ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΜΗΧΑΝΙΚΗ ΤΩΝ ΑΠΑΙΤΗΣΕΩΝ»

Της φοιτήτριας
Μαρίας Δαλαβίκα
Αρ. Μητρώου: 03/2384

Επιβλέπων καθηγητής
Ιγνάτιος Δεληγιάννης

Θεσσαλονίκη 2010

ΠΡΟΛΟΓΟΣ

Η εργασία αυτή έχει ως σκοπό την ανάδειξη της σημασίας και του καθοριστικού ρόλου της Μηχανικής των Απαιτήσεων, στην ορθή ανάπτυξη των Πληροφοριακών Συστημάτων**.** Σκοπός μας είναι, ο σαφής καθορισμός των απαιτήσεων στα πρώτα στάδια της διαδικασίας ανάπτυξης λογισμικού, που είναι το θεμέλιο πάνω στο οποίο κτίζεται το υπόλοιπο έργο. Ένα λάθος σ' αυτό το στάδιο αν δεν διορθωθεί μπορεί να κοστίσει πολύ ακριβά και κάποιες φορές είναι προτιμότερο να γίνεται εξ ολοκλήρου από την αρχή το έργο.

Αρχικά γίνεται μία εισαγωγή στην έννοια των Πληροφοριακών Συστημάτων, δίνοντας τον ορισμό, καθορίζοντας το πλαίσιο των δραστηριοτήτων και αναφέροντας την ιστορική εξέλιξη αυτών. Στα επόμενα κεφάλαια, περιγράφεται αναλυτικότερα ο τρόπος ανάπτυξης των Πληροφοριακών Συστημάτων για να φτάσουμε τελικά να καταλάβουμε ποια είναι η σημασία της Μηχανικής των Απαιτήσεων στην ανάπτυξη ενός έργου λογισμικού καθώς και ποιες είναι οι τεχνικές για την εξαγωγή των απαιτήσεων .

Για την ανάπτυξη των θεμάτων που περιγράφονται στην εργασία χρησιμοποιήθηκε ως κύρια πηγή το διαδίκτυο, επιστημονικά συγγράμματα και άρθρα τα οποία προσεγγίζουν πολύπλευρα το θέμα «Μηχανική των Απαιτήσεων» .

Θα ήταν παράλειψη να μην ευχαριστήσω ιδιαιτέρως τον επιβλέποντα καθηγητή μου κύριο Ιγνάτιο Δεληγιάννη καθώς και τον κύριο Αποστόλη Αμπατζόγλου που με βοήθησαν στην εκπόνηση της εργασίας αυτής, τους γονείς μου αλλά και ορισμένους φίλους των οποίων η βοήθεια ήταν πολύτιμη για την ολοκλήρωση της πτυχιακής μου εργασίας.

ΠΕΡΙΛΗΨΗ

Με τον όρο Μηχανική των Απαιτήσεων στα συστήματα λογισμικού, εννοούμε το τμήμα της Μηχανικής Λογισμικού που περιλαμβάνει τις εργασίες διερεύνησης των αναγκών ή των κατάλληλων συνθηκών, που απαιτούνται για την επίτευξη ενός νέου προϊόντος ή την τροποποίηση ενός ήδη υπάρχοντος, λαμβάνοντας υπ' όψιν τις πιθανόν αντικρουόμενες ανάγκες των διαφόρων φορέων, όπως οι δικαιούχοι ή οι χρήστες.

Η Μηχανική Εξαγωγής των Απαιτήσεων είναι ένα κρίσιμο στάδιο για την επιτυχία ενός έργου λογισμικού. Η σοβαρότητα του κινδύνου της άγνοιας των απαιτήσεων, έγκειται στο γεγονός ότι η καταγραφή των απαιτήσεων γίνεται στα πρώτα στάδια της διαδικασίας ανάπτυξης λογισμικού, που είναι το θεμέλιο πάνω στο οποίο κτίζεται το υπόλοιπο έργο. Ένα λάθος σ' αυτό το στάδιο, αν δεν διορθωθεί μπορεί να κοστίσει πολύ ακριβά και κάποιες φορές είναι προτιμότερο να γίνεται εξ ολοκλήρου από την αρχή το έργο. Ενδεικτικά αναφέρονται δύο σχετικά παραδείγματα:

- Performing Rights Society, PROMS project. Εγκαταλείφθηκε το 1992 μετά τη δαπάνη 11 (έντεκα) εκατομμυρίων λιρών. Η μη πλήρης εφαρμογή της Μηχανικής των Απαιτήσεων ήταν ένας προεξέχων παράγοντας. Αναφέρεται ότι απέτυχαν να παρουσιάσουν τις απαιτήσεις σε φόρμα κατανοητή και ελεγχόμενη από ανθρώπους που δεν ανήκουν στον κλάδο της πληροφορικής και τέλος η τεκμηρίωση των προδιαγραφών δεν ήταν εύκολα αντιληπτές.
- London Stock Exchange TAURUS project. Ακυρώθηκε το 1993 μετά από δαπάνη 75 (εβδομήντα πέντε) εκατομμυρίων λιρών (το συνολικό κόστος της αποτυχίας υπολογίζεται στα 480 εκατομμύρια λίρες). Πολλά προβλήματα δημιουργήθηκαν στις αποτυχίες να συμφιλιωθούν οι συγκρουόμενες απαιτήσεις.

Ουσιαστικά η Εξαγωγή των Απαιτήσεων, αναφέρεται στο **ΤΙ** κάνει το σύστημα σε αντίθεση με το στάδιο που ακολουθεί, τη σχεδίαση, που αφορά

το **ΠΩΣ** λειτουργεί το σύστημα. Σκοπός σ' αυτό το στάδιο είναι να διασαφηνίσουμε τον ορισμό ενός όχι καλά ορισμένου προβλήματος που ζητά λύση. Αρχικά αναγνωρίζεται το πρόβλημα κι έπειτα αναζητείται η λύση. Είναι δύσκολο να αναγνωριστούν μελλοντικά προβλήματα που ίσως προκύψουν, γι αυτό καλό θα ήταν να έχουμε κατά νου τύπους προβλημάτων που έχουν ήδη προκύψει στο παρελθόν. Πρωταρχικό μέλημα γι αυτή τη διαδικασία είναι ο προσδιορισμός του μέρους του κόσμου μέσα στο οποίο το νέο σύστημα πρόκειται να δώσει λύσεις, να λειτουργήσει και να έχει τις απαιτούμενες –επιθυμητές αντιδράσεις (domain name).

Βρισκόμαστε στο στάδιο της εξαγωγής των απαιτήσεων, τι είναι όμως απαίτηση;

Σύμφωνα με τον ορισμό της Pfleeger, απαίτηση είναι ένα χαρακτηριστικό του συστήματος ή μια περιγραφή ενός πράγματος που το σύστημα είναι ικανό να κάνει έτσι ώστε να εκπληρώσει το σκοπό του.

Για την καλύτερη κατανόηση και περιγραφή των απαιτήσεων είναι απαραίτητο να αναφέρουμε τους τύπους στους οποίους διαχωρίζονται, τις λειτουργικές και τις μη λειτουργικές απαιτήσεις.

Οι λειτουργικές απαιτήσεις (functional requirements), περιγράφουν μια αλληλεπίδραση μεταξύ του συστήματος και του περιβάλλοντος και διευκρινίζουν το πως το σύστημα θα πρέπει να αντιδράσει σε συγκεκριμένες εισόδους και καταστάσεις. Γενικά οι λειτουργικές απαιτήσεις περιγράφουν τη συμπεριφορά του συστήματος κάτω από ορισμένες συνθήκες.

Οι μη λειτουργικές απαιτήσεις (non-functional requirements), περιγράφουν περιορισμούς που θα πρέπει να ικανοποιούνται από τις υπηρεσίες και λειτουργίες που προσφέρει το σύστημα.

Αυτό, είναι σε αντίθεση με τις λειτουργικές απαιτήσεις που καθορίζουν την συμπεριφορά ή τις λειτουργίες του συστήματος. Σε γενικές γραμμές, οι λειτουργικές απαιτήσεις ορίζουν τι πρόκειται να κάνει ένα σύστημα, ενώ οι μη-λειτουργικές απαιτήσεις ορίζουν το πώς ένα σύστημα είναι. Οι μη-λειτουργικές απαιτήσεις ονομάζονται συχνά ιδιότητες του συστήματος. Άλλοι όροι για τις μη-λειτουργικές απαιτήσεις είναι "περιορισμοί", "χαρακτηριστικά ποιότητας", "οι στόχοι της ποιότητας". Οι μη-λειτουργικές απαιτήσεις, για παράδειγμα μπορεί να περιλαμβάνουν χρονικούς περιορισμούς, περιορισμούς στη διαδικασία ανάπτυξης.

Και οι δύο τύποι απαιτήσεων εξάγονται από τον πελάτη με τυπικό, προσεκτικό τρόπο.

Για να εξασφαλιστεί η άριστη ποιότητα των απαιτήσεων θα πρέπει να ελέγξουμε αν διαθέτουν κάποια χαρακτηριστικά, τα οποία και θα αναφέρουμε επιγραμματικά. Οι απαιτήσεις λοιπόν πρέπει να είναι,

- **Σωστές**, να είναι καταγεγραμμένες χωρίς λάθη
- **Συνεπείς**, να μην υπάρχουν αλληλοσυγκρουόμενες απαιτήσεις
- **Πλήρεις**, όλες οι λειτουργίες ενός συστήματος πρέπει να περιγράφονται από κάποια απαίτηση. Οι πλήρεις απαιτήσεις χωρίζονται σε
    - **Εξωτερικά πλήρεις**, η περιγραφή περιέχει όλες τις ιδιότητες σχετικά με το περιβάλλον που επιθυμεί ο πελάτης
    - **Εσωτερικά πλήρεις**, δεν υπάρχουν καθόλου αόριστες αναφορές μέσα στις απαιτήσεις
- **Ρεαλιστικές**, να γίνονται πράξη αυτά που ζητάει ο χρήστης από το σύστημα
- **Αναγκαίες**, να σχετίζονται όλες οι απαιτήσεις άμεσα με το τρέχον πρόβλημα
- **Επαληθεύσιμες**, να είναι εφικτή η δημιουργία ελέγχων που αποδεικνύουν ότι οι απαιτήσεις πληρούνται εξ ολοκλήρου
- **Ιχνηλάσιμες**, να ανιχνεύεται το σύνολο των απαιτήσεων μιας λειτουργίας

Αφού ορίσαμε την έννοια της απαίτησης καθώς και τα χαρακτηριστικά των απαιτήσεων, μπορούμε να εμβαθύνουμε στις τεχνικές που χρησιμοποιούμε για την εκμαίευσή τους, στην καταγραφή των προδιαγραφών και τέλος στην επαλήθευση αυτών των προδιαγραφών. Η διαδικασία της Μηχανικής των Απαιτήσεων περιλαμβάνει τα τρία αυτά προαναφερθέντα βήματα, requirements elicitation, specification κα verification αντίστοιχα.

ΠΕΡΙΛΗΨΗ (Abstract)

The term engineering requirements in software systems, we mean the portion of the Software Engineering including the investigation of work needs or appropriate conditions necessary to achieve a new product or modifying an existing one, taking into account the possibly conflicting needs of different stakeholders such as beneficiaries or users.

The Engineering Export requirements are a critical step for the success of a software project. The seriousness of the danger of ignorance of the requirements is that the record of claims made in the early stages of software development process, which is the foundation on which the play is built. A mistake at this stage, if not corrected can cost dearly, and sometimes it is better to fully from the beginning the project. Examples of two examples:

- Performing Rights Society, PROMS project. Abandoned in 1992 after spending 11 (eleven) million pounds. Incomplete implementation of requirements engineering has been a prominent factor. States that fail to present their claims in a form understood and controlled by people outside the field of IT and end the documentation requirements were not easily understood.

- London Stock Exchange TAURUS project. Cancelled in 1993 after spending 75 (seventy five) million pounds (total cost of failure is estimated at 480 million pounds). Many problems were created in the failure to reconcile conflicting demands.

Virtually Extraction of Requirements refers to WHAT makes the system as opposed to the subsequent phase, the design on the POS system operates. The aim at this stage is to clarify the definition of a rather well defined problem calling solution. Originally recognized the problem and then find the solution. It is difficult to identify future problems that may arise, so it would be good to remember types of problems have already arisen in the past. The primary concern for this process is to identify part of the world in which the new system will provide solutions to operate and has the required, desired reactions (domain name).

We currently export requirements, but that claim?

According to the definition of Pfleeger, demand is a feature of the system or a description of something the system is able to do so to fulfill its purpose.

To better understand and describe the requirements necessary to mention the types who are separated, the functional and non functional requirements.

The functional requirements (functional requirements), describe an interaction between the system and the environment and specify how the system should react to particular inputs and situations. General functional requirements describe the system behavior under certain conditions.

Non-functional requirements (non-functional requirements), describe restrictions that must be met by the services and functions offered by the system.

This is in contrast with the functional requirements that define the behavior or functionality of the system. In general, the functional requirements define what is going to make a system, while non-functional requirements define how a system is. Non-functional requirements are often called properties of the system. Other terms for non-functional requirements are "constraints", "quality attributes", "objectives" quality. Non-functional requirements, for example, may include time constraints, constraints in the development process.

Both types of requirements extracted from a typical customer, careful way.

To ensure the high quality requirements will have to check if they have some characteristics, which we will mention briefly. The claims therefore should be,

- **Right**, to be recorded without errors
- **Consistent,** there are no conflicting demands
- **Complete**, all the functions a system must be described by a requirement. The full requirements are divided into

    - **Outside complete**, the description contains all the attributes of the environment that the customer wishes

    - **Inside complete**, there are no reports in the abstract requirements

- **Realistic**, to practice what they are asking the user from the system

- **Necessary**, to relate all claims directly with the current problem

- **Accurate**, to be able to create tests to demonstrate that the requirements are fully met

- **Traceable**, to detect all the requirements of a function

  After defining the meaning of the claim and the nature of the requirements, we can examine the techniques used for the elicitation of the recording standards and finally the verification of specifications. The process of engineering assets comprises the three steps mentioned above, requirements elicitation, specification verification.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ ΚΑΙ ΠΙΝΑΚΩΝ

ΕΙΣΑΓΩΓΗ

Παραδοσιακά, οι συντελεστές παραγωγής περιελάμβαναν το κεφάλαιο, το ανθρώπινο δυναμικό, την γη. Πρόσφατα, στους συντελεστές παραγωγής έχει προστεθεί και η πληροφορία. Χωρίς έγκαιρη και έγκυρη πληροφορία πολλές επιχειρήσεις δεν θα μπορούσαν να λειτουργήσουν.

Τα σύγχρονα πληροφοριακά συστήματα που βασίζονται στον ηλεκτρονικό υπολογιστή (Η/Υ) συλλέγουν, αποθηκεύουν, αναλύουν και

διαχέουν δεδομένα και πληροφορίες. Με τον τρόπο αυτό υποστηρίζουν τις λειτουργίες μίας επιχείρησης και
παρέχουν τις πληροφορίες που χρειάζονται στην διοίκησή της για αποτελεσματικότερες αποφάσεις. Τα πληροφοριακά συστήματα εκτός από τους υπολογιστές περιλαμβάνουν τους ανθρώπους που συλλέγουν και χρησιμοποιούν τις πληροφορίες, τις διαδικασίες που χρησιμοποιούνται για την καταγραφή, την οργάνωση και την χρήση των πληροφοριών, τα μέσα στα οποία καταχωρούνται οι πληροφορίες, κλπ.

Η εισαγωγή των πληροφοριακών συστημάτων σε μία επιχείρηση έχει πολλαπλές επιδράσεις στην επιχείρηση, στους εργαζομένους και στην κοινωνία. Είναι λοιπόν σαφές ότι η μελέτη των πληροφοριακών συστημάτων απαιτεί μία δι-επιστημονική προσέγγιση. Οι επιστημονικοί τομείς που παίζουν σημαντικό ρόλο στην μελέτη των πληροφοριακών συστημάτων είναι η πληροφορική, η κοινωνιολογία, η θεωρία οργάνωσης και συμπεριφοράς, οι πολιτικές επιστήμες, η ψυχολογία, η επιχειρησιακή έρευνα και η γλωσσολογία.

ΚΕΦΑΛΑΙΟ 1^{Ο}
ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

1.1. Δεδομένα και Πληροφορία

Δεδομένα (data) είναι γεγονότα ή παρατηρήσεις που μπορούν να καταγραφούν. Τα δεδομένα στην πραγματικότητα είναι τιμές (μετρήσεις) κάποιων χαρακτηριστικών που ανήκουν σε οντότητες. Για παράδειγμα, αν θεωρήσουμε την οντότητα «πελάτης» μπορούμε να θεωρήσουμε ως χαρακτηριστικά του το όνομά του, την διεύθυνσή του, το τηλέφωνό του, κλπ. Για κάθε πελάτη τα χαρακτηριστικά αυτά έχουν συγκεκριμένες τιμές. (Κόλλιας Γ. 1986, Pressman R.S. 2001)

Τα δεδομένα για να είναι χρήσιμα πρέπει να έχουν τα παρακάτω χαρακτηριστικά τα οποία καθορίζουν την ποιότητά τους:

- *ακριβή* - δηλ. να μην περιέχουν σφάλματα (η μέθοδος συλλογής και εισαγωγής των δεδομένων θα πρέπει να ελέγχει στο μέτρο του δυνατού την ακρίβεια των δεδομένων που συλλέγονται και εισάγονται).
- *πλήρη* - δηλ. πρέπει να υπάρχουν όλα τα δεδομένα που απαιτούνται για την λύση ενός προβλήματος ή για την λήψη μίας απόφασης πρέπει να υπάρχουν
- *σχετικά* - δηλ. τα υπάρχοντα δεδομένα να έχουν σχέση με το πρόβλημα ή την απόφαση που θα ληφθεί.
- *έγκαιρα* - δηλ. να είναι διαθέσιμα όταν τα χρειάζεται η οργάνωση. (Κόλλιας Γ. 1986, Pressman R.S. 2001)

Πληροφορία (information) είναι δεδομένα τα οποία έχουν επεξεργαστεί σε μία μορφή που είναι χρήσιμη για τους τελικούς χρήστες. Η

επεξεργασία αυτή των αρχικών δεδομένων προσθέτει αξία σε αυτά. (Κόλλιας Γ. 1986, Pressman R.S. 2001)

Οι οργανισμοί και οι επιχειρήσεις συλλέγουν δεδομένα, τα αναλύουν για να

δημιουργούν πληροφορίες, διαχέουν τις κατάλληλες πληροφορίες στους κατάλληλους ανθρώπους και λαμβάνουν αποφάσεις βασιζόμενοι στην ερμηνεία της

πληροφορίας αυτής.

Πληροφοριακό Σύστημα (ΠΣ) (information system) είναι ένα σύνολο οντοτήτων το οποίο συλλέγει, αποθηκεύει, αναλύει δεδομένα και διαχέει πληροφορίες. Όπως κάθε σύστημα, το ΠΣ περιέχει εισόδους (δεδομένα, πληροφορίες, εντολές) επεξεργασίες (διαδικασίες, άνθρωποι, εξοπλισμός) και εξόδους (αναφορές, γραφήματα, υπολογισμοί). Ορισμένες από τις οντότητες που απαρτίζουν ένα Π.Σ. είναι κατασκευές (τεχνουργήματα) όπως το μολύβι και το χαρτί που μπορεί να χρησιμοποιηθούν για την καταγραφή των δεδομένων. Ωστόσο, όλα τα Π.Σ. χρειάζονται ανθρώπους που θα σχεδιάσουν, θα κατασκευάσουν και θα χρησιμοποιήσουν τα τεχνουργήματα. (Κόλλιας Γ. 1986, Pressman R.S. 2001)

Ένα Π.Σ. μπορεί να είναι είτε χειρωνακτικό είτε βασισμένο σε ηλεκτρονικό υπολογιστή. Ένα Π.Σ. που βασίζεταιστον ηλεκτρονικό υπολογιστή χρησιμοποιεί την τεχνολογία του υπολογιστή για να εκπληρώσει έναν ή περισσότερους από τους στόχους του.

Επιπλέον, ένα Π.Σ. μπορεί να είναι τυπικό ή άτυπο. Τα τυπικά συστήματα λειτουργούν βάσει διαδικασιών, με προαποφασισμένες εισόδους και εξόδους. Τα άτυπα Π.Σ. από την άλλη μεριά δεν ακολουθούν προσχεδιασμένες διαδικασίες συλλογής, αποθήκευσης και διάδοσης των πληροφοριών. Οι εργαζόμενοι σχηματίζουν τέτοια άτυπα Π.Σ. όταν χρειάζονται πληροφορίες που δεν παρέχονται από τα υπάρχοντα τυπικά Π.Σ. Άτυπα Π.Σ. είναι λόγου χάρη η ανταλλαγή μηνυμάτων μεταξύ φίλων με το ηλεκτρονικό ταχυδρομείο. (Κόλλιας Γ. 1986, Pressman R.S. 2001)

1.2. Ιστορική εξέλιξη των πληροφοριακών συστημάτων

Οι πρώτες εφαρμογές των υπολογιστών στις επιχειρήσεις (μισθοδοσία, τιμολόγηση) απαιτούσαν επαναλαμβανόμενους υπολογισμούς σε μεγάλο αριθμό δεδομένων και εμφανίστηκαν την δεκαετία του 50. Παράλληλα με την εξέλιξη του υλικού που είχε σαν αποτέλεσμα την δημιουργία φθηνότερων, καλύτερων και φιλικότερων υπολογιστών, οι επιχειρήσεις διαπίστωσαν τις ωφέλειες που προκύπτουν από τη χρήση της τεχνολογίας και την δυναμική που μπορεί να προσδώσει. (Κόλλιας Γ. 1986, Pressman R.S. 2001)

Στην δεκαετία του 60 άρχισαν να αναπτύσσονται συστήματα που είχαν την
δυνατότητα να διαχειριστούν δεδομένα σχετικά με την λήψη αποφάσεων (πληροφοριακό σύστημα διοίκησης). Τα συστήματα αυτά χαρακτηρίζονται κυρίως από την δυνατότητα να παρέχουν περιοδικές αναφορές. Στην αρχή, τα συστήματα αυτά είχαν κυρίως ιστορικό χαρακτήρα (έδιναν δηλ. έμφαση κυρίως στο τι έχει συμβεί), ενώ αργότερα, χρησιμοποιήθηκαν για την πρόβλεψη τάσεων και την υποστήριξη αποφάσεων ρουτίνας.

