη πρωτοτύπου και πιλοτικού έργου που καλύπτει τις πιο ριψοκίνδυνες πτυχές του συνολικού έργου. Καθώς επίσης την έγκαιρη δυνατότητα για έλεγχο των αποτελεσμάτων με τη σωστή ιεράρχηση της σειράς ανάπτυξης των υπο-έργων.

A.1.4 Οι Φάσεις Ανάπτυξης

Οι Φάσεις Ανάπτυξης είναι οι εξής:

- Φάση εισαγωγής (Inception Phase)
- Φάση επεξήγησης και επέκτασης-κλιμάκωσης του έργου (Elaboration Phase)
- Φάση κατασκευής (Construction Phase)
- Φάση μετάβασης στο περιβάλλον του χρήστη (Transition Phase)

Φάση εισαγωγής (Inception Phase)

Ο βασικός και θεμελιώδης στόχος είναι η προκαταρκτική συνεργασία όλων των ενδιαφερομένων ομάδων και πλευρών. Επίσης θα πρέπει να εξασφαλιστεί η συνεργασία (με τον ορισμό contact persons) όλων των τμημάτων και ομάδων χρηστών που δημιουργούν εκ των πραγμάτων τις απαιτήσεις. Η αρχική καταγραφή γίνεται με τη συνεργασία των χρηστών και με καλά καθορισμένες (agenda) συναντήσεις. Επίσης οι αναλυτές θα πρέπει εκ των προτέρων να έχουν ανατρέξει σε βιβλιογραφία και σε παράπλευρες διερευνήσεις του αντικειμένου του έργου ώστε να έχουν καλά καθορισμένες ερωτήσεις προς τους χρήστες. (Βασικοί αρχή… ο χρήστης ποτέ δεν προετοιμάζεται πριν την συνάντηση ώστε να παρέχει στοιχεία). Με το τέλος κάθε συνάντηση οι χρήστες θα πρέπει να λαμβάνουν κείμενο που περιγράφει το περιεχόμενο της συνάντησης καθώς και

μια ανάλυση και δόμηση των όσων συζητήθηκαν. Ο χρήστης θα πρέπει να κάνει κριτική στα όσα γράφονται. Η βασική μονάδα κοστολόγησης κάθε εταιρίας είναι ο εργατομήνας. Είναι το μέσω κόστος παρουσίας ενός προσώπου που ανήκει σε ένα επίπεδο αμοιβών σε μία ομάδα. Ο σωστός προϋπολογισμός του έργου προδιαγράφει και το ποιοτικό αποτέλεσμα του έργου. Επίσης στον προϋπολογισμό του έργου θα πρέπει να συμπεριλαμβάνονται και έξοδα μετακίνησης, εκπαίδευσης, μεταφοράς τεχνογνωσίας, αγορών υλικού και λογισμικού. Η αρχική ανάλυση κινδύνου πρέπει να συμπεριλάβει όλα εκείνα τα σημεία που παρουσιάζουν δυσκολία ανάλυσης, ανάπτυξης αλλά και έχουν υψηλό κίνδυνο αποτυχία λόγο έλλειψης εμπειρίας. Ο καθορισμός του σκοπού του έργου. Έχει σημασία η σωστή προδιαγραφή του έργου ώστε το όλο εγχείρημα να περιοριστεί στα απολύτως απαραίτητα επίπεδα. (Μην σκοτώνεις ένα κουνούπι με ένα κανόνι!) Ο καθορισμός μιας υποψήφιας πλατφόρμας ανάπτυξης και υλοποίησης. Η πλατφόρμα που θα επιλεγεί θα πρέπει να μπορεί να καλύψει της απαιτήσεις του έργου, αλλά και να υπάρχει εμπειρία (ή να προϋπολογιστεί σωστά η μεταφορά τεχνογνωσίας) από τις ομάδες υλοποίησης. Η κατασκευή ενός πιλοτικού πρωτοτύπου πιλοτικού μπορεί να παραληφθεί με την επίσπευση του πρώτου iteration το οποίο θα συμπεριλάβει βασικά συστατικά στοιχεία του έργου. Αρχικά Use Case μοντέλα (10% - 20% ολοκλήρωση για το καθένα). Η αρχική περιγραφή των Use Cases βοηθάει πολύ στην κατανόηση του έργου. Σε αυτό το στάδιο θα πρέπει η περιγραφή να περιλαμβάνει βασικά στοιχεία όπως σύντομη περιγραφή και ποιοι οι χρήστες (ρόλοι) που αλληλεπιδρούν με το συγκεκριμένο Use Case.