Στις αρχές της δεκαετίας του 70 τα υπολογιστικά συστήματα χρησιμοποίησαν το τηλεπικοινωνιακό δίκτυο (π.χ. συστήματα κράτησης θέσεων σε πτήσεις). Η χρήση αυτή των επικοινωνιών επεκτάθηκε αργότερα και συνετέλεσε μαζί με την διάδοση των συστημάτων επεξεργασίας κειμένου στην εμφάνιση των συστημάτων αυτοματισμού γραφείου. Την ίδια εποχή εμφανίσθηκε η έννοια του συστήματος στήριξης αποφάσεων με βασικό στόχο την υποστήριξη πολύπλοκων ημι-δομημένων αποφάσεων. Ωστόσο, το κόστος ανάπτυξης των συστημάτων αυτών εξακολουθούσε να είναι υψηλό. (Κόλλιας Γ. 1986, Pressman R.S. 2001)

Η κατάσταση άλλαξε με την εμφάνιση των μικροϋπολογιστών, στις αρχές της δεκαετίας του 80. Το φθηνό κόστος των συστημάτων αυτών καθώς και η ευκολία χρήσης και προγραμματισμού τους, επέτρεψαν σε πολλούς χρήστες να δημιουργήσουν τα δικά τους συστήματα.

Στα μέσα της δεκαετίας του 80 δημιουργήθηκε ένας νέος τομέας: η τεχνητή νοημοσύνη. Νέα έξυπνα συστήματα αναπτύχθηκαν, με περισσότερο δημοφιλή τα έμπειρα συστήματα. Τα συμβουλευτικά αυτά συστήματα είναι τελείως διαφορετικά από τα συστήματα επεξεργασίας συναλλαγών (που δίνουν έμφαση στα δεδομένα) και από τα συστήματα διοίκησης και υποστήριξης αποφάσεων (με έμφαση στην επεξεργασία πληροφοριών). Στα τέλη της δεκαετίας του 80 δημιουργήθηκαν τα συστήματα υποστήριξης ομάδων για την υποστήριξη των εργαζομένων σε ομάδες.
(Κόλλιας Γ. 1986, Pressman R.S. 2001)

1.3. Δραστηριότητες Πληροφορικού Συστήματος (Π.Σ.)

Στις δραστηριότητες ενός Π.Σ. περιλαμβάνονται :

- Συλλογή δεδομένων,

τα δεδομένα συλλέγονται από διάφορες πηγές:

- *από εσωτερικές πηγές* (internal sources) - π.χ. δεδομένα σχετικά με τις παραγγελίες που είναι έτοιμες προς αποστολή.
- *από εξωτερικές πηγές* (external sources) - π.χ. δεδομένα σχετικά με τις παραγγελίες των πελατών
- *από το περιβάλλον* - π.χ. δεδομένα που συλλέγονται από εταιρίες δημοσκοπήσεων

Τα δεδομένα καταγράφονται σε κάποιο μέσο (συνήθως χαρτί) ή εισάγονται κατευθείαν στο σύστημα. Τα δεδομένα ελέγχονται για να εξασφαλισθεί ότι καταγράφηκαν σωστά. (Κόλλιας Γ. 1986, Pressman R.S. 2001)

- **Αποθήκευση δεδομένων**

Με την αποθήκευση τα δεδομένα φυλάσσονται με έναν οργανωμένο τρόπο για μελλοντική χρήση.

- **Επεξεργασία δεδομένων**

Η επεξεργασία των δεδομένων περιλαμβάνει υπολογισμούς, συγκρίσεις, ταξινομήσεις και κατηγοριοποιήσεις. Για παράδειγμα, τα δεδομένα που αφορούν μία αγορά ενός πελάτη μπορεί να:

- προστεθούν στο σύνολο των αγορών του πελάτη
- συγκριθούν με το ποσό που καθιστά τον πελάτη δικαιούχο της έκπτωσης
- ταξινομηθούν σύμφωνα με τους κωδικούς των προϊόντων που αγόρασε ο πελάτης
- ταξινομηθούν σε κατηγορίες προϊόντων (πχ τρόφιμα, απορρυπαντικά.

- **Διάδοση πληροφοριών**

Ο στόχος ενός Π.Σ. είναι η διάδοση πληροφοριών. Η πληροφορία μπορεί να διαδοθεί σε διάφορες μορφές (μηνύματα, φόρμες, αναφορές, λίστες, γραφήματα, κλπ) (Κόλλιας Γ. 1986, Pressman R.S. 2001)

1.4. Πόροι Πληροφοριακού Συστήματος (Π.Σ.)

Οι βασικοί πόροι ενός Π.Σ. είναι:

- ανθρώπινοι πόροι (τελικοί χρήστες, ειδικοί της πληροφορικής),
- υλικοί πόροι (το σύνολο συσκευών το οποίο χρησιμοποιείται για την εισαγωγή την επεξεργασία και την αποθήκευση των δεδομένων),
- πόροι λογισμικού (προγράμματα και διαδικασίες) και
- πόροι δεδομένων (βάσεις δεδομένων, βάσεις μοντέλων και βάσεις γνώσεων).

(Κόλλιας Γ. 1986, Pressman R.S. 2001)

- **Ανθρώπινοι πόροι**

Όλα τα Π.Σ. περιλαμβάνουν ανθρώπους και για τον λόγο αυτό τα Π.Σ. είναι κοινωνικά συστήματα. Οι άνθρωποι που συμμετέχουν σε ένα Π.Σ. είναι είτε τελικοί χρήστες είτε ειδικοί της πληροφορικής.

- Οι *τελικοί χρήστες* είναι αυτοί οι οποίοι χρησιμοποιούν άμεσα ή έμμεσα (την πληροφορία που αυτό παράγει) ένα Π.Σ.. Οι τελικοί χρήστες μπορεί να είναι μηχανικοί, υπάλληλοι, λογιστές, διοικητικοί, κλπ.
- Οι *ειδικοί της πληροφορικής* αναπτύσσουν και χειρίζονται τα Π.Σ. Στους ειδικούς πληροφορικής εντάσσονται οι αναλυτές συστημάτων, οι προγραμματιστές, χειριστές ηλεκτρονικών υπολογιστών, κλπ.

- **Υλικοί πόροι**

Στους υλικούς πόρους ανήκουν:

- το *υλικό* (hardware) δηλ. τα συστήματα ηλεκτρονικών υπολογιστών τα οποία αποτελούνται από κεντρική μονάδα επεξεργασίας, τα περιφερειακά (πληκτρολόγιο, οθόνη, εκτυπωτής, κλπ) και τα δίκτυα τηλεπικοινωνιών
- τα *μέσα που χρησιμοποιούνται για την αποθήκευση δεδομένων* (χαρτί, μαγνητικές ταινίες, σκληροί δίσκοι, κλπ).

- **Πόροι λογισμικού**

Ο όρος αυτός είναι πολύ γενικός και περιλαμβάνει:

- το λογ*ισμικό συστήματος* το οποίο ελέγχει και υποστηρίζει τις λειτουργίες του ηλεκτρονικού υπολογιστή λ.χ τα λειτουργικά συστήματα
- το *λογισμικό εφαρμογών* το οποίο παρέχει στον τελικό χρήστη την δυνατότητα επεξεργασίας ενός συγκεκριμένου προβλήματος (λχ προγράμματα ανάλυσης πωλήσεων, προγράμματα μισθοδοσίας, επεξεργαστές κειμένου).

- τις *διαδικασίες* δηλαδή οδηγίες προς τους ανθρώπους που χρησιμοποιούν το Π.Σ. λ.χ. οδηγίες συμπλήρωσης μίας φόρμας, ή οδηγίες χρήσης ενός προγράμματος.

(Κόλλιας Γ. 1986, Pressman R.S. 2001)

- **Πόροι δεδομένων**

Τα δεδομένα αποτελούν σημαντικό πόρο για έναν οργανισμό. Για τον λόγο αυτό η διαχείριση των δεδομένων πρέπει να γίνεται με τρόπο που να επωφελούνται όλοι οι τελικοί χρήστες. Τα δεδομένα μπορούν να πάρουν διάφορε μορφές (κείμενο, εικόνα, ήχος) και οργανώνονται σε:

- *Βάσεις δεδομένων* που αποθηκεύουν και διαχειρίζονται οργανωμένα δεδομένα,
- *Βάσεις προτύπων* που αποθηκεύουνμαθηματικά και λογικά πρότυπα τα οποία περιέχουν σχέσεις, υπολογισμούς και αναλυτικές τεχνικές και τέλος
- *Βάσεις γνώσεων* που αποθηκεύουν γεγονότα και κανόνες για διάφορα προβλήματα.

(Κόλλιας Γ. 1986, Pressman R.S. 2001)

1.5. Τύποι Πληροφοριακών Συστημάτων (Π.Σ.)

Για να διευκολυνθεί η μελέτη των Π.Σ. έχουν προταθεί διάφοροι τρόποι κατηγοριοποίησής τους. Οι κυριότεροι είναι ανάλογα με:

- το υποσύστημα το οποίο υποστηρίζουν
- την επιχειρηματική δραστηριότητα που υποστηρίζουν
- το είδος της υποστήριξης που παρέχουν
- ανάλογα με την αρχιτεκτονική τους

1.5.1 Τύποι Π.Σ. ανάλογα με το υποσύστημα που υποστηρίζουν

Οι οργανισμοί αποτελούνται από μικρότερες οντότητες (υποσυστήματα) όπως για παράδειγμα από διευθύνσεις, τμήματα ή ομάδες. Οι περισσότεροι οργανισμοί έχουν τμήμα προσωπικού, τμήμα παραγωγής, λογιστικό τμήμα κλπ. Κάθε ένα από τα τμήματα αυτά αναφέρει σε μία προϊστάμενη αρχή. Η πλειονότητα των οργανισμών σήμερα είναι δομημένη σύμφωνα με το τρόπο αυτό (που είναι γνωστός ως ιεραρχική δομή). (Κόλλιας Γ. 1986, Pressman R.S. 2001)

Ένας τρόπος οργάνωσης των Π.Σ. είναι να δομηθούν σύμφωνα με την ιεραρχική δομή του οργανισμού. Έτσι, μπορεί να δημιουργηθούν Π.Σ. για διευθύνσεις, τμήματα, ομάδες ή ακόμη και για συγκεκριμένους εργαζόμενους. Τα συστήματα αυτά μπορεί να είναι είτε αυτόνομα ή συνδεδεμένα μεταξύ τους.

Πληροφοριακά συστήματα σύμφωνα με την ιεραρχική δομή είναι:

- *Π.Σ. για τα τμήματα της επιχείρησης* - συχνά, μία επιχείρηση χρησιμοποιεί αρκετές εφαρμογές (προγράμματα) σε μία λειτουργική περιοχή. Οι εφαρμογές αυτές μπορεί να έχουν κάποια κοινά σημεία, μπορεί όμως και όχι. Το σύνολο των εφαρμογών που χρησιμοποιείται από το τμήμα προσωπικού, αναφέρεται ως πληροφοριακό σύστημα προσωπικού (παρόλο που αποτελείται από επιμέρους προγράμματα). Για παράδειγμα το τμήμα προσωπικού, μπορεί να χρησιμοποιεί ένα πρόγραμμα για την παρακολούθηση των αιτήσεων πρόσληψης και άλλο πρόγραμμα για την παρακολούθηση των απουσιών του προσωπικού.
- *Π.Σ. για όλη την επιχείρηση* - τα Π.Σ. για τα τμήματα της επιχείρησης συνήθως έχουν σχέση με κάποια δραστηριότητα. Μπορούμε να μιλήσουμε για ένα σύνολο εφαρμογών που υποστηρίζει αρκετές (ή όλες) τις δραστηριότητες της επιχείρησης. Ένα τέτοιο Π.Σ. υποστηρίζει όλη την επιχείρηση.

- *διεπιχειρησιακά Π.Σ.* - είναι σύνθετα Π.Σ. που περιλαμβάνουν αρκετούς οργανισμούς. Για παράδειγμα, το παγκόσμιο σύστημα κράτησης θέσεων σε πτήσεις αποτελείται από τα συστήματα που ανήκουν σε διαφορετικές αεροπορικές εταιρίες.

Τύποι Π.Σ. ανάλογα με την επιχειρηματική δραστηριότητα που υποστηρίζουν είναι :

- το λογιστικό,
- το οικονομικό,
- το Π.Σ. παραγωγής,
- το Π.Σ. προώθησης πωλήσεων και
- το Π.Σ. προσωπικού.

(Κόλλιας Γ. 1986, Pressman R.S. 2001)

ΚΕΦΑΛΑΙΟ 2^{Ο}
ΛΟΓΙΣΜΙΚΟ ΓΙΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Η ανάπτυξη του λογισμικού ενός Π.Σ. γίνεται με την βοήθεια εργαλείων που υποστηρίζουν την ανάπτυξη και τον έλεγχο του κώδικα. Ένα πλήθος εργαλείων που καλύπτουν ένα ευρύ φάσμα μεθοδολογικών προσεγγίσεων είναι διαθέσιμο στον προγραμματιστή. (Σκορδαλάκης, Μ. 1991, Pfleeger, Shari Lawrence 2003)

2.1 Γλώσσες Προγραμματισμού

Οι γλώσσες προγραμματισμού είναι τεχνητές γλώσσες με αυστηρά καθορισμένο συντακτικό που χρησιμοποιούνται για την ανάπτυξη λογισμικού. Με βάση το εύρος των προβλημάτων τα οποία μπορούν να αντιμετωπίσουν, οι γλώσσες
προγραμματισμού χαρακτηρίζονται ως γενικές ή εξειδικευμένες, ενώ με βάση την ιστορική τους εξέλιξη ταξινομούνται σε "γενιές".

- *γλώσσες 1ης γενεάς ή γλώσσες μηχανής (machine languages)*- βασίζονται στον δυαδικό κώδικα, είναι άμεσα κατανοητές από τον ηλεκτρονικό υπολογιστή και εξαρτώνται από την συγκεκριμένη μηχανή (machine-dependent) δηλ. προγράμματα που γράφονται σε έναν υπολογιστή δεν είναι κατανοητά από άλλον.
- *γλώσσες 2ης γενεάς ή συμβολικές γλώσσες (assembly languages)* - αναπτύχθηκαν την δεκαετία του '50, απαιτούν

μεταφραστές για την μετατροπή τους σε γλώσσα μηχανής, είναι ευκολότερη η εκμάθηση και απομνημόνευσή τους

- *γλώσσες 3ης γενεάς ή διαδικαστικές ή υψηλού επιπέδου γλώσσες (procedural languages)* - αναπτύχθηκαν από τα τέλη της δεκαετίας του '50, χρησιμοποιούν εκτενώς σύμβολα, υιοθετούν την έννοια της υπορουτίνας, χρησιμοποιούνται για την ανάπτυξη συστημάτων υποστήριξης αποφάσεων παρά το γεγονός ότι δεν περιέχουν ευκολίες για την ανάπτυξη αυτών. Για μεγάλα συστήματα υποστήριξης αποφάσεων έχουν χρησιμοποιηθεί κυρίως η APL (ιδιαίτερα κατάλληλη για μαθηματικά προβλήματα όπως αντιστροφή πινάκων), η PL/1 και η Pascal (χρήσιμη για διαχείριση δεδομένων)

Οι τρεις αυτές κατηγορίες γλωσσών βασίζονται στην εργασία του μαθηματικού von Neumann και ο οποίος πρότεινε (πέραν των άλλων) την έννοια του μετρητή προγράμματος που καθορίζει την επόμενη προς εκτέλεση εντολή ενός προγράμματος. Αυτό σημαίνει ότι για την ορθή ανάπτυξη προγραμμάτων στις γλώσσες αυτές, ο προγραμματιστής είναι υποχρεωμένος να μιμηθεί νοητά την σειρά
εκτέλεσης των εντολών ενός προγράμματος. (Σκορδαλάκης, Μ. 1991, Pfleeger, Shari Lawrence 2003)

- *γλώσσες 4ης γενεάς ή μη διαδικαστικές γλώσσες (non-procedural languages)* – η βασική ιδέα μίας μη διαδικαστικής γλώσσας είναι να μεταφερθεί η ευθύνη της ροής του προγράμματος από τον προγραμματιστή στο λογισμικό. Με τις μη- διαδικαστικές γλώσσες ο προγραμματιστής προσδιορίζει *τι* θέλει να υπολογίσει ο υπολογιστής και όχι τον τρόπο (το πώς) που θα γίνει αυτό. Έτσι, στις γλώσσες αυτές δεν απαιτείται από τον προγραμματιστή να διευκρινίσει χαμηλού επιπέδου λεπτομέρειες όπως για παράδειγμα ο τρόπος με τον οποίο θα ταξινομηθούν τα αποτελέσματα ή ο ακριβής τρόπος

αναπαράστασης των δεδομένων. Οι γλώσσες αυτές χρησιμοποιούνται κυρίως για την ανάπτυξη των συστημάτων υποστήριξης αποφάσεων.

Οι γλώσσες 4ης γενιάς αυξάνουν θεαματικά την παραγωγικότητα του προγραμματιστή (τουλάχιστο 5:1, σε ορισμένες ακραίες περιπτώσεις παρατηρήθηκε
αύξηση έως 300:1) και επιπλέον μπορούν να χρησιμοποιηθούν και από τους τελικούς χρήστες. Οι γλώσσες 4ης γενιάς σχεδιάστηκαν έχοντας υπόψη το κόστος ανάπτυξης λογισμικού. Έτσι, οι γλώσσες 4ης γενιάς μπορούν να χρησιμοποιούνται από χρήστες με περιορισμένες γνώσεις προγραμματισμού εύκολα μετά από σύντομη εκπαίδευσης. (Σκορδαλάκης, Μ. 1991, Pfleeger, Shari Lawrence 2003)

Επιπλέον, οι γλώσσες αυτές, ελαχιστοποιούν το κόστος αποσφαλμάτωσης και συντήρησης των προγραμμάτων.

- *γλώσσες 5ης γενιάς* - είναι συμβολικές γλώσσες που παρέχουν αποτελεσματικούς τρόπους αναπαράστασης αντικειμένων και μεθόδων που χρησιμοποιούνται στην τεχνητή νοημοσύνη.

Οι **εξειδικευμένες γλώσσες** επιτρέπουν στον προγραμματιστή να περιγράψει τα χαρακτηριστικά ενός προβλήματος το οποίο πρέπει να λυθεί. Έχουν αναπτυχθεί πολλές εξειδικευμένες γλώσσες για διάφορες οικογένειες προβλημάτων. Για παράδειγμα η γλώσσα GPSS μπορεί να υιοθετηθεί για την δημιουργία προτύπων προσομοίωσης και θα μπορούσε να χρησιμοποιηθεί για την ανάπτυξη ενός μεγάλου συστήματος υποστήριξης αποφάσεων.

Για την διευκόλυνση του στόχου τους, στις εξειδικευμένες γλώσσες προγραμματισμού εμπεριέχονται έννοιες και λειτουργίες που δεν είναι διαθέσιμες στις γενικές γλώσσες προγραμματισμού. Η γλώσσα GPPS, για παράδειγμα, είναι σε θέση να υποστηρίξει την προσομοίωση διακριτών συστημάτων. Η γλώσσα GPSS (α) υποστηρίζει την έννοια του "χρόνου" και (β) έχει την δυνατότητα να διαχειρίζεται την "χρονική αλληλουχία" διαφόρων γεγονότων. (Σκορδαλάκης, Μ. 1991, Pfleeger, Shari Lawrence 2003)

2.2 Γεννήτριες

Η ανάπτυξη λογισμικού με την βοήθεια των γλωσσών προγραμματισμού έχει μεγάλο κόστος και απαιτεί την απασχόληση εξειδικευμένων προγραμματιστών για μεγάλα χρονικά διαστήματα. Για να διευκολυνθεί η διαδικασία ανάπτυξης λογισμικού δημιουργήθηκαν εργαλεία με στόχο την αυτοματοποίηση της ανάπτυξης ορισμένων τμημάτων του κώδικα. Τα εργαλεία αυτά αποσκοπούν στην αύξηση της παραγωγικότητας των προγραμματιστών αλλά και στην μετακίνηση ορισμένων σημείων της κωδικοποίησης από τους προγραμματιστές προς τους αναλυτές, τους σχεδιαστές, και προς τους τελικούς χρήστες των Π.Σ. Τα περισσότερα από τα εργαλεία αυτά είναι γνωστά ως **γεννήτριες** γιατί δέχονται την περιγραφή ενός τμήματος του Π.Σ. και αναπτύσσουν τον κώδικα του προγράμματος που αντιστοιχεί στο τμήμα αυτό αυτόματα. Οι περισσότερο διαδεδομένες γεννήτριες εξειδικεύονται στην αυτοματοποίηση του κώδικα που σχετίζεται με την επικοινωνία ανθρώπου- υπολογιστή. Έτσι έχουν αναπτυχθεί γεννήτριες αναφορών και γεννήτριες οθονών
που έχουν ως στόχο την αυτοματοποίηση των αναφορών που παράγει ένα Π.Σ. και
τις οθόνες (φόρμες) που χρησιμοποιούνται από το Π.Σ. Ορισμένες φορές οι επιμέρους γεννήτριες υπάρχουν στην μορφή ενός ολοκληρωμένου πακέτου που είναι γνωστό ως γεννήτρια εφαρμογών. (Σκορδαλάκης, Μ. 1991, Pfleeger, Shari Lawrence 2003)

2.2.1. Γεννήτριες αναφορών

Η ανάπτυξη κώδικα για την προετοιμασία αναφορών παρό παρόλο που ακολουθεί συγκεκριμένους κανόνες εμπεριέχει αρκετές λεπτομέρειες (υπολογισμός μερικών αθροισμάτων, αλλαγή σελίδων, σελιδοποίηση, ολικά

αθροίσματα) που επιμηκύνουν τον χρόνο ανάπτυξης. Χρησιμοποιώντας μία γεννήτρια αναφορών (report generator) ένας προγραμματιστής μπορεί να ορίσει τη μορφή της αναφοράς προσδιορίζοντας τα περιεχόμενα της αναφοράς. Οι γεννήτριες αναφορών έχουν πρόσβαση σε αρχεία ή σε βάσεις δεδομένων από τις οποίες εξάγουν δεδομένα τα οποία μορφοποιούν σε αναφορές. Ο κώδικας για την υλοποίηση των αντίστοιχων υπορουτινών δημιουργείται αυτόματα από την γεννήτρια. (Σκορδαλάκης, Μ. 1991, Pfleeger, Shari Lawrence 2003)

2.2.2. Γεννήτριες οθονών

Η ανάπτυξη του κώδικα για την υλοποίηση της επικοινωνίας ανάμεσα στο χρήστη και τον υπολογιστή είναι μία διαδικασία που έχει αρκετά κοινά χαρακτηριστικά με την ανάπτυξη των αναφορών. Οι γεννήτριες αναφορών (screen generator) είναι προγράμματα τα οποία επιτρέπουν την εύκολη και γρήγορη ανάπτυξη του interface ενός Π.Σ., προσδιορίζοντας τα περιεχόμενα της κάθε οθόνης, χωρίς την ανάγκη προγραμματισμού.

Μία άλλη κατηγορία γεννητριών είναι οι **γεννήτριες προγραμμάτων** οι οποίες δέχονται ως είσοδο την περιγραφή ενός συστήματος σε μορφή που είναι εύκολο να δοθεί από τον χρήστη και παράγουν αυτόματα τον κώδικα που αντιστοιχεί στο σύστημα. Μία οικογένεια τέτοιων γεννητριών προγραμμάτων χρησιμοποιείται για την προσομοίωση διακριτών συστημάτων. Οι γεννήτριες αυτές προγραμμάτων αναγνωρίζουν συστήματα τα οποία είναι δυνατόν να περιγραφούν διαγραμματικά (με τα διαγράμματα κύκλου ενεργειών) και παράγουν αυτόματα τον κώδικα που αντιστοιχεί στο διάγραμμα που δόθηκε από τον χρήστη. (Σκορδαλάκης, Μ. 1991, Pfleeger, Shari Lawrence 2003)

2.2.3. Γλώσσες αναζητήσεων

Οι γλώσσες αναζητήσεων επιτρέπουν την εύκολη επικοινωνία του χρήστη με τον υπολογιστή κυρίως για την αναζήτηση δεδομένων που φυλάσσονται σε βάσεις δεδομένων.

Βάση δεδομένων είναι μία συλλογή από σχετιζόμενα δεδομένα.

2.3 Σύστημα Διαχείρισης Βάσεων Δεδομένων

Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ, Database Management System) είναι ένα σύνολο προγραμμάτων που είναι υπεύθυνο για την δημιουργία και συντήρηση βάσεων δεδομένων. Το σύστημα διαχείρισης βάσεων δεδομένων ενός Π.Σ. προσφέρει δυνατότητες για την αποθήκευση, την ανάκτηση (αναζήτηση) και τον έλεγχο των δεδομένων που χρειάζονται για την λήψη αποφάσεων.

Τα ΣΔΒΔ ανάλογα με τον τρόπο που οργανώνουν τα δεδομένα στη βάση δεδομένων κατηγοριοποιούνται σε ιεραρχικά (hierarchical), δικτυωτά (network), σχεσιακά (relational) και αντικειμενοστραφή (object-oriented). Τα περισσότερο δημοφιλή ΣΔΒΔ είναι αυτά που διαχειρίζονται σχεσιακές βάσεις δεδομένων. Τα σχεσιακά ΣΔΒΔ επιβάλλουν την οργάνωση των δεδομένων σε πίνακες (tables) - δηλ. σε γραμμές και στήλες. Κάθε πίνακας έχει ένα όνομα, ενώ κάθε στήλη έχει ένα όνομα και έναν τύπο δεδομένων.

Λεξικό Δεδομένων είναι ένας κατάλογος όλων των δεδομένων που περιέχονται στην βάση δεδομένων. Εκτός από τον ορισμό των δεδομένων μπορεί να περιέχει την περιγραφή και την πηγή τους.

Μία άλλη κατηγορία γεννητριών είναι οι **γεννήτριες προγραμμάτων** οι οποίες δέχονται ως είσοδο την περιγραφή ενός συστήματος σε μορφή πουν είναι εύκολο να δοθεί από τον χρήστη και παράγουν αυτόματα τον κώδικα που αντιστοιχεί στο σύστημα. Μία οικογένεια τέτοιων γεννητριών προγραμμάτων χρησιμοποιείται για την προσομοίωση διακριτών συστημάτων. Οι γεννήτριες αυτές προγραμμάτων αναγνωρίζουν συστήματα

τα οποία είναι δυνατόν να περιγραφούν διαγραμματικά (με τα διαγράμματα κύκλου ενεργειών) και παράγουν αυτόματα τον κώδικα που αντιστοιχεί στο διάγραμμα που δόθηκε από τον χρήστη.

ΚΕΦΑΛΑΙΟ 3^{Ο}
ΥΠΟΛΟΓΙΣΤΕΣ ΚΑΙ ΛΟΓΙΣΜΙΚΟ

Ένα από τα σημαντικότερα γεγονότα που σηματοδότησαν τον αιώνα που πέρασε ήταν η εφεύρεση του Ηλεκτρονικού Υπολογιστή (Η/Υ).

Με τη βοήθεια του Η/Υ έγινε δυνατή η αυτοματοποίηση της εκτέλεσης πολλών κουραστικών, ανιαρών και επιρρεπών σε λάθη εργασιών, καθώς και η εκτέλεση άλλων, η οποία στο παρελθόν ήταν πρακτικά αδύνατη. Από την εποχή που κατασκευάστηκαν οι πρώτοι Η/Υ μέχρι σήμερα σημειώθηκε τεράστια βελτίωση των χαρακτηριστικών και των δυνατοτήτων τους. Κανένα άλλο ανθρώπινο κατασκεύασμα δε σημείωσε τόσο σημαντική πρόοδο σε τόσο μικρό χρονικό διάστημα. Ένα από τα πρακτικά αποτελέσματα αυτής της εξέλιξης ήταν ότι οι Η/Υ
έγιναν προσιτοί σε μεγάλες μάζες ανθρώπων και αναφαίρετο εργαλείο της καθημερινής επαγγελματικής αλλά και ιδιωτικής ζωής για πολλούς από αυτούς. Παράλληλα, έγινε δυνατή η ενσωμάτωση Η/Υ σε πάρα πολλές συσκευές καθημερινής χρήσης, χωρίς αυτό να είναι πάντα αντιληπτό από τους ίδιους τους χρήστες. Σήμερα, σε πολλές από τις καθημερινές μας εργασίες χρησιμοποιούμε Η/Υ χωρίς να το γνωρίζουμε, ενώ συχνά η ίδια μας η ζωή εξαρτάται από Η/Υ (υγεία, μέσα μεταφοράς, υπηρεσίες όπως έλεγχος οδικής και εναέριας κυκλοφορίας κ.ά.). (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

Η σημερινή εποχή μπορεί να χαρακτηριστεί ως μεταβατική σε μια νέα κατάσταση στην οποίαόλοι οι Η/Υ θα είναι διασυνδεδεμένοι μέσω δικτύων και θα εκτελούν σύνθετες εργασίες. Πολλές από τις σύγχρονες δικτυακές εφαρμογές μπορούν να μεταβάλουν κρίσιμες πλευρές του πολιτισμού μας, όπως την επικοινωνία, την εκπαίδευση και την κατάρτιση, αλλά και αυτή την ίδια τη λειτουργία των δημοκρατικών πολιτευμάτων. Η νέα κατάσταση που διαμορφώνεται αναφέρεται ως κοινωνία της πληροφορίας (information society) και έχουμε ήδη εισέλθει στο εξελικτικό της στάδιο με το Internet και τις περί αυτό εφαρμογές να παίζουν πρωταγωνιστικό ρόλο στη διαδικασία. Όλες αυτές οι εξελίξεις γίνονται δυνατές χάρη στην ύπαρξη και λειτουργία ενός συνόλου πολύπλοκων εφαρμογών λογισμικού. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

Η εξάπλωση του ηλεκτρονικού υπολογιστή σε ολοένα και περισσότερες πλευρές της ανθρώπινης ζωής δε θα ήταν δυνατή χωρίς τη

χρήση λογισμικού. Ο ηλεκτρονικός υπολογιστής ως συσκευή μπορεί μόνο να εκτελέσει ορισμένες πολύ απλές λειτουργίες με πάρα πολύ υψηλή ταχύτητα, όμως με τρόπο ιδιαίτερα δυσπρόσιτο στον άνθρωπο. Το λογισμικό είναι εκείνο που καθιστά χρήσιμη και αποδίδει στοιχεία «συμπεριφοράς» στη συσκευή Η/Υ. Ο άνθρωπος δεν αξιοποιεί τον ηλεκτρονικό υπολογιστή άμεσα ως συσκευή, αλλά μόνο μέσω του λογισμικού.

Έχοντας κατά νου τα παραπάνω, δεν είναι εύκολο να δοθεί ένας πλήρης και καθολικά αποδεκτός ορισμός της έννοιας «λογισμικό». Η πρακτική αξία, αλλά και η διαχρονικότητα ενός θεωρητικού ορισμού μπορεί να αμφισβητηθεί σχετικά εύκολα. Μια ικανοποιητική προσέγγιση είναι ο ορισμός του λογισμικού ως ακολούθως, Λογισμικό είναι :

(1) εντολές (προγράμματα ηλεκτρονικού υπολογιστή) οι οποίες όταν εκτελούνται επιτυγχάνουν επιθυμητά αποτελέσματα και επιδόσεις,

(2) δομές δεδομένων που επιτρέπουν σε προγράμματα να διαχειριστούν με επάρκεια πληροφορίες και

(3) κείμενα, διαγράμματα κτλ. που περιγράφουν τη λειτουργία και χρήση των προγραμμάτων. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

3.1. Τεχνικές κατασκευές και Λογισμικό

Το λογισμικό είναι ένα πολύπλοκο τεχνικό κατασκεύασμα, το οποίο δεν έχει αυτοτελή υπόσταση, παρά μόνο όταν χρησιμοποιείται για να καθοδηγήσει έναν ηλεκτρονικό υπολογιστή στην πραγματοποίηση συγκεκριμένων λειτουργιών. Παρά τις αρκετές ομοιότητες που μπορεί κανείς να αναζητά συχνά μεταξύ λογισμικού και λοιπών τεχνικών κατασκευών, υπάρχουν και σημαντικές διαφορές. Η πρώτη είναι η μη απτή φύση του λογισμικού. Μια τεχνική κατασκευή είναι ορατή και απτή, ενώ το λογισμικό δεν είναι αυτό καθεαυτό «ορατό». Μόνο τα αποτελέσματα της χρήσης του μπορούν να είναι αντιληπτά. Η δομή του λογισμικού, τόσο σε μικροσκοπικό όσο και σε

μακροσκοπικό επίπεδο, είναι και αυτή ένα νοητό κατασκεύασμα, που μπορεί να γίνει με διαφορετικούς τρόπους αντιληπτό. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

Η δεύτερη σημαντική διαφορά μπορεί να περιγραφεί από μια παρομοίωση: Σε αντίθεση με τα τεχνικά έργα, η κατασκευή των οποίων συνήθως ακολουθεί μια προκαθορισμένη οδό, γνωστή από την αρχή, η ανάπτυξη του λογισμικού ομοιάζει με σκόπευση κινούμενου στόχου από κινούμενο έδαφος και με όπλο που συνεχώς αλλάζει τη συμπεριφορά του. Ο στόχος είναι κινούμενος γιατί οι απαιτήσεις των χρηστών συνεχώς μεταβάλλονται, ακόμα και μέσα στη διαδικασία ανάπτυξης μιας εφαρμογής που προορίζεται να τις ικανοποιήσει. Το έδαφος είναι κινούμενο γιατί το περιβάλλον ανάπτυξης του λογισμικού είναι και το ίδιο συνεχώς εξελισσόμενο μαζί με το υλικό, αλλά και μαζί με τις επιλογές και την έκβαση των μη τεχνικών διαμαχών στο χώρο της αγοράς τεχνογνωσίας και τεχνολογίας πληροφορικής. Σαν να μην έφταναν τα παραπάνω, το όπλο με το οποίο γίνεται η «σκόπευση», δηλαδή οι μεθοδολογίες, τα εργαλεία και τα περιβάλλοντα ανάπτυξης και λειτουργίας του λογισμικού, είναι επίσης ραγδαία μεταβαλλόμενα με το χρόνο. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

Μολονότι η κατασκευή πολλών τεχνικών έργων είναι δυνατό να τυποποιηθεί σε αρκετά μεγάλο βαθμό, δεν ισχύει το ίδιο με την ανάπτυξη του λογισμικού. Η ανάπτυξη αυτή μέχρι σήμερα δεν έχει γίνει δυνατό να αυτοματοποιηθεί και το λογισμικό παραμένει ένα από τα πολυπλοκότερα και δυσκολότερα τεχνικά κατασκευάσματα του ανθρώπου, στην κατασκευή του οποίου συναντώνται επί μακρόν σημαντικά προβλήματα, τα οποία από πολλούς χαρακτηρίζονται ως «χρόνια».

3.2. Κρίση λογισμικού

Το σύνολο αυτών των χρόνιων προβλημάτων αναφέρεται ως «κρίση λογισμικού». Ενδεχομένως, η χρήση όρων όπως «κρίση» ή «χρόνια προβλήματα» να μπορεί να χαρακτηριστεί υπερβολική, αυτό όμως δεν αναιρεί ούτε τη σοβαρότητα ούτε την παρατεταμένη διάρκεια εκδήλωσης των

προβλημάτων που έχουν καταγραφεί και καθημερινά επιβεβαιώνονται στην ανάπτυξη του λογισμικού. Είναι χαρακτηριστικό ότι το λογισμικό είναι ένα από τα ελάχιστα ανθρώπινα κατασκευάσματα που πωλείται «ως έχει», χωρίς καμία απολύτως εγγύηση για τις ζημιές που μπορεί να προκαλέσει η

Πίνακας 1. Μερικά βασικά σημεία της «κρίσης λογισμικού»

Εξαιρετικά δύσκολη διαδικασία κατασκευής.	Δεν είναι πάντα σαφές ποια βήματα πρέπει να γίνουν, με ποια σειρά, με ποια ενδιάμεσα προϊόντα κτλ.
Ανεπαρκής ή και κακή ποιότητα τελικού προϊόντος	Λάθη στην κατασκευή, μη ικανοποίηση του σκοπού.
Μη τήρηση χρονοδιαγραμμάτων.	Υπερβολικές και «αδικαιολόγητες» καθυστερήσεις.
Υπερβάσεις προϋπολογισμών.	Κακές αρχικές εκτιμήσεις κόστους. Τελικά προϊόντα με πολλαπλάσιο κόστος από το αρχικά προϋπολογισθέν.
Μεγάλη δυσκολία και συνεπαγόμενο κόστος συντήρησης.	Παρενέργειες μεταβολών σε στοιχεία που πριν λειτουργούσαν, πρόχειρες λύσεις.
Δύσκολη κατανόηση εγγράφων, σχεδίων κτλ. από διαφορετικούς κατασκευαστές.	Στην πράξη, η κατανόηση ενός συστήματος λογισμικού από τρίτους, πλην των κατασκευαστών του, είναι συχνά αδύνατη ή ιδιαίτερα ασύμφορη.

χρήση του, οσοδήποτε σημαντικές και αν είναι αυτές. Ο Πίνακας 1. απεικονίζει τα σημαντικότερα από τα προβλήματα αυτά.

Προβλήματα όπως τα παραπάνω έχουν εντοπιστεί εδώ και δεκαετίες από την κοινότητα κατασκευαστών και ακαδημαϊκών ερευνητών στη

γνωστική περιοχή του λογισμικού και έχουν διατυπωθεί με πολλούς τρόπους και με πολλές ευκαιρίες. Συχνά, νέες τεχνολογίες ή προϊόντα που προτείνονται για την ανάπτυξη του λογισμικού κάνουν επίκληση των προβλημάτων αυτών, ισχυριζόμενα ότι διαθέτουν ικανοποιητικές λύσεις. Για ένα διάστημα, οι λύσεις αυτές φέρονταν ως «ασημένια σφαίρα» που θα σκότωνε το «τέρας» των προβλημάτων ανάπτυξης λογισμικού. Κάτι τέτοιο δεν έγινε και η φιλοσοφία της «ασημένιας σφαίρας» εγκαταλείφθηκε, για να πάρουν τη θέση της πιο συνετές επιστημονικές προσεγγίσεις στην ανάπτυξη του λογισμικού. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

Ως αποτέλεσμα, αναπτύχθηκε ένας ειδικός κλάδος της επιστήμης της πληροφορικής, που ονομάστηκε Τεχνολογία Λογισμικού (Software Engineering). Πρόσφατα προτάθηκε η Τεχνολογία Λογισμικού να αποτελέσει εξειδίκευση της επιστήμης του μηχανικού, οπότε μια πιο εύστοχη απόδοση στα ελληνικά είναι αυτή της Μηχανικής Λογισμικού.

3.3. Τεχνολογία Λογισμικού

Όπως ακριβώς με τον ορισμό της έννοιας «λογισμικό», προβλήματα και αρκετές διαφορετικές απόψεις υπάρχουν και στον ορισμό της Τεχνολογίας Λογισμικού.

Τεχνολογία Λογισμικού είναι:

η περιοχή εκείνη της επιστήμης της μηχανικής η οποία ασχολείται με την εύρεση και θεμελίωση μεθόδων για να περιγράφεται, να κατασκευάζεται και να συντηρείται λογισμικό.

Επιθυμητά χαρακτηριστικά του λογισμικού και της διαδικασίας κατασκευής του είναι η ποιότητα, η μεγαλύτερη δυνατή αυτοματοποίηση και παραγωγικότητα και το ελάχιστο δυνατό κόστος παραγωγής και συντήρησης. Οι έννοιες «ποιότητα», «αυτοματοποίηση», «παραγωγικότητα» και «κόστος» είναι σε πολλές περιπτώσεις αντίθετες ως αντικειμενικοί σκοποί. Είναι φυσικό να μιλάμε όχι για ταυτόχρονη μεγιστοποίηση ποιότητας και παραγωγικότητας, από τη μία και απόλυτη ελαχιστοποίηση του κόστους,

από την άλλη, αλλά για αποδεκτή — στις εκάστοτε συνθήκες— ισορροπία μεταξύ αυτών των μεγεθών. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

Εντός του πεδίου της Τεχνολογίας Λογισμικού είναι ο καθορισμός των ενεργειών και της αλληλουχίας με την οποία αυτές πρέπει να γίνονται (software process), καθώς και η περιγραφή με σαφή και κατανοητό τρόπο όλων των προϊόντων που παράγονται κατά την εκτέλεση αυτών των ενεργειών. Το τελικό παραδοτέο προϊόν κάθε ενέργειας ανάπτυξης λογισμικού είναι ο «εκτελέσιμος κώδικας», δηλαδή ένα σύνολο εντολών άμεσα εκτελέσιμων από έναν ηλεκτρονικό υπολογιστή κάτω από συγκεκριμένες (και γνωστές εκ των προτέρων) προϋποθέσεις. Το σύνολο αυτών των εντολών αποτελεί μια περιγραφή του τρόπου εκτέλεσης των εργασιών που αυτοματοποιούνται με τη χρήση μιας εφαρμογής λογισμικού.

Δεν είναι δυνατό η κατασκευή του λογισμικού να οδηγήσει κατευθείαν στον εκτελέσιμο κώδικα, όπως, άλλωστε, καμία απολύτως τεχνική κατασκευή δεν μπορεί να γίνει κατευθείαν, χωρίς να έχουν προηγηθεί μελέτες και σχέδια. Ωστόσο, ένα στοιχείο που διαφοροποιεί σημαντικά το λογισμικό από τις κλασικές τεχνικές κατασκευές είναι ότι η κατασκευή του δεν είναι μια σειριακά ακολουθούμενη διαδικασία, η οποία ολοκληρώνεται με την κατασκευή του παραδοτέου προϊόντος, αλλά το αρχικό αυτό παραδοτέο (πρώτη έκδοση εκτελέσιμου κώδικα και αντίστοιχο υλικό τεκμηρίωσης) υπόκειται συχνά πολλές τροποποιήσεις. Συνήθεις αιτίες για τροποποιήσεις στο λογισμικό είναι:

- η διόρθωση σφαλμάτων,
- η βελτιστοποίηση της απόδοσης,
- η αυτοματοποίηση της εκτέλεσης νέων εργασιών,
- η ενσωμάτωση μεταβολών που οφείλονται σε αλλαγές που συμβαίνουν στον πραγματικό κόσμο,

Η πραγματοποίηση μεταβολών . διορθώσεων στις εφαρμογές λογισμικού αναφέρεται με τον όρο συντήρηση λογισμικού (software maintenance). Όλες οι φάσεις από τις οποίες διέρχεται το λογισμικό

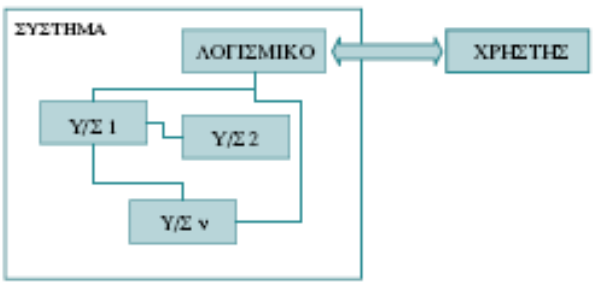
αναφέρονται ως κύκλος ζωής λογισμικού (software life cycle). Γίνεται σαφές ότι η Τεχνολογία Λογισμικού δεν ασχολείται μόνο με την κατασκευή, αλλά με ολόκληρο τον κύκλο ζωής του λογισμικού. Χρονικά, πρόκειται για το διάστημα από τη σύλληψη της ιδέας της κατασκευής μιας εφαρμογής λογισμικού μέχρι την απόσυρση αυτής από τη χρήση. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

3.4. Το Λογισμικό ως μέρος συστημάτων

Συχνά, το λογισμικό αντιμετωπίζεται λανθασμένα, όχι ως μέρος ενός ευρύτερου συστήματος με το οποίο αλληλεπιδρά με πολλούς τρόπους, αλλά ως αυθύπαρκτη οντότητα. Στην Τεχνολογία Λογισμικού συχνά γίνεται ξεχωριστά λόγος για το σύστημα και ξεχωριστά για το λογισμικό και δεν είναι λίγες οι περιπτώσεις όπου μπορεί να δημιουργηθεί σύγχυση σχετικά με τις έννοιες και την προσέγγιση πολλών οντοτήτων του πραγματικού κόσμου κατά την ανάπτυξη λογισμικού. Θα διακρίνουμε δύο περιπτώσεις:

- Το λογισμικό αποτελεί εσωτερικό συστατικό ενός τεχνητού μη υπολογιστικού συστήματος.
- Το λογισμικό λειτουργεί αυτοτελώς σε ένα υπολογιστικό σύστημα.