Φάση Κλιμάκωσης (Elaboration Phase)

Η φάση ανάλυσης και κλιμάκωσης του έργου περιλαμβάνει τους εξής στόχους:

- Ανάλυση των Use Case
    - Καθορισμός και ανάλυση του 80% των σημαντικότερων use cases και τους αντίστοιχους ρόλους που συμμετέχουν μέχρι το τέλος αυτής της φάσης.
    - Δημιουργία των Prototype User Interfaces (dummy screens)
    - Ιεραρχούμε τα Use Cases μέσα στο Use Case Model μα βάση τη δυσκολία τους αλλά και το ρίσκο που εμπεριέχουν για την ολοκλήρωση του έργου
    - Ανάλυση σε βάθος των πιο σημαντικών Use Cases (γράψιμο και διόρθωση τους)
- Προετοιμασία ενός μοντέλου από σημαντικές για το έργο κλάσεις και εκτίμηση της αξίας τους.

  Δημιουργία ενός βασικού User Interface.

Φάση Κατασκευής Construction Phase)

Η φάση κατασκευής επικεντρώνει το ενδιαφέρον της στην κατασκευή του έργου. Η φάση κατασκευής χαρακτηρίζεται από:

- Σταδιακή ανάπτυξη της λειτουργικότητας του έργου
- Η ανάπτυξη γίνεται σε όλο το βάθος του έργου και όχι επιλεκτικά όπως στις προηγούμενες φάσεις
- Το έργο αποκτά σταδιακά ισορροπία και προσεγγίζει στη μορφή του τελικού παραδοτέου.
- Κατασκευάζεται κάθε επιμέρους λεπτομέρεια που στις προηγούμενες φάσεις δεν αντιμετωπιζόταν λόγο μικρού ρίσκου
- Η ανάλυση συνεχίζεται και βελτιώνονται τα UC αλλά η ανάπτυξη του software παίζει πλέον πρωτεύοντα ρόλο.

Το παραγόμενο αυτής της φάσης είναι έτοιμο προϊόν για να παραδοθεί στους τελικούς χρήστες. Κατ' ελάχιστο το προϊόν αποτελείτε από: Το λογισμικό κατασκευασμένο στις κατάλληλες πλατφόρμες, τα user manuals και μια περιγραφή της εφαρμογής.

Τα σημεία που θα πρέπει να εξασφαλισθούν είναι:

- Είναι το προϊόν σταθερό και αρκετά έτοιμο για παράδοση στους τελικούς χρήστες;
- Είναι το πραγματικό πλέον έργο αρκετά κοντά στο σχεδιαζόμενο;

Φάση Μετάβασης στο περιβάλλον του χρήστη (Transition Phase)

Η φάση αυτή αποτελεί την μετάβαση και εφαρμογή του έργου στο περιβάλλον των τελικών χρηστών. Περιλαμβάνει την τυποποίηση (ανάπτυξη install application), την αποστολή, την εκπαίδευση και την τεχνική υποστήριξη. Η ομάδα των developers μειώνεται σε αριθμό και σε συμμετοχή. Ο έλεγχος του έργου μεταφέρεται στην ομάδα υποστήριξης. ALFA, BETA releases και τελικό προϊόν. Ολοκλήρωση με άλλα λογισμικά και υποδομή της εταιρίας του χρήστη
Αυτή η φάση περιλαμβάνει:

- Beta testing" για να αξιολογηθεί το προϊόν σε σχέση με τις προδιαγραφές και τις απαιτήσεις
- Παράλληλη λειτουργία του προϊόντος με τα υπάρχοντα συστήματα ώστε να αξιολογηθεί η ικανότητα του να τα αντικαταστήσει
- Μετατροπή των δεδομένων από την μορφή των παλαιών συστημάτων που είχε η εταιρία (πχ παλαιές databases) στις απαιτήσεις του νέου προϊόντος (νέες databases)
- Εκπαίδευση τόσο των χρηστών όσο και του τμήματος υποστήριξης
- Προώθηση του προϊόντος προς άλλα τμήματα της δικιάς μας εταιρίας (marketing, sales)

A.2 The Quality Function Deployment (QFD)

A.2.1 Ορισμός και ιστορική αναδρομή

Η μεθοδολογία Quality Function Deployment (QFD) είναι ένα σύστημα διαχείρισης ποιότητας προσανατολισμένο προς τον πελάτη, το οποίο ενσωματώνει τη "φωνή του πελάτη" (Voice of the customer, VOC) σε όλα τα στάδια ανάπτυξης ενός προϊόντος, από το σχεδιασμό του και τη διαδικασία ανάπτυξής του, μέχρι την παραγωγή και τη διανομή, με στόχο να επιτύχει τη μεγαλύτερη δυνατή ικανοποίηση των πελατών από το προϊόν.

A.2.2 Περιγραφή του μοντέλου

Δύο είναι τα πιο γνωστά μοντέλα εφαρμογής της μεθοδολογίας QFD:
α) Το μοντέλο των Τεσσάρων Φάσεων (ASI, 1987), γνωστό και ως μοντέλο ASI (American Supplier Institute) και
β) το μοντέλο Πίνακα Πινάκων (Matrix of Matrices) του Akao (Akao, 1990).

Το μοντέλο των Τεσσάρων Φάσεων αποτελεί τον αναλυτικό σχεδιασμό για την ανάπτυξη προϊόντων και καλύπτει τα βασικά βήματα ανάπτυξης ενός προϊόντος, ενώ το μοντέλο Πίνακα Πινάκων υποστηρίζει καλύτερα τις ανάγκες της Διοίκησης Ολικής Ποιότητας (Total Quality

Management, TQM) και επίσης καλύπτει δραστηριότητες, όπως σχεδιασμός αξιοπιστίας, ανάλυση κόστους και έλεγχος ποιότητας παραγωγής.

Το μοντέλο των Τεσσάρων Φάσεων περιλαμβάνει τα ακόλουθα στάδια εφαρμογής : Σχεδιασμός προϊόντος – Ανάπτυξη προϊόντος – Σχεδιασμός διεργασιών – Σχεδιασμός παραγωγής.

Η αναλυτική περιγραφή των τεσσάρων φάσεων που ακολουθεί βασίζεται σε μια πρόσφατη και πολύ περιεκτική εργασία του Mehrjerdi (2010).

Φάση Ι (Σχεδιασμός προϊόντος)

Οι απαιτήσεις των πελατών («φωνή του πελάτη») μεταφράζονται σε ανεξάρτητα, μετρήσιμα και ποιοτικά χαρακτηριστικά σχεδιασμού του προϊόντος.

Φάση ΙΙ (Ανάπτυξη προϊόντος)

Στη φάση αυτή εξετάζεται η σχέση μεταξύ των ποιοτικών χαρακτηριστικών σχεδιασμού που προέκυψαν από τη Φάση Ι και των διαφόρων κατασκευαστικών στοιχείων ή τμημάτων του σχεδίου. Το αποτέλεσμα της φάσης είναι η ιεράρχηση των συστατικών στοιχείων του σχεδιασμού, με βάση την ικανότητά τους να πληρούν το επιθυμητό επίπεδο επιδόσεων των ποιοτικών τους χαρακτηριστικών.

Φάση ΙΙΙ (Σχεδιασμός διεργασιών)

Στη φάση αυτή πραγματοποιείται ιεράρχηση των προδιαγραφών και των διεργασιών κατασκευής για τις βασικές παραμέτρους που θα αναπτυχθούν στη συνέχεια στο τέταρτο και τελευταίο στάδιο.