Ως παραδείγματα για το πρώτο μπορούμε να αναφέρουμε όλες τις περιπτώσεις όπου μια συσκευή λειτουργεί χρησιμοποιώντας λογισμικό, όπως οι μηχανές αυτόματης πώλησης, οι ψηφιακοί αυτοματισμοί και, σύντομα, αρκετές οικιακές συσκευές. Επίσης, στην πρώτη κατηγορία ανήκουν σύνθετα συστήματα όπου το λογισμικό ή η υπολογιστική μονάδα στην οποία αυτό εκτελείται λειτουργεί συνδεδεμένη με άλλες συσκευές, όπως τα συστήματα χρονομέτρησης αγώνων, τα ιατρικά μηχανήματα ανάλυσης και απεικόνισης, τα συστήματα ελέγχου εναέριας κυκλοφορίας κ.ά. (Σχήμα 1.). (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)



Σχήμα 1. Το λογισμικό μπορεί να είναι μέρος πολλών συστημάτων. Ο χρήστης, αλληλεπιδρώντας με τα συστήματα, μπορεί να χρησιμοποιεί λογισμικό χωρίς να έχει άμεση αντίληψη του γεγονότος αυτού.

3.5. Το Λογισμικό ως προϊόν

Οι περισσότερες προσπάθειες ανάπτυξης λογισμικού στοχεύουν στη δημιουργία ενός προϊόντος, δηλαδή ενός αγαθού, το οποίο προορίζεται να βρει το δρόμο του στην αγορά, είτε ως προϊόν μαζικής κατανάλωσης είτε ως κατά παραγγελία κατασκευασμένο. Η περίπτωση στην οποία μπορεί κανείς να κατασκευάσει λογισμικό για δική του προσωπική χρήση δε διαφοροποιείται σε τίποτε από τεχνική άποψη, είναι, ωστόσο, περιορισμένου ενδιαφέροντος, με το σκεπτικό ότι αποτελεί μια εξαίρεση στην οποία όποιοι κανονισμοί επιβάλλονται από επιστημονικά θεμελιωμένες προσεγγίσεις, μπορούν να τηρηθούν με μεγάλη χαλαρότητα. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

Σε αντίθεση με τις θεωρητικές επιστήμες, όπου η δυνατότητα χρήσης απεριόριστου χρόνου προκειμένου να επιτευχθεί ένας στόχος είναι μεγαλύτερη, οι νόμοι του ανταγωνισμού και η πίεση του χρόνου δεν επιτρέπουν κάτι τέτοιο με το λογισμικό. Η επιδίωξη της κατάκτησης μιας καλής θέσης στην αγορά περιορίζει τη δυνατότητα εφαρμογής εξαντλητικών διαδικασιών εξασφάλισης ποιότητας και αποτελεί μια από τις βασικές αιτίες για πολλές από τις πληγές του λογισμικού σήμερα. Η λειτουργία των νόμων

τηςαγοράς έχει και άλλο ένα ενδιαφέρον αποτέλεσμα: το καλύτερο από τεχνικής πλευράς λογισμικό δεν είναι κατ. ανάγκη και το επικρατέστερο στην αγορά. Σε αρκετές περιπτώσεις, διαμάχες για τεχνολογίες και προϊόντα λογισμικού, οι οποίες εμφανίζονται ως τεχνικές διαμάχες, μόνο τέτοιες δεν είναι. Ο ανταγωνισμός δυσχεραίνει τη συνεργασία με σκοπό την αναζήτηση του καλύτερου δυνατού αποτελέσματος. Συχνά, περισσότερες από μία πλευρές κατέχουν η καθεμία ένα μέρος της επιθυμητής λύσης σε ένα πρόβλημα, χωρίς καμία να κατέχει ολόκληρη τη λύση και χωρίς να βαδίζουν από κοινού σε δρόμο σύγκλισης. Κάποιες φορές οι διαμάχες αυτές αποβαίνουν σε όφελος του τελικού καταναλωτή, ενώ σε πολλές περιπτώσεις συμβαίνει ακριβώς το αντίθετο.

Επιχειρώντας μια γενική ταξινόμηση των προϊόντων λογισμικού, μπορούμε να διακρίνουμε δύο γενικές κατηγορίες. Το λογισμικό συστήματος και το λογισμικό εφαρμογών. Ως λογισμικό συστήματος γίνεται αντιληπτό εκείνο το λογισμικό χωρίς το οποίο δεν είναι δυνατή η λειτουργία ενός Η/Υ, δηλαδή τα λειτουργικά συστήματα (operating systems). Εντοπίζοντας υποκατηγορίες, έχουμε τα λειτουργικά συστήματα γενικής χρήσης (Unix, DOS, Windows), καθώς και ειδικές περιπτώσεις, όπως λογισμικό προγραμματισμού (όχι εφαρμογές) αυτόματων ελεγκτών στη βιομηχανία. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

Στην κατηγορία του λογισμικού εφαρμογών ανήκουν, γενικά, όλες οι υπόλοιπες περιπτώσεις. Και εδώ μπορούμε να διακρίνουμε υποκατηγορίες, όπως το λογισμικό επιχειρηματικών εφαρμογών, εφαρμογών πραγματικού χρόνου, επιστημονικών εφαρμογών, εκπαιδευτικών εφαρμογών, προσωπικής χρήσης, τεχνητής νοημοσύνης κ.ά. Μπορούν να αναζητηθούν και άλλες κατηγορίες ή υποκατηγορίες, σύμφωνα με άλλη κεντρική ιδέα ταξινόμησης, ώστε ο κατάλογος να συνεχίζεται και να εξειδικεύεται επί μακρόν. Ο ρυθμός των εξελίξεων και η χρήση
του λογισμικού σε ολοένα και περισσότερες πλευρές της καθημερινής ζωής, όχι μόνο στις επαγγελματικές, έχει καταστήσει ανεπίκαιρες όλες τις

απόπειρες ολοκληρωμένης ταξινόμησης του λογισμικού σε κατηγορίες οι οποίες έχουν γίνει στο παρελθόν.

Η υπόσταση του λογισμικού ως προϊόντος επιβάλλει για την Τεχνολογία Λογισμικού τη διατύπωση ενός συνόλου κανόνων και διαδικασιών ανάπτυξης που να ισορροπούν μεταξύ τεχνικής ορθότητας (στο μέτρο που αυτή είναι θεμελιωμένη), από τη μία, και οικονομικής εφικτότητας, από την άλλη. Επιπλέον, η ανάπτυξη λογισμικού οφείλει να γίνεται σε «λογικό» χρόνο, ώστε αυτό να εισέρχεται στην αγορά σε στιγμή που η ζήτηση είναι υψηλή και να αποφέρει κέρδη στον κατασκευαστή του. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

3.6 Συστατικά στοιχεία Λογισμικού

Λογισμικό δεν είναι μόνο ένα εκτελέσιμο πρόγραμμα. Οι χρήστες συνήθως αντιλαμβάνονται ως λογισμικό το «πρόγραμμα» μαζί με το αντίστοιχο εγχειρίδιο χρήσης. Πέραν αυτών, μέρος του λογισμικού είναι και πολλά ενδιάμεσα προϊόντα που παράγονται στις φάσεις που μεσολαβούν από τον καθορισμό των εργασιών που θα αυτοματοποιηθούν με τη βοήθεια του λογισμικού μέχρι την παραγωγή του εκτελέσιμου κώδικα. Τα προϊόντα αυτά είτε είναι ενδιάμεσα συστατικά λογισμικού που παράγονται μέχρι να φτάσουμε στον εκτελέσιμο κώδικα (πηγαίος κώδικας, κώδικας μορφής object, βιβλιοθήκες κ.ά.) είτε περιγράφουν τη δομή και τη συμπεριφορά του λογισμικού. Στη δεύτερη περίπτωση αναφέρονται με τον όρο τεκμηρίωση λογισμικού (software documentation) και βρίσκονται σε έντυπη ή σε ηλεκτρονική μορφή. Κατά παρέκκλιση τηςαυστηρής λεξικογραφικής σημασίας της λέξης, σε αυτή την τεκμηρίωση του λογισμικού δεν καταγράφεται το γιατί το λογισμικό εκτελεί κάποιες εργασίες ή γιατί τις εκτελεί με ένα συγκεκριμένο τρόπο, αλλά το ποιες εργασίες θα εκτελεί, πώς θα τις εκτελεί, ποιες δομές δεδομένων θα χρησιμοποιηθούν κ.ά. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

Συστατικά λογισμικού: είναι όλα τα προϊόντα που παράγονται κατά την ανάπτυξη του λογισμικού, τα οποία αποτελούν αναπόσπαστο μέρος αυτού.

Τα συστατικά λογισμικού μπορούν να ταξινομηθούν ως προς τη φύση τους, τον τρόπο παραγωγής τους, τη φάση του κύκλου ζωής στην οποία παράγονται, την εσωτερική τους δομή, τα πρότυπα στα οποία ενδεχομένως συμμορφώνονται κ.ά. Ως προς τη φύση διακρίνουμε αυτά που βρίσκονται σε ηλεκτρονική μορφή και αυτά που βρίσκονται σε έντυπη.

Ως προς τον τρόπο παραγωγής τους διακρίνουμε αυτά που παράγονται αυτόματα (κώδικας μορφής object, εκτελέσιμος κώδικας, περιγραφή σχημάτων βάσεων δεδομένων κ.ά.) και αυτά που παράγονται με το χέρι.

Η ταξινόμηση ως προς την εσωτερική τους δομή ποικίλλει ανάλογα με την τεχνική φύση του περιβάλλοντος ανάπτυξης και λειτουργίας. Τέλος, η συμμόρφωση με πρότυπα αναφέρεται στη δομημένη περιγραφή ορισμένων χαρακτηριστικών του λογισμικού, με το «δομημένη» να αφορά την πειθαρχία απέναντι σε όσα ορίζονται σε ένα ή περισσότερα πρότυπα (standards) που χρησιμοποιούνται για το σκοπό αυτό. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

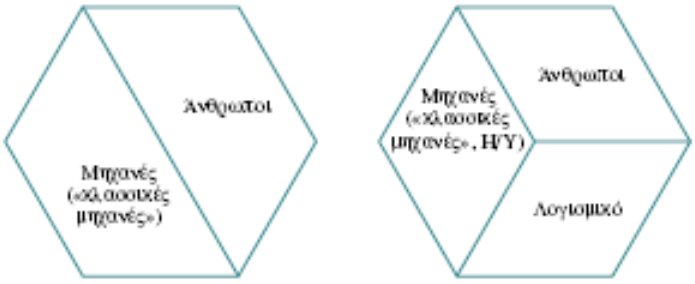
Η αναφορά στα πρότυπα μας φέρνει σε ένα σημαντικό πρόβλημα στο χώρο του λογισμικού. Σήμερα, μέσα στην κοινότητα κατασκευαστών και ερευνητών του λογισμικού υπάρχει ένας πλουραλισμός συμβόλων, τίτλων, ορισμών εννοιών, δομών κτλ. που αναφέρονται σε παρεμφερείς οντότητες, χωρίς να διευκολύνουν μια καθολική, σαφή και χωρίς διφορούμενα κατανόηση των οντοτήτων που σχετίζονται με το λογισμικό. Ο πλουραλισμός αυτός είναι έκδηλος παντού: στις μεθοδολογίες ανάπτυξης, στις γλώσσες προγραμματισμού, στα εργαλεία και αλλού και οδηγεί τους κατασκευαστές στην επιλογή δικών τους επιλύσεων σε διφορούμενα θέματα τεκμηρίωσης και στη χρήση δικών τους άτυπων συμβολισμών και δομών, γεγονός που μάλλον δυσχεραίνει το πρόβλημα και αυξάνει τη σύγχυση.

Αιτίες της σύγχυσης αυτής μπορούν να αναγνωριστούν σε αρκετά επίπεδα. Πέρα από την αρχική ανωριμότητα, η οποία χαρακτηρίζει κάθε νέο ερευνητικό πεδίο, υπάρχει ο ανταγωνισμός για την εμπορική επικράτηση σε τομείς όπως τα εργαλεία ανάπτυξης και η παροχή τεχνογνωσίας για την ανάπτυξη λογισμικού. Ακόμα και οι συμβολισμοί και η ορολογία αποτελούν πεδίο διαμάχης και σύγχυσης. Μια αξιοσημείωτη προσπάθεια για την ανάπτυξη προτύπων για την περιγραφή πολλών συστατικών λογισμικού έκανε ο οργανισμός IEEE (the Institute
of Electrical and Electronics Engineers). Τα πρότυπα αυτά προσαρμόζονται ανάλογα με τη μεθοδολογία ανάπτυξης και τον ακολουθούμενο κύκλο ζωής. Ωστόσο, έχουν ένα κάθε άλλο παρά αμελητέο κόστος συγγραφή και ιδιαίτερα διατήρησής τους σε επίκαιρη κατάσταση, με αποτέλεσμα σε αρκετές περιπτώσεις είτε να καταργούνται στην πράξη είτε να μένουν χωρίς να ενημερώνονται με τις μεταβολές που λαμβάνουν χώρα από την αρχική συγγραφή τους και μετά. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009)

ΚΕΦΑΛΑΙΟ 4^{Ο}
ΜΗΧΑΝΙΚΗ ΤΩΝ ΑΠΟΦΑΣΕΩΝ

4.1. Απαίτηση

Σε κάθε σύστημα του πραγματικού κόσμου μπορούμε να διακρίνουμε δύο συνιστώσες: τους ανθρώπους και τις μηχανές. Σύμφωνα με τη διάκριση αυτή, ο ηλεκτρονικός υπολογιστής είναι, ασφαλώς, μια μηχανή, η οποία όμως παρουσιάζει σημαντικές διαφορές από τις μηχανές που ήταν γνωστές πριν από την εμφάνισή του. Συνοπτικά, οι διαφορές αυτές εστιάζονται στο ότι ο ηλεκτρονικός υπολογιστής δεν έχει υπόσταση παρά μόνο με τη βοήθεια λογισμικού, και μάλιστα η υπόσταση αυτή είναι διαφορετική ανάλογα με το λογισμικό που χρησιμοποιείται. Όταν, λοιπόν, σε ένα σύστημα συμπεριλαμβάνεται και ο ηλεκτρονικός υπολογιστής, είναι σκόπιμο να διακρίνουμε και μια Τρίτη συνιστώσα, αυτή του λογισμικού (Σχήμα 2.).



Σχήμα 2. Μια διάκριση των συνιστωσών των συστημάτων πριν (αριστερά) και μετά (δεξιά) την εμφάνιση των Η/Υ και του λογισμικού

Ο «προσδιορισμός των απαιτήσεων από το σύστημα» είναι μια εργασία παρόμοια με τον «προσδιορισμό των απαιτήσεων από το λογισμικό», μόνο που εκτός από το λογισμικό αφορά και τις άλλες συνιστώσες ενός συστήματος, δηλαδή τους ανθρώπουςκαι τις μηχανές. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Αρκετές από τις απαιτήσεις από το σύστημα μπορεί να σχετίζονται έμμεσα ή άμεσα με απαιτήσεις από το λογισμικό. Η διάκριση μεταξύ απαιτήσεων από το σύστημα και απαιτήσεων από το λογισμικό δεν είναι πάντα εύκολη και συχνά δημιουργεί σύγχυση, είναι, ωστόσο, χρήσιμη για το σαφέστερο προσδιορισμό της υπόστασης του λογισμικού, αλλά και για την καλύτερη κατανόηση ολόκληρου του συστήματος. Ένας απλός τρόπος να διακρίνουμε αν μια απαίτηση που διατυπώνεται αφορά ολόκληρο το σύστημα ή το λογισμικό είναι να προσπαθούμε να απαντήσουμε στην ερώτηση: «Ποια από τις συνιστώσες του συστήματος πρέπει να ικανοποιήσει την απαίτηση αυτή;». Αν η απάντηση «δείχνει» το λογισμικό, τότε μιλάμε για απαίτηση από το λογισμικό. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

4.1.2. Απαίτηση από το σύστημα

Μια απαίτηση από το σύστημα είναι η περιγραφή μιας εργασίας που θα πρέπει να εκτελείται από κάποια εκ των συνιστωσών του συστήματος (άνθρωποι, μηχανές, λογισμικό) ή ενός χαρακτηριστικού το οποίο θα πρέπει να έχει ένα σύστημα.

Σύμφωνα με τον ορισμό αυτό, οι απαιτήσεις από το λογισμικό είναι στην ουσία απαιτήσεις από το σύστημα. Κατά τον προσδιορισμό των απαιτήσεων από το σύστημα η προσοχή είναι εστιασμένη στην κατασκευή ολόκληρου του συστήματος και ότι η δομή και η λεπτομέρεια με τις οποίες γίνεται η περιγραφή των απαιτήσεων που αφορούν το λογισμικό δεν είναι επαρκείς για την κατασκευή του. Οι απαιτήσεις από το σύστημα μπορούν να καταγράφονται με διάφορους τρόπους, ο πιο απλός εκ των οποίων είναι το απλό κείμενο, συνοδευόμενο ενδεχομένως από σχήματα. Περισσότερο δομημένοι τρόποι περιγραφής έχουν προταθεί με διάφορα πρότυπα, όπως αυτά του IEEE. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

4.1.3. Τι είναι απαίτηση από το λογισμικό

Όταν κατασκευάζουμε λογισμικό, το πρώτο που πρέπει να συλλάβουμε, με όσο το δυνατό μεγαλύτερη σαφήνεια, είναι οι εργασίες που αυτό θα πρέπει να κάνει, καθώς και άλλα χαρακτηριστικά που είναι επιθυμητό να έχει, όπως, για παράδειγμα, η εμφάνιση, οι επιδόσεις, ο τρόπος χρήσης, η ασφάλεια κ.ά. Τόσο οι εργασίες όσο και τα χαρακτηριστικά αυτά θα καθορίσουν σε σημαντικό βαθμό τις δραστηριότητες που θα ακολουθήσουν κατά την ανάπτυξη του λογισμικού. Είναι φανερό ότι λανθασμένη ή έστω αποκλίνουσα αντίληψη των απαιτούμενων από το λογισμικό εργασιών και χαρακτηριστικών μπορεί να οδηγήσει στην κατασκευή λογισμικού που δεν επιτελεί το σκοπό του. Ο προσδιορισμός και η σαφής περιγραφή των απαιτήσεων είναι ένας ιδιαίτερα σημαντικός κρίκος στην αλυσίδα των εργασιών που εκτελούνται κατά τον κύκλο ζωής μιας εφαρμογής λογισμικού. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Απαίτηση από το λογισμικό

Μια απαίτηση από το λογισμικό είναι μια λειτουργία που αυτό θα πρέπει να επιτελεί ή μια συνθήκη που θα πρέπει να ικανοποιεί, όταν θα έχει ολοκληρωθεί η κατασκευή του.

Ακόμη και από τη θέση του χρήστη λογισμικού, δεν είναι δύσκολο να αντιληφθεί κανείς ότι οι απαιτήσεις έστω και από μια μικρή εφαρμογή λογισμικού είναι πολλές και διαφορετικού χαρακτήρα, ενώ μπορούν να περιγραφούν με μικρότερη ή μεγαλύτερη λεπτομέρεια. Συνήθως ο πελάτης εκφράζει τις απαιτήσεις του με μεγάλο βαθμό γενικότητας, ενώ ο κατασκευαστής τις διατυπώνει με μεγαλύτερη λεπτομέρεια και σαφήνεια, ώστε να μπορεί να κάνει τη δουλειά του.
(Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

4.1.3 Ταξινόμηση απαιτήσεων από το λογισμικό

Οι απαιτήσεις από το λογισμικό διακρίνονται σε δύο μεγάλες κατηγορίες. Στις «λειτουργικές» και στις «μη λειτουργικές».

Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις περιγράφουν τις εργασίες (λειτουργίες) που θα πρέπει να εκτελεί το λογισμικό.

Οι λειτουργικές απαιτήσεις καθορίζουν πλήρως τη συμπεριφορά του συστήματος, δηλαδή τα επιθυμητά αποτελέσματα που αυτό πρέπει να παράγει ή γενικά την απόκριση που πρέπει να εμφανίζει στο περιβάλλον του όταν ισχύουν συγκεκριμένες συνθήκες. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Μη λειτουργικές απαιτήσεις

Οι μη λειτουργικές απαιτήσεις περιγράφουν χαρακτηριστικά που πρέπει να έχει το λογισμικό τα οποία δεν αφορούν την εκτέλεση κάποιας λειτουργίας από αυτό.

Οι μη λειτουργικές απαιτήσεις καθορίζουν ιδιώματα εμφάνισης, περιβάλλοντος λειτουργίας, επιδόσεων κ.ά., τα οποία γενικά χαρακτηρίζουν το λογισμικό, χωρίς όμως να μπορούν να ιδωθούν ως λειτουργίες που αυτό επιτελεί. Ως μη λειτουργικές χαρακτηρίζονται και οι απαιτήσεις που αφορούν κάποια από τις επόμενες φάσεις του κύκλου ζωής του λογισμικού. Οι μη λειτουργικές απαιτήσεις μπορούν να ταξινομηθούν σε επιμέρους κατηγορίες, οι οποίες αναφέρονται ακολούθως. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Απαιτήσεις χρήσης: Καθορίζουν τα χαρακτηριστικά της χρήσης του συστήματος, την αισθητική της επικοινωνίας με το χρήστη (user interface), καθώς και το υλικό τεκμηρίωσης και εκπαίδευσης που θα έχει στη διάθεσή του ο τελικός χρήστης.