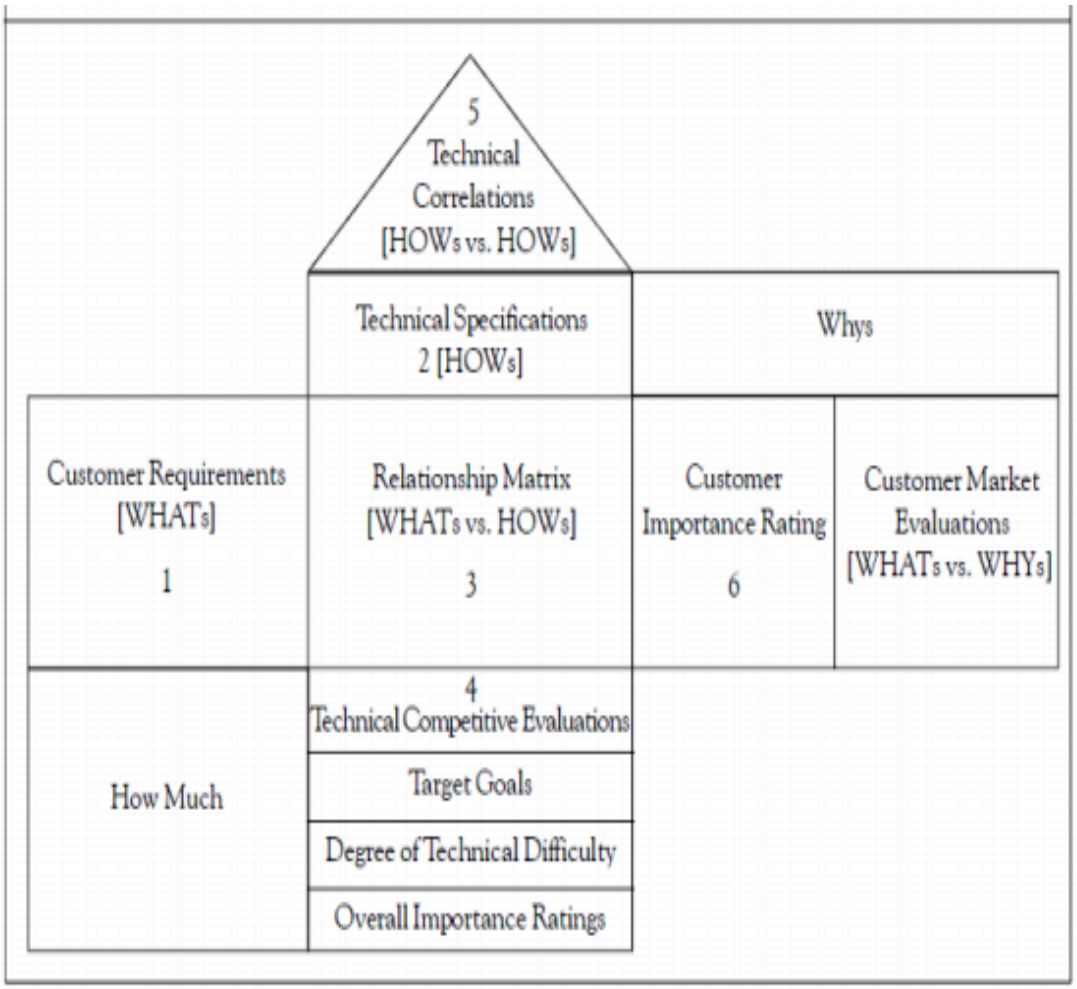
Φάση ΙV (Σχεδιασμός παραγωγής)

Οι βασικές κατασκευαστικές διεργασίες και συναφείς παράμετροι μετατρέπονται σε οδηγίες κατασκευής, σχέδια ελέγχου και αντίδρασης και απαιτήσεις εκπαίδευσης που είναι αναγκαία για να διασφαλιστεί η ποιότητα των βασικών διεργασιών και τμημάτων του τελικού προϊόντος.

Το πρώτο στάδιο (House of Quality)

Η τεχνική που χρησιμοποιείται στην πρώτη φάση (Σχεδιασμός του προϊόντος) είναι το «σπίτι ποιότητας» (House of Quality, HOQ). Το HOQ μεταφράζει τη «φωνή του πελάτη» (VOC) σε απαιτήσεις σχεδιασμού που ικανοποιούν συγκεκριμένες τιμές και τις συσχετίζει με τη στρατηγική που εφαρμόζει η επιχείρηση ή ο οργανισμός για να ανταποκριθεί στις απαιτήσεις αυτές. Η «φωνή του πελάτη» συλλέγεται με διάφορους τρόπους και από διάφορες πηγές, όπως έρευνες γνώμης, συνεντεύξεις με πελάτες και πωλητές, ομάδες εστίασης (focus groups), εμπορικές εκθέσεις και περιοδικά και παράπονα πελάτη (Bossert, 1991).

Η μεθοδολογία περιλαμβάνει τη συμπλήρωση έξι πινάκων που αλληλεπιδρούν κατά τέτοιο τρόπο, ώστε να δίνουν την εικόνα ενός σπιτιού (Εικόνα 2.3). Ακολουθεί μια σύντομη παρουσίαση των έξι πινάκων με βάση την εργασία των Chan L.-K., and Wu M.-L. (2003).



Σχήμα 9: House of Quality

Οι έξι συσχετιζόμενοι πίνακες περιγράφονται στη συνέχεια ως συστατικά μέρη του σπιτιού.

- Απαιτήσεις του Πελάτη (WHATs): Η αριστερή εξωτερική πλευρά του σπιτιού αντιπροσωπεύει τις απαιτήσεις του πελάτη (voice of the customer) και προσδιορίζεται με έρευνες αγοράς και άλλες μεθόδους.
- Τεχνικές Προδιαγραφές (HOWs): Ο πάνω όροφος του σπιτιού περιέχει τις τεχνικές προδιαγραφές που περιγράφουν πώς το προϊόν ή η υπηρεσία μπορεί να επιτύχει τις απαιτούμενες επιδόσεις ή αποτελέσματα σε γενικές γραμμές, χωρίς να αποτελούν εξειδικευμένη τεχνικά λύση και αντιπροσωπεύουν τη φωνή του σχεδιαστή (της επιχείρησης).
- Πίνακας Συσχετίσεων (Relationship Matrix): Το εσωτερικό του σπιτιού καταλαμβάνει ο βασικός πίνακας που περιγράφει το βαθμό συσχέτισης μεταξύ των απαιτήσεων των πελατών και των τεχνικών προδιαγραφών.
- Πίνακας Ιεράρχησης (Technical Matrix): Οι τεχνικές προδιαγραφές ιεραρχούνται στο κάτω μέρος του σπιτιού. Η ιεράρχηση γίνεται βάσει των συσχετίσεων μεταξύ των απαιτήσεων των πελατών και των τεχνικών προδιαγραφών. Δεδομένα όπως η συγκριτική τεχνική αξιολόγηση, ο βαθμός τεχνικής δυσκολίας, καθώς και οι τιμές-στόχοι απαριθμούνται και χρησιμοποιούνται για την ιεράρχηση των τεχνικών προδιαγραφών.
- Αλληλεπιδράσεις τεχνικών χαρακτηριστικών (Technical Correlations): Η στέγη του σπιτιού, είναι ένας πίνακας στον οποίο αποτυπώνονται ποσοτικά οι αλληλεπιδράσεις μεταξύ των τεχνικών προδιαγραφών. Αυτή η διαδικασία μας επιτρέπει να ελέγξουμε σε ποιο βαθμό οι τεχνικές προδιαγραφές μπορεί να είναι αμοιβαία υποστηριζόμενες ή ανταγωνιστικές.
- Πίνακας Σχεδιασμού (Planning Matrix): Ο πίνακας αυτός περιλαμβάνει ποσοτικά δεδομένα για καθένα από τα χαρακτηριστικά του πελάτη. Τα δεδομένα μπορούν να προέρχονται από

έρευνα αγοράς, ανάλυση ανταγωνισμού ή ομάδα αξιολόγησης και χρησιμοποιούνται για να ιεραρχηθούν οι απαιτήσεις του πελάτη.