Απαιτήσεις αξιοπιστίας: Καθορίζουν τη συμπεριφορά του λογισμικού σε καταστάσεις ενδογενών ή εξωγενών σφαλμάτων, τη διαδικασία αποκατάστασης, την πρόβλεψη τέτοιων καταστάσεων, καθώς και την επιθυμητή διαθεσιμότητα του λογισμικού.

Απαιτήσεις επιδόσεων: Εισάγουν περιορισμούς σε λειτουργικές απαιτήσεις σχετικά με το χρόνο εκτέλεσής τους και με τη χρήση πόρων, όπως η μνήμη και οι μονάδες επεξεργασίας. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Απαιτήσεις υποστήριξης: Καθορίζουν τα επιθυμητά χαρακτηριστι
κά για τον έλεγχο και τη συντήρηση του λογισμικού.

Απαιτήσεις σχεδίασης: Καθορίζουν τον τρόπο με τον οποίο θα πρέπει να γίνει η σχεδίαση του λογισμικού.

Απαιτήσεις υλοποίησης: Καθορίζουν τον τρόπο με τον οποίο θα πρέπει να γίνει η συγγραφή του *πηγαίου κώδικα* (source code) του λογισμικού.

Απαιτήσεις επικοινωνίας με άλλα συστήματα: Καθορίζουν τα εξωτερικά συστήματα, λογισμικού ή άλλα, με τα οποία το λογισμικό θα επικοινωνεί, καθώς και τον τρόπο (λ.χ. πρότυπα, φυσική σύνδεση) πραγματοποίησης της επικοινωνίας αυτής.

Απαιτήσεις Βάσεων Δεδομένων: Καθορίζουν τις οντότητες για τη διαχείριση των οποίων είναι υπεύθυνο το σύστημα λογισμικού, καθώς και τα ιδιώματα καθεμίας από αυτές, όπως αυτά είναι αναγνωρίσιμα στην παρούσα φάση της ανάπτυξης. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Φυσικές απαιτήσεις: Καθορίζουν τα επιθυμητά φυσικά χαρακτηριστικά του λογισμικού και του συστήματος.

4.2 Μηχανική των Απαιτήσεων

Το πλήθος και η πολυπλοκότητα που χαρακτηρίζει πολλές από τις απαιτήσεις από το λογισμικό, ο σαφής προσδιορισμός και η παρακολούθηση των συσχετίσεων μεταξύ αυτών, καθώς και με απαιτήσεις από το σύστημα, τα διαφορετικά επίπεδα λεπτομέρειας στην περιγραφή τους είναι μερικά από τα σημαντικότερα προβλήματα που συναντά κανείς όταν καλείται να αντιμετωπίσει ένα πρόβλημα προσδιορισμού απαιτήσεων από το λογισμικό. Είναι ευνόητο ότι η επιτυχής αντιμετώπιση ενός τέτοιου προβλήματος δεν μπορεί να γίνει παρά μόνο με πειθαρχία, ακολουθώντας συγκεκριμένα βήματα και καταγράφοντας τα αποτελέσματα που παράγονται σε κάθε βήμα. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Υπάρχουν διάφορες προσεγγίσεις στο πρόβλημα αυτό, καθεμιά εκ των οποίων προτείνει τα δικά της βήματα ή τις δικές της λεπτομέρειες

εκτέλεσης κάθε βήματος. Στην πράξη δεν υπάρχει μια «καλύτερη» από άλλες λύση και κάθε κατασκευαστής λογισμικού ακολουθεί τελικά μια δική του εκδοχή, που περιέχει στοιχεία μίας ή και περισσότερων προσεγγίσεων. Ο τρόπος επίλυσης του προβλήματος προσδιορισμού των απαιτήσεων που προτείνει καθεμία από αυτές αναφέρεται ως *μηχανική απαιτήσεων* (requirements engineering). Θα προσεγγίσουμε το θέμα της μηχανικής απαιτήσεων ως μια γενική αλληλουχία ενεργειών που πρέπει να γίνονται κατά τον προσδιορισμό των απαιτήσεων από το λογισμικό. Κατά τις ενέργειες αυτές παράγονται ορισμένα προϊόντα με τη μορφή εγγράφων και διαγραμμάτων.

4.2.1. Προσδιορισμός απαιτήσεων

Η πρώτη πηγή για τον καθορισμό των απαιτήσεων από μια εφαρμογή λογισμικού είναι, ασφαλώς, ο πελάτης, ο οποίος περιγράφει στον κατασκευαστή τις εργασίες που θεωρεί απαραίτητο να εκτελούνται από το λογισμικό. Η περιγραφή αυτή συνήθως γίνεται με τη μορφή μιας έκθεσης, η οποία ενίοτε δεν είναι πλήρης και περιέχει ασάφειες και διφορούμενα. Η έκθεση αυτή αποτελεί το πρώτο υλικό που έχει στη διάθεσή του ο κατασκευαστής προκειμένου να καθορίσει όλα τα στοιχεία που θα του επιτρέψουν να κατασκευάσει λογισμικό που ικανοποιεί τον πελάτη, καθώς και να καθορίσει το κόστος και να εκτιμήσει το χρόνο που θα απαιτηθεί. Εκτός από τις εργασίες που θα πρέπει να εκτελεί το λογισμικό, πρέπει να προσδιοριστούν και άλλα χαρακτηριστικά του, όπως, για παράδειγμα, το περιβάλλον λειτουργίας, ο τρόπος χρήσης και οι επιδόσεις. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Από την άλλη πλευρά, ο τρόπος με τον οποίο αντιλαμβάνεται ο πελάτης τις εργασίες που εκτελεί το λογισμικό δε βρίσκεται πάντα σε αντιστοιχία με τον τρόπο με τον οποίο αυτές μπορούν να ενσωματωθούν σε μια εφαρμογή λογισμικού. Σε πολλές περιπτώσεις αυτό που ο πελάτης

αντιλαμβάνεται ως μια και μοναδική λειτουργία απαιτείται να αναλυθεί σε περισσότερες προκειμένου να υλοποιηθεί στο λογισμικό. Από τα παραπάνω συνάγεται ότι μια πρόσφορη για τον καθορισμό των απαιτήσεων από το λογισμικό διαδικασία περιγράφεται ως μια ακολουθία βημάτων, σε καθένα από τα οποία παράγεται μια και ολοένα και λεπτομερέστερη εκδοχή των απαιτήσεων από το λογισμικό. Τελικό προϊόν της διαδικασίας αυτής είναι το έγγραφο «Προδιαγραφές των απαιτήσεων από το λογισμικό», καθώς και ένα σύνολο από διαγράμματα τα οποία το συνοδεύουν. Η γενική μορφή της διαδικασίας φαίνεται στο Σχήμα 3. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

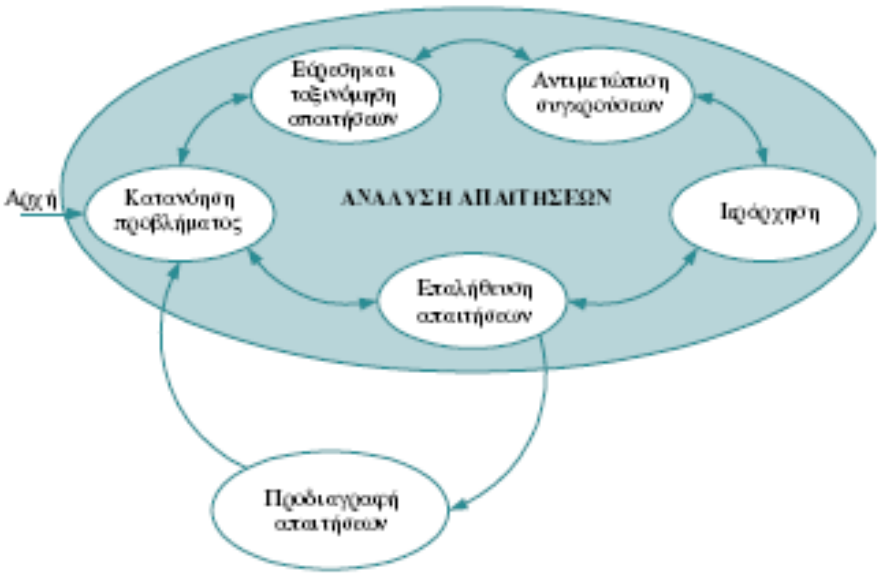
Σχήμα 3. Μηχανική απαιτήσεων:
Η γενική μορφή της διαδικασίας προσδιορισμού των απαιτήσεων από το λογισμικό

Η διαδικασία τροφοδοτείται με το έγγραφο των απαιτήσεων από το σύστημα ή, αν αυτό δεν είναι διαθέσιμο, με μια έκθεση αναγκών του πελάτη. Το πρώτο βήμα είναι η **μελέτη του εγγράφου απαιτήσεων από το σύστημα ή/και των αναγκών του πελάτη**, η οποία στοχεύει στην αρχική κατανόησητου πεδίου του προβλήματος για την επίλυση του οποίου καλείται να χρησιμοποιηθεί το λογισμικό που κατασκευάζεται. Η μελέτη αυτή

συνήθως πραγματοποιείται από διοικητική και οργανωτική σκοπιά, προκειμένου να εκτιμηθεί η βιωσιμότητα, τα ρίσκα, ο προϋπολογισμός, το χρονοδιάγραμμα και άλλες διαχειριστικές παράμετροι της ανάπτυξης λογισμικού. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

4.3 Ανάλυση απαιτήσεων

Κατά την *ανάλυση των απαιτήσεων* εντοπίζονται για πρώτη φορά οι απαιτήσεις από το λογισμικό και ακολουθούν έναν κύκλο ταξινόμησης, ιεράρχησης και επαλήθευσης, όπως φαίνεται στο Σχήμα 4. Αποτέλεσμα των εργασιών που εκτελούνται στη φάση αυτή είναι ένα σύνολο απαιτήσεων από το λογισμικό οι οποίες περιγράφονται με μορφή διαγραμμάτων. Η περιγραφή αυτή αποτελεί την είσοδο στο επόμενο βήμα, αυτό της διάκρισης και προδιαγραφής των απαιτήσεων από το λογισμικό.



Σχήμα 4. Επιμέρους βήματα που εκτελούνται κατά τη φάση της ανάλυσης απαιτήσεων από το λογισμικό.

Η έκθεση των αναγκών του πελάτη δεν είναι συνήθως επαρκής για την αντίληψη της ουσίας του αντικειμένου σχετικά με το οποίο αναπτύσσεται λογισμικό. Χωρίς την επίτευξη από πλευράς του κατασκευαστή κάποιας εξοικείωσης με την ουσία του προβλήματος στην επίλυση του οποίου συμβάλλει το λογισμικό που κατασκευάζει, δεν είναι δυνατή η επιτυχής ανάπτυξη λογισμικού που αφορά την επίλυση αυτού. Η εξοικείωση αυτή θα πρέπει να γίνει στο αρχικό στάδιο ανάπτυξης και θα σημαδέψει ακολούθως τη διαδικασία ανάπτυξης του λογισμικού. Ακολούθως **συλλέγονται οι απαιτήσεις** των εμπλεκομένων με το λογισμικό και γίνεται μια αρχική καταγραφή τους σε λίστα. Η συλλογή αυτή γίνεται με τη βοήθεια συνεντεύξεων, ερωτηματολογίων, συζητήσεων με ειδικούς, ή με άλλους, κατά περίπτωση, πρόσφορους τρόπους. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Η λίστα που δημιουργείται αρχικά περιέχει τις απαιτήσεις από το λογισμικό «ατάκτως ερριμμένες». Στη συνέχεια γίνεται μια πρώτη ταξινόμηση των απαιτήσεων σε ομάδες, ανάλογα με το υποσύνολο του προβλήματος που αφορούν, ή με άλλο, κατά περίπτωση, πρόσφορο τρόπο.

Στο σημείο αυτό ενδεχομένως να εντοπίζονται ασυνέπειες, δηλαδή δύο ή περισσότερες απαιτήσεις η ικανοποίηση των οποίων δεν μπορεί να γίνει ταυτόχρονα, οπότε είναι αναγκαία η **επίλυση συγκρούσεων**, η οποία μπορεί να επαναφέρει στο προσκήνιο τις επαφές με τον πελάτη, ή άλλη σχετική διαδικασία. Όταν έχει ολοκληρωθεί η επίλυση συγκρούσεων, οι απαιτήσεις τοποθετούνται σε μια **σειρά προτεραιότητας** ως προς τη σειρά ικανοποίησής τους. Η σειρά αυτή θα καθορίσει όχι μόνο τη χρονική αλληλουχία με την οποία ενσωματώνονται στο λογισμικό λειτουργίες που ικανοποιούν τις απαιτήσεις, αλλά και το ποιες από αυτές δε θα ικανοποιηθούν καθόλου, αν κάτι τέτοιο επιβληθεί από εξωτερικούς παράγοντες (λ.χ. κόστος). (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Η διαδικασία ολοκληρώνεται με την **επαλήθευση των απαιτήσεων**, όπως έχουν διαμορφωθεί και ιεραρχηθεί. Για να γίνει η επαλήθευση,

συνήθως απαιτείται νέα επαφή με τον πελάτη με τη μορφή συσκέψεων ή ανταλλαγής εγγράφων. Στην καλύτερη περίπτωση, οι απαιτήσεις ικανοποιούν τον πελάτη και μπορεί να ξεκινήσει η κατασκευή των μοντέλων περιγραφής λογισμικού που φαίνονται στο Σχήμα 3. Σε περίπτωσηπου οι απαιτήσεις δεν ικανοποιούντον πελάτη, τότε λαμβάνουν χώρα τόσα πισωγυρίσματα όσα είναι αναγκαία προκειμένου αυτό να γίνει.

Δεν αποτελούν εξαίρεση οι περιπτώσεις που πρέπει κανείς να επανέλθει στη διαδικασία κατανόησης του προβλήματος και να ανακαλύψει νέες πλευρές ή/και νέες ερμηνείες.

4.3.1 Προδιαγραφή απαιτήσεων

Όταν η παραπάνω διαδικασία έχει ολοκληρωθεί, έχουν κατασκευαστεί οι πρώτες εκδοχές των διαγραμμάτων ροής δεδομένων, οντοτήτων . συσχετίσεων και μετάβασης καταστάσεων. Είναι η στιγμή που μπορεί να πραγματοποιηθεί η αναλυτική προδιαγραφή των απαιτήσεων με τη σύνταξη του εγγράφου «Προδιαγραφές των απαιτήσεων από το λογισμικό». (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Προδιαγραφή

Με την έννοια «προδιαγραφή» αναφερόμαστε στη δομημένη και λεπτομερή περιγραφή των απαιτήσεων από το λογισμικό, η οποία γίνεται με τη μορφή γραπτού λόγου και, όπου απαιτείται, διαγραμμάτων.

Το έγγραφο προδιαγραφών των απαιτήσεων από το λογισμικό είναι αναμφίβολα το σημαντικότερο από τα έγγραφα τεκμηρίωσης του λογισμικού. Οι ελλείψεις και οι αστοχίες όσων αναφέρονται σε αυτό θα μεταφερθούν σε όλη την υπόλοιπη διαδικασία κατασκευής του λογισμικού και ασφαλώς στο τελικό προϊόν, γεγονός που μπορεί να έχει ως αποτέλεσμα αυτό να είναι

άχρηστο. Έχουν προταθεί αρκετοί εναλλακτικοί τρόποι δόμησης του εγγράφου προδιαγραφών των απαιτήσεων από το λογισμικό. Είναι γενικά αποδεκτό ότι μερικά επιθυμητά χαρακτηριστικά του εγγράφου αυτού είναι τα ακόλουθα:

- Θα πρέπει να περιγράφει τη συμπεριφορά του λογισμικού προς το εξωτερικό του περιβάλλον (χρήστης, άλλες εφαρμογές λογισμικού) και όχι εσωτερικά του στοιχεία.
- Θα πρέπει να περιγράφειόλους τους περιορισμούς που αφορούν την ανάπτυξη του λογισμικού.
- Θα πρέπει να είναι εύκολο να αλλαχτεί.
- Θα πρέπει να είναι χρήσιμο στη συντήρηση του λογισμικού.
- Θα πρέπει να περιγράφει τη συμπεριφορά του λογισμικού σε ανεπιθύμητες καταστάσεις.

Στο Σχήμα 5 παρατίθεται μια δομή του εγγράφου προδιαγραφών των απαιτήσεων από το λογισμικό βασισμένη σε διεθνές πρότυπο του IEEE (830.1993), το οποίο αναφέρεται στη βιβλιογραφία. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

1. Εισαγωγή
- 1.1 Ταυτότητα του εγγράφου
- 1.2 Σκοπός
- 1.3 Εμβέλεια
- 1.4 Ορισμοί, ακρωνύμια, συντομογραφίες
- 1.5 Πηγές αναφορών
- 1.6 Περίληψη

2. Γενική περιγραφή του λογισμικού
- 2.1 Στίγμα
- 2.2 Προοπτική
- 2.3 Γενικές λειτουργίες του λογισμικού
- 2.4 Χαρακτηριστικά χρηστών
- 2.5 Περιορισμοί
- 2.6 Παραδοχές και εξαρτήσεις

3. Ειδικές απαιτήσεις
- 3.1 Απαιτήσεις εξωτερικών διαπροσωπειών
  - 3.1.1 Διαπροσωπείες χρήστη
  - 3.1.2 Διαπροσωπείες υλικού
  - 3.1.3 Διαπροσωπείες λογισμικού
  - 3.1.4 Διαπροσωπείες επικοινωνιών
- 3.2 Λειτουργικές απαιτήσεις
  - 3.2.1 Τρόπος λειτουργίας 1
    - 3.2.1.1 Λειτουργική απαίτηση 1.1
      Περιγραφή, είσοδοι, επεξεργασία, έξοδοι
    - 3.2.1.2 Λειτουργική απαίτηση 1.2
      Περιγραφή, είσοδοι, επεξεργασία, έξοδοι
    - ...
  - 3.2.2 Τρόπος λειτουργίας 2
    - 3.2.2.1 Λειτουργική απαίτηση 2.1
      Περιγραφή, είσοδοι, επεξεργασία, έξοδοι
    - 3.2.2.2 Λειτουργική απαίτηση 2.2
      Περιγραφή, είσοδοι, επεξεργασία, έξοδοι
    - ...
  - ...
  - 3.2.N Τρόπος λειτουργίας N
    - 3.2.N.1 Λειτουργική απαίτηση N.1
      Περιγραφή, είσοδοι, επεξεργασία, έξοδοι
    - 3.2.N.2 Λειτουργική απαίτηση N.2
      Περιγραφή, είσοδοι, επεξεργασία, έξοδοι
    - ...
- 3.3 Απαιτήσεις επιδόσεων
- 3.4 Περιορισμοί σχεδίασης
  - 3.4.1 Περιορισμοί από το υλικό
  - 3.4.2 Συμμόρφωση με πρότυπα
- 3.5 Χαρακτηριστικά του λογισμικού
  - 3.5.1 Αξιοπιστία
  - 3.5.2 Διαθεσιμότητα
  - 3.5.3 Ασφάλεια
  - 3.5.4 Χαρακτηριστικά συντήρησης
  - 3.5.5 Μεταφερσιμότητα
- 3.6 Άλλες απαιτήσεις

Σχήμα 5. Μια προτεινόμενη από το IEEE δομή εγγράφου προδιαγραφών των απαιτήσεων
από το λογισμικό

Η παραπάνω περιγραφή αφορά τη δομή, δηλαδή τα κεφάλαια και τα περιεχόμενα καθενός εξ αυτών, τα οποία θα πρέπει να έχει το έγγραφο προδιαγραφών των απαιτήσεων από το λογισμικό. Το μόνο σχετικά νέο στοιχείο του εγγράφου είναι ο χωρισμός του κεφαλαίου 3 ανάλογα με τον «τρόπο λειτουργίας». Σε μια απλή εφαρμογή λογισμικού υπάρχει μόνο ένας τρόπος λειτουργίας (mode). Δεν ισχύει το ίδιο για μεγάλες εφαρμογές με ιδιαίτερες απαιτήσεις, οι οποίες μπορεί να έχουν περισσότερους του ενός τρόπους λειτουργίας. Η γενική δομή του εγγράφου, η οποία οφείλει να καλύπτει τη γενική περίπτωση, πρέπει να το προβλέψει αυτό. Παράδειγμα αποτελεί μια εφαρμογή συλλογής και επεξεργασίας δεδομένων σε πραγματικό χρόνο. Η συμπεριφορά της εφαρμογής τη στιγμή της συλλογής των δεδομένων, όπου οι απαιτήσεις σε επιδόσεις μπορεί να είναι ιδιαίτερα υψηλές, είναι εντελώς διαφορετική από τη συμπεριφορά της τη στιγμή της στατιστικής επεξεργασίας ή της συντήρησης των δεδομένων αυτών, οπότε διακρίνουμε δύο τρόπους λειτουργίας. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Επανερχόμενοι, μελετώντας τη δομή του εγγράφου, διαπιστώνουμε ότι όλες οι απαιτήσεις από το λογισμικό μπορούν να ταξινομηθούν ώστε να περιγραφούν σε αυτό. Η ταξινόμηση αυτή δεν είναι πάντα εύκολη υπόθεση και συχνά δημιουργούνται συγχύσεις και ερωτήματα του τύπου «Τι πρέπει να γραφτεί σε αυτή την παράγραφο;». Η απάντηση δεν είναι εύκολη και απαιτείται αρκετή τριβή με το αντικείμενο μέχρις ότου να μπορεί κανείς να νιώθει εμπιστοσύνη στον τρόπο με τον οποίο αντιμετωπίζει το θέμα. Η εμπειρία δείχνει ότι σε επόμενο στάδιο της ανάπτυξης θα φανεί το πόσο

επαρκές είναι το έγγραφο προδιαγραφών των απαιτήσεων από το λογισμικό και θα δοθεί η ευκαιρία, ενδεχομένως με κάποια αναθεώρηση, αυτό να βελτιωθεί. Η ανάγκη για τέτοιες αναθεωρήσεις συνεχώς θα μειώνεται όσο αποκτάται εμπειρία.