Β. Παράρτημα Β – Ερωτήσεις για ημι-δομημένη συνέντευξη

Ερωτήσεις για ημι-δομημένη συνέντευξη

1. Ποιες είναι οι σημαντικότερες τρωτότητες που αντιμετωπίζετε / έχετε αντιμετωπίσει σε εφαρμογές και συστήματα που χρησιμοποιείτε;

2. Χρησιμοποιείτε κάποια μέθοδο αντιμετώπισης των τρωτοτήτων αυτών; Αν ναι:
    a. Είναι αυτοματοποιημένη ή αντιμετωπίζετε κάθε περίπτωση ξεχωριστά και με βάση τις γνώσεις του κάθε προγραμματιστή;
    b. Σε ποιο στάδιο της διαδικασίας (π.χ. σχεδιασμός, υλοποίηση) χρησιμοποιείται αυτή η μέθοδος;
    c. Σε ποιο επίπεδο της εφαρμογής (λογισμικό, βάση δεδομένων, δίκτυο - υλικό) χρησιμοποιείται αυτή η μέθοδος;

3. Σε τι περιβάλλοντα ανάπτυξης (framework) δουλεύετε;

4. Ποιο συντακτικό χρησιμοποιείτε;

5. Δουλεύετε σε επίπεδο έργου ή προϊόντος;

6. Έχετε πιστοποίηση (ISO) στη μεθοδολογία που ακολουθείτε και πως την εφαρμόζεται στην πράξη;

7. Σε ένα σύστημα / σετ εργαλείων για την ανίχνευση τρωτοτήτων σε επίπεδο κώδικα και διαδικτύου (σαν το TRACER), τι απαιτήσεις έχει ο τελικός χρήστης;

8. Άλλα σχόλια / παρατηρήσεις;

Σχήμα 10: Ερωτήσεις που ερωτήθηκαν σε άτομα εντός και εκτός της SingularLogic

C. Παράρτημα Γ – Γλώσσες Προγραμματισμού

Η αξιολόγηση των διάφορων γλωσσών προγραμματισμού που χρησιμοποιούνται στην υλοποίηση συστημάτων παλαιών γενεών έγινε με βάση τον αριθμό των γραμμών του κώδικα της κάθε μιας. Στο δείγμα που χρησιμοποιήσαμε συμπεριλάβαμε 50 διαφορετικά (μικρά και μεγάλα) συστήματα παλαιών γενεών τα οποία καλύπτουν το μεγαλύτερο μέρος των δραστηριοτήτων της εταιρείας μας.

Για την μέτρηση των γραμμών του κώδικα χρησιμοποιήθηκαν τόσο χειροκίνητοι τρόποι (π.χ. αναζήτηση συγκεκριμένων regular expression) όσο και εργαλεία που επιτρέπουν την μέτρηση των γραμμών του κώδικα παραβλέποντας τα σχόλια ή τις κενές γραμμές (π.χ. Clock Tool[4]).

Στον παρακάτω πίνακα μπορείτε να δείτε τα συγκεντρωτικά αποτελέσματα:

Γλώσσα Προγραμματισμού	Αριθμός Γραμμών Κώδικα	%
C#	23.918.117	63,1752%
Delphi	8.500.000	22,4512%
XML	1.375.201	3,6323%
Java	1.096.677	2,8967%
VisualBasic	859.083	2,2691%
Oracle PL-SQL Code	448.842	1,1855%
ASP	275.249	0,7270%
HTML	258.234	0,6821%
MSBUILD scripts	253.162	0,6687%
Objective C	160.845	0,4248%
XSLT	151.227	0,3994%
ASP .Net	116.457	0,3076%
Javascript	110.189	0,2910%
SKILL	95.018	0,2510%
JSP	82.852	0,2188%
PHP	40.259	0,1063%
CSS	37.739	0,0997%
XSD	34.558	0,0913%
Pascal	22.704	0,0600%
C/C++ Header	19.541	0,0516%
XAML	2.616	0,0069%
C++	1.397	0,0037%
Σύνολο	37.859.967	100,00%

Σχήμα 11: Στοιχεία σχετικά με τις γλώσσες προγραμματισμού και τις γραμμές κώδικα σε αυτές

[4] http://cloc.sourceforge.net/

D. Παράρτημα Δ – Γλωσσάρι

Όρος	Ορισμός	Πηγή
Εμπλεκόμενοι (stakeholders)	Οι Εμπλεκόμενοι (stakeholders) σε ένα έργο είναι οι οντότητες εντός ή εκτός ενός οργανισμού που χρηματοδοτούν το έργο, που ενδιαφέρονται ή κερδίζουν από την επιτυχή ολοκλήρωση του έργου ή επηρεάζουν, θετικά ή αρνητικά, την ολοκλήρωση αυτού.	http://en.wikipedia.org/wiki/Project_stakeholder
Περίπτωση Χρήσης	Μια Περίπτωση Χρήσης περιγράφει μια σειρά από ενέργειες που εκτελεί το σύστημα ώστε να αποδώσει χρήσιμα αποτελέσματα σε ένα ρόλο του συστήματος (χρήστη, άλλο συνεργαζόμενο λογισμικό).	
UML	Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of object-oriented software engineering. It is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as: activities, actors, business processes, database schemas, (logical) components, programming language statements and reusable software components.	Wikipedia: http://en.wikipedia.org/wiki/Unified_Modeling_Language
Διάγραμμα Περιπτώσεων Χρήσης	Στην Unified Modeling Language, οι σχέσεις μεταξύ όλων (ή ενός σετ από) των περιπτώσεων χρήσης και των δραστών παρουσιάζονται σε ένα ή περισσότερα Διαγράμματα Περιπτώσεων Χρήσης.	Wikipedia: http://en.wikipedia.org/wiki/Use_case
Λειτουργικές Απαιτήσεις Συστήματος	Περιγράφουν τι πρέπει να κάνει το σύστημα (π.χ. ως συναρτήσεις που λαμβάνουν είσοδο και δίδουν έξοδο)	http://www.arnos.gr/dmdocuments/yliko/plhroforiki/texnologialogismikou/shmeiwseis.leitourgikes.apaithseis.g.tziztikas.panmio.kritis.pdf
Μη-λειτουργικές Απαιτήσεις Συστήματος	Περιγράφουν ιδιότητες του συστήματος που συνήθως εκφράζονται βάσει χαρακτηριστικών της μορφής: • Απόδοση (performance) • Χρηστικότητα (usability) • Ασφάλεια (security) • Νομιμότητα (legislative) • Ιδιωτικότητα (privacy) Περιγράφουν το πώς (ή το πόσο καλά) το σύστημα θα υποστηρίξει τις λειτουργικές απαιτήσεις.	http://www.arnos.gr/dmdocuments/yliko/plhroforiki/texnologialogismikou/shmeiwseis.leitourgikes.apaithseis.g.tziztikas.panmio.kritis.pdf

Πίνακας 4: Γλωσσάρι

Ε. Παράρτημα Ε – Βιβλιογραφία

1. Μαρνελάκης Εμμανουήλ, Δραγατάκη Ευσταθία, Οικονομοτεχνική μελέτη ανάπτυξης λογισμικού σε εταιρεία που εφαρμόζει το σύστημα RUP, http://nefeli.lib.teicrete.gr/browse/sdo/log/2008/MarnelakisEmmanouil/document.tkl
2. Μιχαήλ Λιτσαρδάκη, Ανάπτυξη Μοντέλου QFD (Quality Function Deployment) με τη χρήση ποσοτικής προσέγγισης στον κλάδο του τουρισμού, http://dspace.lib.uom.gr/bitstream/2159/14302/1/QFD+%CF%83%CF%84%CE%BF%CE%BD+%CE%A4%CE%BF%CF%85%CF%81%CE%B9%CF%83%CE%BC%CF%8C.pdf
3. Chatzieleftheriou G. and Katsaros P.: "Test Driving Static Analysis Tools in Search of C Code Vulnerabilities", Proceedings 35th IEEE Computer Software and Applications Conference Workshops (COMPSACW'2011), pp.96-103, Munich, Germany, 2011