4.4. Απαιτήσεις από το λογισμικό

Μια εφαρμογή λογισμικού μπορεί να ιδωθεί από πολλές διαφορετικές πλευρές, ανάλογα με το σημείο που εστιάζεται το ενδιαφέρον του παρατηρητή. Για παράδειγμα, μπορεί κανείς να έχει άποψη για μια εφαρμογή είτε παρατηρώντας τον τρόπο με τον οποίο γίνεται η διαχείριση των δεδομένων σε αυτή είτε παρατηρώντας την εσωτερική δομή των δεδομένων είτε τη συμπεριφορά της εφαρμογής προς το χρήστη είτε και άλλα χαρακτηριστικά. Η παρατήρηση από όλες τις δυνατές οπτικές γωνίες προσεγγίζει την πλήρη περιγραφή της εφαρμογής, χωρίς καμία επιμέρους παρατήρηση να είναι επαρκής από μόνητης για το σκοπό αυτό. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Μοντέλο παράστασης λογισμικού

Ένα μοντέλο παράστασης λογισμικού είναι ένα διάγραμμα ή ένα σύνολο από ομοειδή διαγράμματα, το οποίο περιγράφει το λογισμικό από μια συγκεκριμένη οπτική γωνία.

Η διατύπωση «συγκεκριμένη οπτική γωνία» στον προηγούμενο ορισμό δηλώνει ότι *κανένα μοντέλο παράστασης λογισμικού δεν είναι πλήρες*, δηλαδή δεν περιέχει *όλες* τις δυνατές πληροφορίες για το λογισμικό. Είναι μια *αφαιρετική* (abstract) περιγραφή κάποιων επιλεγμένων χαρακτηριστικών του και μόνο. Ο χαρακτηρισμός «αφαιρετική» έχει εδώ το νόημα ότι αφαιρούνται τα στοιχεία που δεν ενδιαφέρουν και περιγράφονται μόνο εκείνα στα οποία εστιάζεται η προσοχή.

Υπάρχουν πολλά μοντέλα παράστασης λογισμικού, τα οποία προτείνονται από διαφορετικές μεθοδολογίες ανάπτυξης ή και από διαφορετικές εκδοχές της ίδιας μεθοδολογίας. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

Διαγράμματα που περιγράφουν συμπληρωματικά μια εφαρμογή λογισμικού, φαίνεται στο Σχήμα 6.

Σχήμα 6. Συμπληρωματικότητα των μοντέλων παράστασης λογισμικού

Η συμπληρωματικότητα έχει το νόημα ότι κανένα μοντέλο δεν περιγράφει πλήρως την εφαρμογή από μόνο του, ενώ όλα μαζί το κάνουν.

Εκτός από συμπληρωματικά, τα μοντέλα που φαίνονται στο Σχήμα 6 πρέπει να είναι και *συνεπή* μεταξύ τους. Η *συνέπεια* έχει εδώ την έννοια ότι οι οντότητες που αναφέρονται σε κάθε τέτοιο μοντέλο δεν μπορεί να είναι εντελώς ξένες με αυτές που αναφέρονται στα υπόλοιπα. Η εξασφάλιση της απαίτησης αυτής επιτυγχάνεται με τη βοήθεια του *λεξικού δεδομένων*, το οποίο περιέχει αναφορές σε όλες τις οντότητες που περιλαμβάνονται στα μοντέλα παράστασης λογισμικού. (Sommerville, 2009, Γιακουμάκης Μαν., Διαμαντίδης Νικ. 2009, IEEE1993, IEEE1987)

ΚΕΦΑΛΑΙΟ 5^{Ο}
ΜΗΧΑΝΙΚΗ ΤΩΝ ΑΠΑΙΤΗΣΕΩΝ ΣΕ ΜΙΚΡΕΣ ΚΑΙ ΜΕΣΑΙΕΣ ΕΠΙΧΕΙΡΗΣΕΙΣ: ΠΡΑΚΤΙΚΗ ΠΡΟΗΓΜΕΝΗΣ ΤΕΧΝΟΛΟΓΙΑΣ, ΠΡΟΒΛΗΜΑΤΑ, ΛΥΣΕΙΣ ΚΑΙ ΜΕΤΑΦΟΡΑ ΤΕΧΝΟΛΟΓΙΑΣ

Γενικά – Ελάχιστα είναι γνωστά για τις πρακτικές της μηχανικής απαιτήσεων σε μικρές και μεσαίες επιχειρήσεις (SMEs). Αυτή η εργασία παρουσιάζει περιληπτικά τα αποτελέσματα ενός εργαστηρίου, το οποίο ασχολήθηκε με την μηχανική απαιτήσεων και οργανώθηκε από επαγγελματίες 10 SMEs. Η τρέχουσα πρακτική προηγμένης τεχνολογίας, όπως αναφέρθηκε από τους επαγγελματίες, διαφέρει σημαντικά, όπως και τα προβλήματα τους, τα οποία οφείλονται σε συναφείς λόγους (εσωτερική υλοποίηση σε αντίθεση με υλοποίηση μετά από συμβόλαιο, τύπος προϊόντος κτλ). Παρουσιάστηκε στους συμμετέχοντες μία ομάδα τρεχόντων προτύπων της μηχανικής απαιτήσεων, τεχνικών, μεθόδων και εργαλείων. Οι σημαντικές έννοιες

υπογραμμίστηκαν από μικρές μελέτες περιπτώσεων και πειράματα, τα οποία θεωρήθηκαν ως ένα καλό μέσο για την μεταφορά τεχνολογίας (technology transfer). Περιγράφεται ο σχεδιασμός και τα αποτελέσματα, καθώς και η βαθμολογίες των επαγγελματιών στις τεχνικές και μεθόδους.

5.1. Εισαγωγή

Οι μελέτες που ασχολούνται με τις πρακτικές της μηχανικής απαιτήσεων στην βιομηχανία είναι λίγες. Η πιο συχνά αναφερόμενη μελέτη πραγματοποιήθηκε από τον Boehm στις αρχές της δεκαετίας του 1980 (‘κανόνας 1-10-100’) (Barry W. Boehm, 1981). Πρόσφατα ορισμένες μελέτες πεδίου παρείχαν λεπτομερείς πληροφορίες, όπως η μελέτη των Lubars et al. (Mitch Lubars, Colin Potts and Charles Richter, 1993) σχετικά με την μηχανική περιπτώσεων και των Gerhart et al. (Susan Gerhart, Dan Craigen and Ted Ralston, 1994) σχετικά με τις συγκεκριμένες μεθόδους. Παρόλα αυτά, ελάχιστα είναι γνωστά σχετικά με τις πρακτικές της μηχανικής απαιτήσεων σε μικρές και μεσαίες επιχειρήσεις.

Οι Μικρές και Μεσαίες Επιχειρήσεις (SMEs) θεωρούνται συχνά ως ο κινητήριος άξονας της βιομηχανικής ανάπτυξης. Είναι πολύ δυναμικές, καινοτόμες και αποδοτικές, ενώ από την πλευρά του Συμβούλου Μηχανικής Λογισμικού είναι ενδιαφέρουσες. Παρόλα αυτά, απαιτούν πελάτες με δυνατότητα μεγάλης ανάπτυξης. Για παράδειγμα, ορισμένες SMEs αναπτύσσουν μεγάλα συστήματα λογισμικού με σημαντική πολυπλοκότητα (π.χ. ένα γνωστό σύστημα έχει 2.000.000 γραμμές κωδικού). Σε σύγκριση με τις μεγάλες επιχειρήσεις, παρόλα αυτά, οι SMEs έχουν συγκεκριμένα προβλήματα:

- Το επίπεδο ωριμότητας της Μηχανικής Λογισμικού είναι πολύ χαμηλό.
- Οι εργαζόμενοι και η διεύθυνση καταβάλλονται από την καθημερινή εργασία και δεν αφήνουν χώρο για τα στρατηγικά ζητήματα, όπως η βελτίωση της ποιότητας και των διαδικασιών.

- Υπάρχει μεγάλη ζήτηση για μεταφορά τεχνογνωσίας όσο αφορά στα βασικά ζητήματα του τρόπου επίτευξης.
- Οι ιδιοκτήτες και η διεύθυνση συχνά δεν μπορούν να συνεργαστούν με εξωτερικούς συμβούλους.

Παρόλα αυτά, ο αυξημένος αριθμός εργαλείων για την μηχανική των απαιτήσεων, καθώς και της μοντελοποίησης στην αγορά μπορεί να θεωρηθεί ως υπόδειξη ότι οι SMEs αρχίζουν να κινούνται από την ad hoc ανάπτυξη λογισμικού στις πιο συστηματικές πρακτικές (αυτό το συμπέρασμα βασίζεται στην υπόθεση ότι οι μεγάλες εταιρίες έχουν ήδη μία σταθερή διαδικασία μηχανικής απαιτήσεων).

Η Πρωτοβουλία Τεχνολογίας Λογισμικού (STIe.V.) είναι μία ένωση περισσότερων από τριάντα εταιριών, συμβούλων και δημόσιων οργανισμών, η οποία προσφέρει συμβουλευτική και μία ποικιλία σεμιναρίων, προγραμμάτων εκπαίδευσης και εργαστηρίων κυρίως στις SMEs όσο αφορά στο λογισμικό και τα βιομηχανικά συστήματα. Αυτή η μελέτη βασίστηκε σε ένα εργαστήριο STI σχετικά με την μηχανική των απαιτήσεων. Η γενική εμπειρία με SMEs σε αυτό το πλαίσιο είναι ότι:

- Οι SMEs δεν συμμετέχουν σε μεγάλα συνεργατικά έργα.
- Οι SMEs απαιτούν την μεταφορά ώριμων αποτελεσμάτων σε μικρές χρονικές περιόδους.
- Οι SMEs είναι ευαίσθητες στο θέμα της τιμής.

Το εργαστήριο STI για την μηχανική των απαιτήσεων αποτελούνταν από πέντε συναντήσεις, με την κάθε συνάντηση να διαρκεί τέσσερις ώρες. Η κάθε συνάντηση αποτελούνταν από μία αναφορά εμπειρίας από έναν συμμετέχοντα, ένα μάθημα για ένα θέμα μηχανικής απαιτήσεων, μία μικρή μελέτη περίπτωσης ή πείραμα και συζητήσεις. Προσπαθήσαμε να κάνουμε τα μαθήματα συμμετοχικά.

Σε αυτή την εργασία παρουσιάζουμε αρχικά την πρακτική τελευταίας τεχνολογίας σε ένα τυχαίο δείγμα SMEs, οι οποίες συμμετείχαν στο εργαστήριο μας. Παρουσιάζουμε μία επισκόπηση των θεμάτων που παρουσιάστηκαν στο εργαστήριο και την λογική της επιλογής τους. Στην

συνέχεια παρουσιάζονται οι μελέτες περιπτώσεων και τα πειράματα. Επισκοπούμε αυτές τις μελέτες και τις πληροφορίες που δόθηκαν στους επαγγελματίες. Η εργασία τελειώνει με μία περίληψη και τα συμπεράσματα.

5.2. Πρακτική τελευταίας τεχνολογίας και προβλήματα

Το κεφάλαιο αυτό περιγράφει την πρακτική τελευταίας τεχνολογίας που παρατηρήσαμε σε ένα τυχαίο δείγμα SMEs του εργαστηρίου μας. Οι πληροφορίες που παρουσιάζονται στην συνέχεια βασίζονται σε παρουσιάσεις, οι οποίες δόθηκαν από αντιπροσώπους των εταιριών, και σε συζητήσεις που εξελίχθηκαν στο εργαστήριο.

Οι SMEs που συμμετείχαν στο εργαστήριο λειτουργούν σε μία μεγάλη κλίμακα τομέων εφαρμογών: συστήματα επιχειρηματικών πληροφοριών, συστήματα γεωγραφικών πληροφοριών, συστήματα multi-media, συστήματα εφαρμογών στην αεροπορία, λογισμικό για μηχανές κατασκευής ρουχισμού και μαθηματικό λογισμικό για πολιτικούς μηχανικούς. Οι συμμετέχοντες ήταν διευθυντές του έργου ή της ομάδας, μηχανικοί απαιτήσεων και προγραμματιστές λογισμικών.

Η μετέπειτα συζήτηση της πρακτικής τελευταίας τεχνολογίας είναι οργανωμένη γύρω από τα εξής ζητήματα: χαρακτηριστικά προϊόντος και έργου, πηγές απαιτήσεων και δραστηριότητες εκμαίευσης, προϊόντα της μηχανικής απαιτήσεων (π.χ. προδιαγραφή απαιτήσεων λογισμικού), επιβεβαίωση απαιτήσεων και έγκριση των δραστηριοτήτων και εξέλιξη του συστήματος (δηλαδή περαιτέρω δραστηριότητες ανάπτυξης).

5.3. Χαρακτηριστικά προϊόντος και έργου

Οι πρακτικές της μηχανικής των απαιτήσεων που παρατηρήσαμε διαφέρουν ανάμεσα στις SMEs. Ανάμεσα στους πιο σημαντικούς παράγοντες περιλαμβάνονται:

1. Τύπος συστήματος: καθοδηγούμενο από την αγορά vs καθοδηγούμενο από τον πελάτη σύστημα.
2. Βαθμός προσαρμοστικότητας: ρύθμιση από τον χρήστη vs ρύθμιση από τον πωλητή.
3. Τύπος ανάπτυξης λογισμικού: εσωτερική vs υπεργολαβία.

Η πλειοψηφία των SMEs (9 από τις 10) αναπτύσσουν συστήματα που καθοδηγούνται από την αγορά. Μία εταιρία αναπτύσσει καθορισμένα από τον πελάτη συστήματα για εσωτερικές εφαρμογές. Ήταν ορατές δύο διαφορετικές στρατηγικές ανάπτυξης προϊόντος για τα καθοδηγούμενα από την αγορά συστήματα: 1) η έναρξη από ένα καθοδηγούμενο από την αγορά σύστημα και 2) η γενίκευση μίας λύσης για έναν αρχικό πελάτη στις ανάγκες άλλων παρόμοιων πελατών. Οι SMEs που επέλεξαν την δεύτερη στρατηγική βρίσκονταν σε διαφορετικά στάδια της διαδικασίας γενίκευσης. Ορισμένες επιχειρήσεις βρίσκονταν στο αρχικό στάδιο, αναπτύσσοντας ένα σύστημα με την προοπτική μετέπειτα γενίκευσης του, ενώ άλλες είχαν ήδη μία γενικευμένη λύση.

Όλες οι SMEs αναπτύσσουν προϊόντα λογισμικού, τα οποία πρέπει να προσαρμόζονται στις ανάγκες των πελατών και των χρηστών. Ο βαθμός της προσαρμοστικότητας ποικίλει από τα ζητήματα ρύθμισης από τον χρήστη (όπως η εμφάνιση μίας διεπιφάνειας χρήστη) έως στα ζητήματα ρύθμισης από τον έμπορο (όπως η συγκεκριμένη λειτουργικότητα που απαιτείται από τον πελάτη). Το μέγεθος και η πολυπλοκότητα του προϊόντος λογισμικού είναι αρκετά μεγάλα, ενώ μία SME ανέφερε ότι το λογισμικό της έχει 2.000.000 γραμμές κωδικού. Επίσης, η λογική της εφαρμογής (π.χ. αρχιτεκτονική πληροφοριών, χώρος ή πιθανές αλληλεπιδράσεις χαρακτηριστικών) είναι ιδιαίτερα περίπλοκη και στις επιχειρήσεις και στους χώρους τεχνολογικής εφαρμογής.

Τα έργα των SMEs οφείλονται συνήθως σε 1) τεχνολογικές αλλαγές (π.χ. νέων λειτουργικό, νέο υλικό hardware), 2) νέα χαρακτηριστικά που πρέπει να ενσωματωθούν λόγω πίεσης της αγοράς ή 3) προσαρμογές ξεχωριστές για τον κάθε πελάτη.

Δύο από τις SMEs είχαν εμπειρία στην ανάπτυξη συστημάτων μέσω υπεργολαβίας για την Ευρώπη και τις Ηνωμένες Πολιτείες.

3 Πηγές απαιτήσεων και δραστηριότητες εκμαίευσης

Οι πηγές απαιτήσεων είναι οι πελάτες και οι χρήστες, η παρατήρηση της αγοράς και οι ειδικοί του χώρου. Ο βαθμός της εμπλοκής του χρήστη/πελάτη εξαρτάται από τον τύπο του συστήματος (δηλαδή σύστημα που καθορίζεται από τον πελάτη ή από την αγορά). Εκτός από μία συγκεκριμένη SME, η οποία αναπτύσσει λογισμικό συγκεκριμένο για κάθε πελάτη, καμία SME δεν ανέφερε έντονη εμπλοκή του πελάτη/χρήστη.

Η στρατηγική ανάπτυξης προϊόντος για συστήματα που καθορίζονται από την αγορά έχει αντίκτυπο στην διαδικασία εκμαίευσης των απαιτήσεων. Οι απαιτήσεις επινοούνται από την ίδια την εταιρία στην περίπτωση που ξεκινά με ένα προϊόν που καθορίζεται από την αγορά. Στην αρχική φάση της διαδικασίας γενίκευσης, πραγματοποιούνται συνεντεύξεις με το υπόδειγμα πελάτη και εμπλέκονται οι ειδικοί του τομέα. Πραγματοποιούνται επίσης εργαστήρια με πιθανούς και υπάρχοντες πελάτες ώστε να συγκεντρωθούν περισσότερες απαιτήσεις όσο αφορά σε ένα προϊόν που θα γενικευτεί. Ο τελικός στόχος αυτών των εργαστηρίων είναι η συμφωνία ανάμεσα στους πελάτες όσο αφορά στα επιθυμητά χαρακτηριστικά. Ένα επιπρόσθετο όφελος είναι ότι οι πιθανοί πελάτες αποκτούν μία γνώση του προϊόντος λογισμικού που θα αγοράσουν. Οι SMEs δεν ανέφεραν κάποια έντονα προβλήματα με την διαδικασία συγκέντρωσης των απαιτήσεων.

5.4 Προσδιορισμός απαιτήσεων, επαλήθευση και έγκριση

Η καταγραφή των απαιτήσεων κατά την διάρκεια του προσδιορισμού των απαιτήσεων λογισμού (SRS) εξαρτάται από τον τύπο της ανάπτυξης

λογισμικού (δηλαδή εσωτερική ή με υπεργολαβία) και από τον βαθμό προσαρμοστικότητας (ρύθμιση από χρήστη ή ρύθμιση από τον έμπορο). Οι δύο SMEs που λειτουργούν με υπεργολαβία έχουν φυσικά ένα SRS. Οι SMEs που αναπτύσσουν συστήματα ρυθμισμένα από τον έμπορο δηλώνουν μόνο τις απαιτήσεις της κάθε προσαρμογής ρυθμισμένης από τον πελάτη, διότι αυτές αποτελούν μέρος του νομικού συμβολαίου με τον πελάτη. Το κύριο σύστημα συχνά δεν έχει SRS, διότι κατασκευάστηκε στην δυναμική, αρχική φάση της SME.

Υπάρχει έντονη διαμάχη για τον βαθμό στον οποίο οι απαιτήσεις πρέπει να δηλώνονται στην περίπτωση των συστημάτων που καθορίζονται από την αγορά, τα οποία αναπτύσσονται εσωτερικά (δηλαδή στην περίπτωση όπου οι πηγές απαιτήσεων και οι προγραμματιστές λογισμικού βρίσκονται εσωτερικά), διότι τότε οι απαιτήσεις μπορούν να επικοινωνηθούν λεκτικά. Ως αποτέλεσμα, συχνά δεν υπάρχει κάποιο έγγραφο απαιτήσεων, αλλά μόνο μία συλλογή συναντήσεων και ένα μετέπειτα εγχειρίδιο χρήστη. Ορισμένες SMEs προσπαθούν να χρησιμοποιήσουν το εγχειρίδιο χρήστη ως SRS, αλλά αντιμετωπίζουν προβλήματα, διότι δεν είναι αρκετά λεπτομερές και οι προσδοκίες δεν περιλαμβάνονται συνήθως σε αυτό.

Οι SMEs που παράγουν ένα SRS αντιμετωπίζουν πιο λεπτά προβλήματα:

- Δεν είναι ξεκάθαρος ο τρόπος καθορισμού μίας γραφικής διεπιφάνειας χρήστη (GUI). Οι αναπαραστάσεις του ελέγχου ροής δεν είναι χρήσιμες, λόγω του μεγάλου αριθμού πιθανών ελέγχων ροών.
- Η πολυπλοκότητα των εγγράφων απαιτήσεων δυσκολεύει την κατανόηση και επισκόπηση τους. Η πολυπλοκότητα είναι ακόμη μεγαλύτερη, εάν το έγγραφο έχει γραφτεί με συντάκτη κειμένου και οι απαιτήσεις δεν έχουν στηριχθεί επαρκώς (όπως θα γινόταν από ένα εργαλείο διαχείρισης των απαιτήσεων).

Παρατηρήσαμε τεχνικές μοντελοποίησης μόνο σε δύο SMEs, αλλά αυτές εφαρμόζονταν σε τελείων διαφορετικούς στόχους. Η πρώτη SME χρησιμοποιεί μοντέλα για να βρίσκει ατέλειες στο SRS, αλλά το RSR

παραμένει η κύρια έξοδος δεδομένων της φάσης απαιτήσεων. Η δεύτερη SME συλλαμβάνει τις απαιτήσεις σε μοντέλα, τα οποία αποτελούν την κύρια έξοδο δεδομένων για τις δραστηριότητες ανάπτυξης. Η πρώτη SME εξομοιώνει τα μοντέλα απαιτήσεων και αναπτύσσει περιπτώσεις τεστ ώστε να αυξήσει περαιτέρω την ποιότητα του SRS. Πέρα από αυτή την εταιρία, δεν πληροφορηθήκαμε για κάποια συστηματική επιθεώρηση, εξομοίωση ή έγκριση στην φάση απαιτήσεων.

5.5 Εξέλιξη του συστήματος

Οι SMEs, οι οποίες δεν παράγουν ένα SRS, δεν αντιμετώπισαν κάποια προβλήματα κατά την διάρκεια της υλοποίησης του λογισμικού. Φαίνεται ότι οι προγραμματιστές λογισμικού αγαπούν την ‘δημιουργική ελευθερία’. Τα προβλήματα ξεκινούν με τον έλεγχο, διότι το πρώτο έργο του προσωπικού ελέγχου είναι η επανάληψη της εκμαίευσης των απαιτήσεων ώστε να δημιουργηθούν περιπτώσεις τεστ. Έτσι, πρέπει να δοθεί μεγάλη προσπάθεια στον έλεγχο και στην επανάληψη της εργασίας, διότι κατά την διάρκεια του ελέγχου εμφανίζονται εννοιολογικά προβλήματα, τα οποία θα εμφανιστούν κατά την διάρκεια της φάσης των απαιτήσεων. Η λύση των εννοιολογικών προβλημάτων είναι δύσκολη. Όσο αφορά στην φάση της συντήρησης, αναφέρθηκε ένα πρόβλημα: η υλοποίηση νέων απαιτήσεων μπορεί να προκαλέσει απρόβλεπτες αλληλεπιδράσεις με τις απαιτήσεις που έχουν ήδη ενσωματωθεί (ένα SRS μπορεί να βοηθήσει στον εντοπισμό αυτών των αλληλεπιδράσεων πριν υλοποιηθεί και ελεγχθεί μία νέα απαίτηση). Δεν αναφέρθηκαν άλλα προβλήματα, τα οποία οφείλονται σε αυτή την φάση, γεγονός που μπορεί να οφείλεται στον βαθμό αυξομοίωσης των εργαζομένων, ο οποίος δεν είναι τόσο μεγάλος όσο στις μεγάλες εταιρίες. Η πρόσθεση νέων ατόμων σε ένα έργο είναι και αυτή ιδιαίτερα δύσκολη. Λόγω της έλλειψης τεκμηρίωσης, πρέπει να εκπαιδευτούν από έμπειρα άτομα. Ένας συμμετέχοντας υποστήριξε ότι απαιτείται ένας χρόνος

για την συμμετοχή ενός νέου μηχανικού λογισμικού, ενώ απαιτείται και εκπαίδευση από έμπειρα μέλη της ομάδας.

Πρέπει να αναφέρουμε ξανά ότι οι SMEs που παράγουν ένα SRS αντιμετωπίζουν πιο λεπτά προβλήματα. Για παράδειγμα:

- Η γνώση του τομέα, η οποία περιέχεται στις απαιτήσεις, αυξάνει την δυσκολία κατανόησης του SRS από τους προγραμματιστές λογισμικού. Επιπροσθέτως, οι προγραμματιστές υποστηρίζουν ότι τα έγγραφα απαιτήσεων δεν είναι αρκετά ακριβή και συχνά δεν είναι ολοκληρωμένα. Αυτό το πρόβλημα γίνεται μεγαλύτερο εάν το λογισμικό δεν έχει αναπτυχθεί στην περιοχή και έτσι οι ατέλειες δεν έχουν ανιχνευτεί κατά την διάρκεια του ελέγχου αποδοχής.
- Οι απαιτήσεις είναι πολύ αόριστες ή κοινότητες και δεν μπορούν να ελεγχθούν. Έτσι, οι μηχανικοί απαιτήσεων πρέπει να εργαστούν και ως ελεγκτές, καθώς γνωρίζουν τον τρόπο που πρέπει να συμπεριφερθεί το σύστημα.
- Οι απαιτήσεις δεν είναι ανιχνεύσιμος (δηλαδή μία αλλαγή σε μία απαίτηση θα επηρεάσει όλα τα στοιχεία του συστήματος).

Όπως έχει αναφερθεί παραπάνω, οι περισσότερες από τις SMEs αναπτύσσουν προσαρμόσιμα προϊόντα λογισμικού. Παρόλα αυτά, καμία από τις SMEs δεν έχει μία στρατηγική για την αντιμετώπιση της προσαρμοστικότητας (π.χ. μία προσέγγιση γραμμής προϊόντος). Αυτές οι SMEs, οι οποίες ανέπτυξαν και βελτίωσαν τα προϊόντα τους με το πέρασμα των χρόνων ανέπτυξαν πρακτικούς τρόπους για την βεβαίωση της προσαρμοστικότητας. Ορισμένες SMEs έλυσαν το ζήτημα στο αρχιτεκτονικό επίπεδο διατηρώντας ένα κύριο σύστημα και υλοποιώντας προσαρμογές βασισμένες στους χρήστες ως επιπρόσθετα στοιχεία. Άλλες έλυσαν το ζήτημα αυτό στο επίπεδο υλοποίησης μέσω των αρχείων διευθέτησης, τα οποία διαβάζονται από το λογισμικό ενώ λειτουργεί. Ο κάθε τρόπος εξαρτάται από τον χώρο εφαρμογής και τον απαιτούμενο βαθμό προσαρμοστικότητας.

5.6. Διαδικασία βελτίωσης της μηχανικής των απαιτήσεων

Οι παράγοντες καθοδήγησης για την διαδικασία βελτίωσης της μηχανικής απαιτήσεων για τις SMEs αποτελούν τα προβλήματα για τον έλεγχο και την πιστοποίηση ISO 9001. Τα κύρια εμπόδια για την μεταφορά τεχνολογίας είναι οι μικρές χρηματοδοτήσεις και τα χρονοδιαγράμματα των έργων. Όταν το προϊόν λογισμικού έχει ωριμάσει στην αγορά, η πίεση βελτίωσης των δραστηριοτήτων βελτίωσης των απαιτήσεων για αυτό το προϊόν δεν είναι τόσο μεγάλη.

Ο βασικός στόχος του εργαστηρίου ήταν να καλυφθεί ολόκληρος ο κύκλος ζωής της μηχανικής απαιτήσεων ξεκινώντας από την εκμαίευση και την τεκμηρίωση και καταλήγοντας με την έγκριση/επαλήθευση, καθώς και η παρουσίαση όλων των οδηγιών/προτύπων της μηχανικής απαιτήσεων, των τεχνικών, των μεθόδων και των εργαλείων. Από αυτό τον στόχο δημιουργήσαμε έξι θέματα για το εργαστήριο, το οποία παρουσιάζονται στον Πίνακα 2.

Πίνακας 2. Θέματα για το εργαστήριο

Θέμα RE	**Λεπτομέρειες**
Βελτίωση SRS	Κινητοποίηση για ένα SRS
	Οικονομικές πλευρές (εξοικονόμηση για δραστηριότητες εργασίας μετά τον έλεγχο)
	Περιεχόμενα (γλωσσάριο κτλ)
	Δομή (π.χ. IEEE-Std. 830-1993)
	Ιδιότητες (ορθότητα, αρτιότητα, συνέπεια, σαφήνεια, ανιχνευσιμότητα κτλ.)
Εκμαίευση	Αναγνώριση των πηγών απαιτήσεων
	Τεχνικές εκμαίευσης (συνεντεύξεις, εργαστήρια, μελέτες κτλ)

	Σχεδιασμός και πραγματοποίηση συνεντεύξεων
	Χρήση περιπτώσεων για την καταγραφή των απαιτήσεων
Επιθεωρήσεις	Διαδικασία επιθεώρησης (σχεδιασμός, προετοιμασία/ανάγνωση, συνάντηση, εργασία, παρακολούθηση)
	Τεχνικές ανάγνωσης
Απαιτήσεις Μοντελοποίησης	Κινητοποίηση και οφέλη
	Μοντέλα και πρακτικοί συμβολισμοί (μοντέλα περιβάλλοντος/διαγράμματα πλαισίου, αρχιτεκτονικά μοντέλα/διαγράμματα ομάδων, συμπεριφορικά μοντέλα/γραφήματα ακολουθίας κτλ)
	Επισκόπηση μεθόδων (ΟΟ μοντελοποίηση και ειδικευμένες μέθοδοι π.χ. Statecharts)
Διαχείριση απαιτήσεων με εργαλεία	Αρχές της διαχείρισης απαιτήσεων (γραμμές αναφοράς, ανιχνευσιμότητα κτλ)
	Επισκόπηση εργαλείων
	Εισαγωγή σε ένα συγκεκριμένο εργαλείο διαχείρισης απαιτήσεων
Νομικές πλευρές της υποβολής και της υπεργολαβίας	Αρχές
	Τύποι και στοιχεία συμβολαίων (Γερμανικός νόμος)
	Προβλήματα με συμβόλαια σταθερής τιμής ενώ το SRS δεν έχει λάβει τελικό σχήμα

5.7. Μελέτες περιπτώσεων και πειράματα

Όλα τα θέματα του εργαστηρίου κινητοποιήθηκαν από μικρές μελέτες περιπτώσεων και πειράματα, με μία μελέτη περίπτωσης να πραγματοποιείται ανά συνάντηση. Πραγματοποιήθηκε ένα πείραμα, το οποίο απαίτησε δύο συναντήσεις.

5.7.1 Μελέτη περίπτωσης 'Ταξινόμηση και Έλεγχος Απαιτήσεων'

Ο στόχος ήταν η κινητοποίηση ενός καλώς δομημένου SRS και η παροχή παραδειγμάτων σχετικά με τις 'καλές' και 'κακές' απαιτήσεις. Ο κάθε συμμετέχοντας έλαβε μία μη δομημένη δήλωση απαιτήσεων, ένα διάγραμμα SRS και μία φόρμα καταγραφής των ατελειών των απαιτήσεων. Το έργο των συμμετεχόντων ήταν να ταξινομήσουν τις απαιτήσεις σύμφωνα με τυπικές κατηγορίες (λειτουργικές απαιτήσεις, μη λειτουργικές απαιτήσεις, ιδιοκτησία χώρου κτλ), να τις τοποθετήσουν στο κατάλληλο τμήμα του SRS και να ελέγξουν τις απαιτήσεις για τις συνηθισμένες ιδιότητες.

Το αποτέλεσμα της μελέτης περίπτωσης ήταν ότι αναγνωρίστηκε η αξία ενός καλώς δομημένου SRS σε σύγκριση με ένα μη δομημένο έγγραφο. Παρόλα αυτά, δεν οδηγηθήκαμε σε συμφωνία σχετικά με το τμήμα που θα τοποθετηθεί η κάθε απαίτηση. Μπορέσαμε να βρούμε μόνο ορισμένες ατέλειες ανά απαίτηση.

5.7.2 Μελέτη περίπτωσης 'Εκμαίευση και Επεξεργασία των Απαιτήσεων'

Ο στόχος αυτής της μελέτης περίπτωσης ήταν η κινητοποίηση των συστηματικών μεθόδων ώστε να εκμαιεύσουν και να επεξεργαστούν τις απαιτήσεις ενός συστήματος. Οι συμμετέχοντες χωρίστηκαν σε 'πελάτες' και 'μηχανικούς απαιτήσεων'. Οι πελάτες έλαβαν μία πλήρη περιγραφή ενός συστήματος και οι μηχανικοί απαιτήσεων έλαβαν μόνο μία γενική περιγραφή. Ο στόχος των μηχανικών απαιτήσεων ήταν η προετοιμασία και η πραγματοποίηση μίας συνέντευξης με τον πελάτη ώστε να καθορίσουν τις απαιτήσεις. Ο στόχος των πελατών ήταν να απαντήσουν στις ερωτήσεις.

Το αποτέλεσμα ήταν ότι οι περισσότεροι συμμετέχοντες συμφώνησαν όσο αφορά στις συστηματικές μεθόδους της επεξεργασίας των απαιτήσεων.

Δεν ήταν ικανοί να επεξεργαστούν λεπτομερώς τις περισσότερες απαιτήσεις του συστήματος μέσα στο χρονικό πλαίσιο, διότι δεν αναπτύχθηκε μία ολοκληρωμένη άποψη του συστήματος. Χρησιμοποιήσαμε αυτή την μελέτη περίπτωσης για να κινητοποιήσουμε την χρήση περιπτώσεων, αλλά παραμένει άλυτο το εάν η χρήση αυτή μπορεί να βοηθήσει σε μεγάλο βαθμό, διότι η γνώση του χώρου παίζει σημαντικό ρόλο στην επεξεργασία των απαιτήσεων.

5.7.3 Πείραμα 'Έγκριση Απαιτήσεων'

Ο στόχος του πειράματος ήταν να συγκρίνει τις διαφορετικές τεχνικές εύρεσης των ατελειών στα αρχεία απαιτήσεων. Οι τεχνικές αυτές είναι οι επιθεωρήσεις (με λίστες) και η μοντελοποίηση των απαιτήσεων. Στην πρώτη συνάντηση, οι συμμετέχοντες έλαβαν ένα έγγραφο απαιτήσεων, μία λίστα και μία φόρμα ατελειών. Ο στόχος ήταν η χρήση της λίστας. Στην δεύτερη συνάντηση ο στόχος των συμμετεχόντων ήταν να δημιουργήσουν μία ομάδα μοντέλων για το SRS. Έπρεπε ξανά να καταγράψουν όλες τις ατέλειες στην φόρμα.

Οι συμμετέχοντες ανακάλυψαν πολλές ατέλειες με την χρήση της φόρμας, αν και κυρίως αυτές ήταν υπηρεσιακές ή προφανείς. Είναι ενδιαφέρον το γεγονός ότι οι περισσότερες ατέλειες αναφέρθηκαν από έναν συμμετέχοντα, γεγονός που αποδεικνύει ότι είναι δύσκολο να ορίσουμε την ατέλεια σε αντίθεση με τον κώδικα. Η δημιουργία μοντέλων απαιτήσεων ανάγκασε τους συμμετέχοντες να αναλύσουν σε βάθος τις απαιτήσεις και έτσι ανακάλυψαν πιο ουσιώδεις ατέλειες. Οι συμμετέχοντες διάβασαν δύο φορές το SRS , γεγονός στο οποίο μπορεί να οφείλεται η ανακάλυψη ζωτικών ατελειών κατά την δεύτερη φορά.

5.7.4 Μελέτη περίπτωσης 'Στήριξη εργαλείων'

Ο στόχος ήταν η εξερεύνηση της αξίας της ανιχνευσιμότητας των απαιτήσεων και η δυνατότητα μίας βάσης δεδομένων και χειρίζεται τις απαιτήσεις και τις σχέσεις τους. Οι συμμετέχοντες έλαβαν σύντομη εκπαίδευση με εργαλείο διαχείρισης των εμπορικών απαιτήσεων. Ο στόχος ήταν η ανάλυση του αντίκτυπου αρκετών απαιτήσεων καθώς και η ανίχνευση των πληροφοριών υλοποίησης στο στάδιο απαιτήσεων. Άλλοι στόχοι περιλάμβαναν την εξαγωγή πληροφοριών που είναι απαραίτητες για συγκεκριμένα έργα, τον έλεγχο της πρόσβασης και την δημιουργία γραμμής αναφοράς για τις απαιτήσεις.

Η ανιχνευσιμότητα θεωρήθηκε χρήσιμη, αλλά εμφανίστηκαν διαφωνίες σχετικά με την εισαγωγή και διατήρηση της. Η εφαρμογή εργαλείων διαχείρισης των απαιτήσεων θεωρήθηκε αρκετά χρήσιμη.

5.8. Βαθμολόγηση των θεμάτων RE

Μετά την ολοκλήρωση του εργαστηρίου, ζητήσαμε από τους συμμετέχοντες να βαθμολογήσουν το περιεχόμενο σύμφωνα με δύο ερωτήσεις: 1) Ποια θέματα είναι σχετικά/άσχετα με το δικό σας περιβάλλον;, 2) Ποιο θέμα έχει την μεγαλύτερη προτεραιότητα για το δικό σας περιβαλλοντικό πλάνο; . Ο Πίνακας 3 παρουσιάζει τα αποτελέσματα. Οι αριθμοί στην δεύτερη, Τρίτη και τέταρτη στήλη υποδεικνύουν τον αριθμό των συμμετεχόντων που θεώρησαν ένα θέμα σχετικό ή που έδωσαν την μεγαλύτερη προτεραιότητα σε ένα θέμα.

Πίνακας 3. Αποτελέσματα

Θέμα	**Σχετικότητα**	**Μη σχετικότητα**	**Υψηλότερη προτεραιότητα**
Βελτίωση SRS	10	2	2

Εκμαίευση	7	5	1
Επιθεωρήσεις	9	3	2
Μοντελοποίηση	12	0	5
Εργαλεία	7	5	2
Νομικές πλευρές	4	8	0

Η παρακάτω ιεραρχία των θεμάτων όσο αφορά στην σχετικότητα και στην προτεραιότητα μπορεί να δημιουργηθεί από τον Πίνακα 2:

- Σχετικότητα: Μοντελοποίηση > Βελτίωση SRS > Επιθεωρήσεις > Εργαλεία = Εκμαίευση > Νομικές πλευρές.
- Προτεραιότητα: Μοντελοποίηση >> Βελτίωση SRS = Επιθεωρήσεις = Εργαλεία > Εκμαίευση > Νομικές πλευρές.

Οι ‘νομικές πλευρές’ και η ‘εκμαίευση’ βαθμολογήθηκαν ως τα λιγότερο σχετικά, γεγονός που μπορεί να οφείλεται στο ότι οι περισσότερες SMEs αναπτύσσουν συστήματα που κατευθύνονται από την αγορά εσωτερικά. Οι εταιρίες που έδιναν την ανάπτυξη του λογισμικού με υπεργολαβία δεν αντιμετώπισαν κάποια σημαντικά προβλήματα όσο αφορά στις νομικές πλευρές. Η στήριξη εργαλείων θεωρήθηκε αρκετά χρήσιμη, αλλά όχι και η πιο ζωτική, γεγονός που μπορεί να οφείλεται στο ότι οι περισσότερες SMEs δεν έχουν εκτεταμένο SRS. Η ‘μοντελοποίηση’ είναι το πιο σχετικό θέμα, το οποίο επιθυμούν να βελτιώσουν οι περισσότεροι συμμετέχοντες. Η ικανοποίηση είναι η αιτία της επιθυμίας εισαγωγής ενός νέου (ή βελτιωμένου) SRS, το οποίο είναι η κύρια αναγκαιότητα για όλες τις δραστηριότητες στην μηχανική απαιτήσεων.

5.9. Περίληψη και Συμπεράσματα

Σε αυτό το άρθρο παρουσιάσαμε την πρακτική τελευταίας τεχνολογίας και τα προβλήματα των Μικρών και Μεσαίων Επιχειρήσεων (SMEs) όσο αφορά στην μηχανική απαιτήσεων, όπως παρατηρήθηκε σε

εργαστήριο με συμμετοχή 10 SMEs. Παρουσιάσαμε αρκετές βελτιώσεις και την αξιολόγηση τους από τους συμμετέχοντες.

Οι περισσότερες SMEs αναπτύσσουν συστήματα που κατευθύνονται από την αγορά, τα οποία αποτελούν νέα περιοχή για την έρευνα της μηχανικής απαιτήσεων. Οι περισσότερες SMEs δεν έχουν εκτεταμένο SRS. Αναπτύσσουν τα συστήματα τους εσωτερικά και έτσι μπορούν να επικοινωνούν τις απαιτήσεις λεκτικά. Από την άλλη, διαθέτουν αρκετό χρόνο στον έλεγχο, ώστε να δημιουργούν περιπτώσεις ελέγχου. Το πρώτο έργο του προσωπικού ελέγχου είναι η επανάληψη της εκμαίευσης των απαιτήσεων (καθώς μόνο λίγες απαιτήσεις τεκμηριώνονται). Η ποσότητα της εργασίας είναι μεγάλη, διότι τα εννοιολογικά προβλήματα ανιχνεύονται κατά την διάρκεια του ελέγχου. Τα προβλήματα αυτά θα μπορούσαν να ανιχνευτούν και να αφαιρεθούν κατά την διάρκεια της ανάλυσης απαιτήσεων με πολύ λιγότερη προσπάθεια. Επιπροσθέτως, η έλλειψη τεκμηρίωσης δυσκολεύει την πρόσθεση νέων ατόμων στο έργο. Όλοι οι συμμετέχοντες συμφώνησαν για την σημασία των SRS αλλά δημιουργήθηκε ένα σημαντικό ερώτημα: πόσο λεπτομερές πρέπει να είναι το SRS εάν το λογισμικό δημιουργείται εσωτερικά; Αντί για την λεπτομερή περιγραφή των απαιτήσεων σε γραπτή μορφή, οι SMEs ενδιαφέρονται έντονα για την απαιτήσεις μοντελοποίησης, καθώς τα μοντέλα μπορούν να επαναχρησιμοποιηθούν (σε ένα βαθμό) για τον σχεδιασμό. Παρόλα αυτά, η προσέγγιση μοντελοποίησης που θα χρησιμοποιηθεί για ένα συγκεκριμένο έργο και η εφαρμογή της για την μεγιστοποίηση της αποδοτικότητας αποτελούν αναπάντητα ερωτήματα, όπως και ο καθορισμός των γραφικών διεπιφανειών χρηστών με έναν πρακτικό τρόπο.

Οι SMEs επιθυμούν να παράγουν ένα SRS με μικρή επιπρόσθετη προσπάθεια. Καθώς ένα εγχειρίδιο χρήστη πρέπει να παραχθεί έτσι και αλλιώς, μπορεί να αξίζει να μελετηθεί η σχέση ανάμεσα στο SRS και το εγχειρίδιο αυτό. Εάν το SRS είναι μία υπερομάδα του εγχειριδίου χρήστη, τότε το SRS πρέπει να οργανωθεί παρόμοια με τέτοιο τρόπο ώστε το εγχειρίδιο να παράγεται από αυτό αυτόματα. Θα απαιτούνταν ελάχιστη

προσπάθεια για την συμπλήρωση των τμημάτων του SRS που δεν περιλαμβάνονται στο εγχειρίδιο του χρήστη.

Το προϊόν λογισμικού μίας SMEs αναπτυσσόταν συχνά κατά την διάρκεια της πρώτης φάσης της εταιρίας και έτσι δεν υπήρχε τεκμηρίωση του βασικού συστήματος αλλά μόνο των πρόσφατων προσαρμογών. Έτσι, ο τρόπος κατασκευής ενός εγγράφου απαιτήσεων αποτελεί ανοιχτή ερώτηση.

Θεωρήσαμε ότι οι SMEs είναι ένας ενδιαφέρων τομέας για έρευνα μηχανικής απαιτήσεων, διότι το μέγεθος και η πολυπλοκότητα των συστημάτων τους είναι συγκρίσιμα με αυτά των μεγάλων εταιριών. Παρόλα αυτά, οι μικρότεροι προϋπολογισμοί απαιτούν μικρότερα βήματα βελτίωσης. Καθώς τα περισσότερα έργα των SMEs στοχεύουν στην προσαρμογή και στην ενσωμάτωση νέων χαρακτηριστικών επιθυμητών από την αγορά, η βελτίωσης της διαδικασίας RS πρέπει να είναι αυξητική και εξελικτική και όχι επαναστατική. Η επίδειξη των οφελών των τεχνολογιών RE αποτελεί ζωτικό βήμα για την διατήρηση των βελτιώσεων της διαδικασίας στο περιβάλλον μίας εταιρίας.

ΚΕΦΑΛΑΙΟ 6^{Ο}
ΜΗΧΑΝΙΚΗ ΤΩΝ ΑΠΑΙΤΗΣΕΩΝ ΓΙΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΤΩΝ ΕΠΙΧΕΙΡΗΣΕΩΝ: ΕΠΤΑ ΠΡΟΚΛΗΣΕΙΣ ΣΕ ΠΕΡΙΒΑΛΛΟΝ ΑΝΩΤΑΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ

Γενικά – Αυτό το άρθρο περιγράφει τις προκλήσεις της μηχανικής των απαιτήσεων όσο αφορά στην ανάπτυξη εφαρμογών μίας επιχείρησης του τομέας της ανώτατης εκπαίδευσης. Οι δραστηριότητες ανάπτυξης περιλαμβάνουν την υλοποίηση λογισμικού, την συντήρηση και την ενίσχυση και στήριξη της απευθείας επεξεργασίας συναλλαγών και της διαδικασίας επεξεργασίας συνόλου δεδομένων. Γενικά, μία επιχειρηματική εφαρμογή περιβάλλοντος ανώτατης εκπαίδευσης μπορεί να περιλαμβάνει ένα Σύστημα Πληροφοριών των Μαθητών (SIS), ένα σύστημα HR/Payroll, τα Οικονομικά Συστήματα κτλ. Υπάρχουν πολλές προκλήσεις όσο αφορά στις φάσεις της μηχανικής των απαιτήσεων εάν επιθυμούμε να παρέχουμε δύο ξεχωριστές υπηρεσίες, δηλαδή την στήριξη της διαδικασίας παραγωγής και την ανάπτυξη συστημάτων.

Λέξεις κλειδιά: Ανάπτυξη επιχειρηματικών εφαρμογών, συστήματα επιχειρηματικών πληροφοριών, επιχειρηματική διαδικασία, μηχανική των απαιτήσεων, πρότυπα απαιτήσεων δραστηριότητες ανάπτυξης λογισμικού, ανασκόπηση των απαιτήσεων λογισμικού.

6.1. Εισαγωγή

Ο στόχος της μηχανικής των απαιτήσεων είναι να βεβαιώσει ότι η διαδικασία της ανάπτυξης μίας εφαρμογής είναι ικανή να ικανοποιήσει τις

ανάγκες των πελατών με την ελάχιστη προσπάθεια όσο αφορά στον χρόνο και το κόστος. Γενικά, μία επιχειρηματική εφαρμογή για περιβάλλον ανώτατης εκπαίδευσης μπορεί να περιλαμβάνει ένα Σύστημα Πληροφοριών Μαθητών (SIS), ένα σύστημα HR/Payroll, Οικονομικά Συστήματα κτλ. Υπάρχουν πολλές προκλήσεις στις φάσεις της μηχανικής των απαιτήσεων εάν επιθυμούμε να παρέχουμε δύο ξεχωριστές υπηρεσίες, οι οποίες είναι η στήριξη της επεξεργασίας παραγωγής και η ανάπτυξη συστημάτων. Η επεξεργασία παραγωγής παρέχει στήριξη για συστήματα που αποτελούν τα θεμέλια της τεχνικής υποδομής, για την διαδικασία επεξεργασίας ενός συνόλου δεδομένων και για την ασφάλεια, ενώ η ανάπτυξη συστημάτων παρέχει στήριξη προγραμματισμού για τις υλοποιήσεις συστημάτων καθώς και για την συντήρηση του λογισμικού.

'Ο κόσμος θα ήταν απλούστερος εάν οι απαιτήσεις σταματούσαν να έρχονται όταν ένα έργο έχει συλλέξει όλες τις απαιτήσεις που υπήρχαν και τις πέρασε από επισκόπηση με τους συμμετόχους. Στο παρελθόν, οι μηχανικοί πίστευαν ότι μόνο αυτό έπρεπε να κάνουν και εκτελούσαν έργα με φάσεις, οι οποίες ονομάζονταν Απαιτήσεις Χρηστών, Ανάλυση Συστημάτων, Αρχιτεκτονική Συστημάτων κτλ. Μετά το τέλος της πρώτης φάσης οι απαιτήσεις παγώνονταν και όλοι απλά έκλειναν τα μάτια τους ελπίζοντας ότι όλα είναι σωστά και ολοκληρωμένα. Δεν είναι όμως τόσο εύκολο. Ακόμη και εάν οι άνθρωποι συμφωνήσουν σε μία συνάντηση, εμφανίζονται αλλαγές, οι οποίες μπορούν να αυξήσουν το κόστος, τον χρόνο και την πολυπλοκότητα του έργου. Οι αλλαγές της τελευταίας στιγμής αποτελούν ιδιαίτερο πρόβλημα, διότι οδηγούν στην απόρριψη της προηγούμενης εργασίας, καθώς και στην πρόσθεση νέων στοιχείων. Το καλό έργο είναι αυτό που μπορεί να αντιμετωπίσει την αλλαγή (Ian Alexander 2002).

Βάσει της παραπάνω δήλωσης, μπορούμε να συμπεράνουμε ότι στην σύγχρονη εποχή τα άτομα που αναπτύσσουν συστήματα προσαρμόζονται στην ανάπτυξη και παράδοση συστημάτων και πολλοί από αυτούς έχουν αποδεχτεί αυτή την πρόκληση με απροθυμία και αβεβαιότητα. Αυτό συμβαίνει και στο περιβάλλον της ανώτατης εκπαίδευσης.

6.2. Επτά προκλήσεις σε ποικίλες δραστηριότητες

Α. Πρώτη πρόκληση: Η κατανόηση του τομέα του προβλήματος

Το πρόβλημα της επιχειρηματικής εφαρμογής έχει πολλά συμπτώματα: συστήματα που δεν μπορούν να συνεργαστούν, μεγάλη ποσότητα δεδομένων και μικρή ποσότητα πληροφοριών, ασύμφωνα και λανθασμένα δεδομένα και μεγάλα κόστη συντήρησης. Το 70% με 90% του προϋπολογισμού του πανεπιστημίου IT δαπανάται σε τρέχοντες εφαρμογές. Υπάρχουν πολλοί λόγοι για αυτό, αλλά η κύρια αιτία είναι ο τρόπος ανάπτυξης αυτών των εφαρμογών:

- Τα συστήματα είναι ανεξάρτητα και δεν είναι σχεδιασμένα ώστε να λειτουργούν διαλειτουργικά.
- Δεν είναι βασισμένα σε ένα επιχειρηματικό πρότυπο.
- Δεν υπάρχει μία κοινή αρχιτεκτονική δεδομένων, η οποία θα βεβαιώνει την κοινή χρήση δεδομένων.
- Έχουν χρησιμοποιηθεί μη δομημένες γλώσσες προγραμματισμού.
- Δεν υπάρχει τεκμηρίωση.
- Τα συστήματα είναι σχεδιασμένα για να αντικατοπτρίζουν την ερμηνεία των επιχειρηματικών απαιτήσεων πριν από δέκα ή είκοσι χρόνια ή και ακόμη παλαιότερα.

Όλες αυτές οι προσεγγίσεις έχουν αξίας και ορισμένες μπορούν να παρέχουν ένα τουλάχιστον προσωρινό όφελος. Παρόλα αυτά, εάν δεν έχουν ως στόχο την επιχείρηση και δεν βασίζονται σε ένα μοντέλο, είναι πιθανό ότι θα επιδεινώσουν τα προβλήματα και δεν θα παρέχουν μία λύση. Η παρανόηση και η έλλειψη κατανόησης του τομέα του πραγματικού προβλήματος είναι η κύρια πρόκληση αυτής της περίπτωσης. Οι μηχανικοί λογισμικού μπορεί να κάνουν λανθασμένες υποθέσεις και έτσι να δημιουργήσουν λανθασμένες απαιτήσεις ή σχεδιασμούς. Ακόμη και οι χρήστες, οι οποίοι είναι ειδικοί σε αυτό τον τομέα, μπορεί να μην έχουν τον

τύπο των γνώσεων που μπορεί να εφαρμοστεί και έτσι να μην μπορούν να ορίσουν το πρόβλημα.

Β. Δεύτερη πρόκληση: Η αναγνώριση της επιχειρηματικής διαδικασίας

'Η επιχειρηματική διαδικασία' είναι ένας ευρύς όρος. Για τους σκοπούς αυτού του άρθρου, η επιχειρηματική διαδικασία είναι μία ομάδα δράσεων που διευκολύνουν την διεξαγωγή της επιχείρησης, η οποία είναι είτε εξωτερική είτε εσωτερική οντότητα. Αν και η χρήση συστημάτων πληροφοριών για την αυτοματοποίηση των επιχειρηματικών διαδικασιών είναι πιο αποδοτική και προσοδοφόρα σε σχέση με την πρόσληψη οικονομικών ελεγκτών, υπάρχουν αποτελέσματα που δεν έχουν κατανοηθεί πλήρως. Μία βασική αρχή στις επιχειρήσεις του τομέα ανώτατης εκπαίδευσης είναι η απαλοιφή των οποιοδήποτε σπαταλών σε όλες τις επιχειρήσεις και τις εκπαιδευτικές διαδικασίες. Ένα βασικό συστατικό της υλοποίησης και διαχείρισης ενός αποτελεσματικού περιβάλλοντος είναι η ικανότητα της αυτοματοποίησης των διαδικασιών. Το προσωπικό του ΙΤ θα δυσκολευτεί να αυτοματοποιήσει τις διαδικασίες του εκπαιδευτικού περιβάλλοντος χρησιμοποιώντας μία αυτοματοποιημένη μηχανή. Έτσι, η αναγνώριση της επιχειρηματικής διαδικασίας για αυτή την ανάπτυξη εφαρμογής αποτελεί πρόκληση.

Στο περιβάλλον ανώτατης εκπαίδευσης, οι Επιχειρηματικές Διαδικασίες για την επιχειρηματική εφαρμογή αναγνωρίζονται ως οι σειρές συνδεδεμένων λειτουργικών δραστηριοτήτων, οι οποίες λαμβάνουν δεδομένα εισόδου και παράγουν δεδομένα εξόδου. Το προσωπικό του ΙΤ πρέπει να δώσει έμφαση στο γεγονός ότι η περιγραφή μίας επιχειρηματικής διαδικασίας δεν αντιμετωπίζει τον τρόπο αλλαγής των δεδομένων εισόδου σε δεδομένα εξόδου. Περιγράφει μόνο τι κάνει η διαδικασία και όχι πως το κάνει. Αυτό εφαρμόζεται σε όλες τις περιπτώσεις, στο διοικητικό, πανεπιστημιακό και μαθητικό τμήμα.

Τρίτη πρόκληση: Η ανάδειξη των προτύπων

Η ανάδειξη του προτύπου αποτελεί μία από τις μεγάλες προκλήσεις, διότι μόνο το καλύτερο πρότυπο μπορεί να βοηθήσει στην δραστηριότητα της μηχανικής απαιτήσεων και στην ροή εργασίας. Για αυτή την μελέτη έχει ανασκοπηθεί το Πρότυπο Λογισμικού IEEE. Η υιοθέτηση των προτύπων μηχανικής του λογισμικού IEEE είναι μία πιο αποτελεσματική και προτιμώμενη προσέγγιση. Αντιπροσωπεύει επίσης την πιο κατανοητή και ώριμη ομάδα διαθέσιμων προτύπων. Παρόλα αυτά, μπορεί να είναι περίπλοκη και σε αυτό το σημείο μπορεί να βοηθήσει η υλοποίηση των Προτύπων Λογισμικού IEEE. Παράγει απαραίτητες πληροφορίες όσο αφορά στις απαιτήσεις των συστημάτων λογισμικού όπως αυτές καθορίζονται από τους πελάτες/χρήστες ή τους σχεδιαστές/παραγωγούς και αποτελεί την ουσία μίας συμφωνίας ανάμεσα τους. Η Προδιαγραφή των Απαιτήσεων του Λογισμικού (SRS) εστιάζει στην συλλογή και οργάνωση όλων των απαιτήσεων που περικυκλώνουν το έργο. Παρέχει μία πλήρη απεικόνιση του μοντέλου ανάλυσης διαδικασίας As-Is και To-Be. Το As-Is είναι η μελέτη απαιτήσεων και η ανάλυση των τρεχόντων επιχειρηματικών διαδικασιών. Το To-Be θα αποτελεί την προτεινόμενη λύση. Η To-Be ανάλυση μοντέλου επεξεργασίας θα αποτελεί τα δεδομένα εισαγωγής για την ανάλυση As-Is και θα περάσει μέσω ενός είδους επιχειρηματικής διαδικασίας. Ορίζει επίσης την λειτουργικότητα, τις συμπεριφορικές απαιτήσεις, τις εξωτερικές διεπαφές, τα χαρακτηριστικά και η απόδοση του συστήματος εφαρμογής. Αυτό το αρχείο θα χρησιμοποιηθεί στον σχεδιασμό του συστήματος. Οι ειδικευμένες απαιτήσεις αναγνωρίζονται ως μία από τις πιο δύσκολες και σημαντικές περιοχές επιχειρηματικής εφαρμογής. Ένα ιδιαίτερα ζωτικό ζήτημα είναι η έλλειψη πραγματικών παραδειγμάτων προδιαγραφών απαιτήσεων. Έτσι, αυτή είναι η πρόκληση του προσωπικού του IT.

Δ. Η τέταρτη πρόκληση: Η ανάλυση και η εκμαίευση των απαιτήσεων

Στην θεωρία, οι απαιτήσεις προέρχονται από τους χρήστες και τους συμμετόχους ενός συστήματος. Ένα μέρος της εργασίας των απαιτήσεων είναι η εκμαίευση των απαιτήσεων από τους συμμετόχους. Η υπόθεση είναι

ότι οι συμμέτοχοι έχουν ορισμένες απαιτήσεις και ο ρόλος του αναλυτή είναι να τις εκμαιεύσει, να τις αναλύσει όσο αφορά στην συνέπεια, στην δυνατότητα επίτευξης και στην πληρότητα και να τις διαμορφώσει. Οι άνθρωποι που εκμαιεύουν και διαμορφώνουν τις απαιτήσεις ονομάζονται μηχανικοί απαιτήσεων ή αναλυτές. Στην πράξη, οι αναλυτές μπορούν να είναι προγραμματιστές, ειδήμονες χρήστες (κατά προτίμηση μία ομάδα που περιλαμβάνει και τους δύο), ανεξάρτητοι σύμβουλοι και άνθρωποι μάρκετινγκ κτλ. Οι συμμέτοχοι περιλαμβάνουν χρήστες με ποικίλους ρόλους, τον πελάτη (που πληρώνει για το προϊόν), το τμήμα IT και ορισμένες άλλες ομάδες, με τις οποίες συνεργάζεται ο πελάτης. Εάν το σύστημα είναι ένα προϊόν, το οποίο προσφέρεται σε μία ευρύτερη αγορά, οι συμμέτοχοι μπορούν να περιλαμβάνουν τους διανομείς και ορισμένες φορές τις εταιρίες λογισμικού που προσθέτουν ειδικά χαρακτηριστικά σε αυτό.

Η εκμαίευση είναι μία πολύ δύσκολη διαδικασία για πολλούς λόγους:

1. Οι συμμέτοχοι μπορεί να έχουν δυσκολία στην έκφραση των αναγκών τους ή μπορεί να ζητούν μία λύση που δεν καλύπτει τις πραγματικές τους ανάγκες.
2. Οι συμμέτοχοι μπορεί να έχουν αντικρουόμενες ανάγκες.
3. Οι χρήστες μπορεί να αντιμετωπίσουν δυσκολίες όσο αφορά στο να υποθέσουν νέους τρόπους ή τα αποτελέσματα αυτών που ζητούν. Όταν, για παράδειγμα, παρατηρούν ένα σύστημα που έχει δημιουργηθεί για αυτούς, συχνά καταλαβαίνουν ότι δεν ικανοποιεί τις προσδοκίες τους, αν και ικανοποιεί τις γραπτές απαιτήσεις.
4. Ορισμένες φορές δεν υπάρχουν χρήστες διότι το προϊόν είναι καινούριο και κανείς δεν έχει χρησιμοποιήσει IT για αυτό τον σκοπό.
5. Οι απαιτήσεις συχνά αλλάζουν με το πέρασμα του χρόνου. Όταν οι χρήστες, για παράδειγμα, βλέπουν ένα έξυπνο σύστημα κάπου, μπορεί να συνειδητοποιήσουν ότι χρειάζονται κάτι παρόμοιο. Οι εξωτερικοί παράγοντες μπορούν επίσης να αλλάξουν, όπως για παράδειγμα μία δημιουργία ενός νέου λειτουργικού συστήματος ή νέων νόμων.

Ακόμη και όταν οι χρήστες μπορούν να εκφράσουν τις ανάγκες τους, οι μηχανικοί απαιτήσεων δυσκολεύονται να τις σημειώσουν χωρίς να σχεδιάζουν ταυτόχρονα και την λύση. Το αποτέλεσμα είναι ότι οι πραγματικές απαιτήσεις και οι γραπτές απαιτήσεις δεν ταυτίζονται. Για τον λόγο αυτό, είναι σημαντικό για τους συμμετόχους να ελέγχουν εάν οι απαιτήσεις ικανοποιούν τις ανάγκες τους.

Ε. Πέμπτη πρόκληση: Έγκριση, επαλήθευση και εξακρίβωση

Οι πελάτες της ανώτατης εκπαίδευσης είναι διευθυντές, ακαδημαϊκοί, προσωπικό και μαθητές. Συχνά διαφωνούν σχετικά με την διαφορά ανάμεσα στις δραστηριότητες μηχανικής των απαιτήσεων και οι τεχνικοί όροι χρησιμοποιούνται διαφορετικά. Η επαλήθευση και η έγκριση αποτελούν ορολογία που χρησιμοποιείται συχνά και υπάρχει μία χρήσιμη διαφορά ανάμεσα τους:

1. Έγκριση: Ο πελάτης πρέπει να είναι ικανός να εγκρίνει τις απαιτήσεις ώστε να συμπεράνει ότι ικανοποιούν τις ανάγκες του. Αυτό σημαίνει ότι πρέπει να μπορεί να διαβάσει τις προδιαγραφές, να τις κατανοήσει και να συμφωνήσει. Στην πρακτική είναι καλή ιδέα η έγκριση των ενδιάμεσων προϊόντων, όπως για παράδειγμα των σχεδίων ή των εικόνων, έτσι ώστε το προϊόν να καλύπτει τις προσδοκίες του πελάτη.
2. Επαλήθευση: Η επαλήθευση πραγματοποιείται για να ελεγχθεί το εάν το προϊόν ικανοποιεί τις απαιτήσεις. Η ελάχιστη επαλήθευση είναι ένα τεστ αποδοχής, κατά το οποίο οι ομάδες ελέγχουν τις απαιτήσεις και το εάν το προϊόν τις ικανοποιεί. Είναι επίσης καλή ιδέα να επαληθευτεί το εάν τα ενδιάμεσα προϊόντα ικανοποιούν τις απαιτήσεις. Οι δημιουργοί και οι πελάτες πρέπει να γνωρίζουν ότι οι απαιτήσεις μελετούνται σε όλη την διάρκεια της ανάπτυξης.
3. Εξακρίβωση: Η εξακρίβωση των απαιτήσεων είναι αναγκαία ώστε να συγκρίνονται οι απαιτήσεις με άλλες πληροφορίες. Υπάρχουν τέσσερις τύποι εξακρίβωσης:

- Προς τα εμπρός. Η εξακρίβωση από τις απαιτήσεις προς το πρόγραμμα συμπεραίνει εάν όλες οι απαιτήσεις μελετούνται. Η διαδικασία αυτή μοιάζει με την επαλήθευση. Η εξακρίβωση από την ζήτηση στις απαιτήσεις συμπεραίνει εάν η ζήτηση αντικατοπτρίζεται στις απαιτήσεις. Αυτό αποτελεί τμήμα της έγκρισης, αλλά συχνά παραμελείται και ως αποτέλεσμα χάνονται οι στόχοι της επιχείρησης.
- Προς τα πίσω. Η εξακρίβωση από τις απαιτήσεις προς την ζήτηση συμπεραίνει εάν όλες οι απαιτήσεις έχουν έναν σκοπό. Αυτό αποτελεί ένα άλλο τμήμα της έγκρισης. Η εξακρίβωση από το πρόγραμμα προς τις απαιτήσεις συμπεραίνει εάν όλα τα τμήματα του προγράμματος απαιτούνται. Αυτό δεν αποτελεί ούτε έγκριση ούτε επαλήθευση, αλλά έναν χρήσιμο έλεγχο για την αποφυγή της δημιουργίας ενός προϊόντος που δεν ζητήθηκε από τους πελάτες.
- Δικαστικές περιπτώσεις. Εάν τα πράγματα δεν πάνε καλά, ο προσδιορισμός των απαιτήσεων και το συμβόλαιο μπορούν να χρησιμοποιηθούν ως αποδείξεις της αρχικής συμφωνίας στο δικαστήριο. Πολλοί προγραμματιστές θεωρούν ότι εάν ικανοποιούν τις γραπτές απαιτήσεις θα νικήσουν και τον δικαστικό αγώνα. Παρόλα αυτά, τα δικαστήρια των περισσότερων χωρών δεν λειτουργούν με αυτό τον τρόπο. Εάν ο πελάτης έχει λογικές προσδοκίες, οι οποίες δεν περιλαμβάνονταν στον προσδιορισμό, το δικαστήριο θα αποφασίσει ότι ο παροχέας πρέπει να ικανοποιήσει αυτή την προσδοκία. Με άλλα λόγια, το δικαστήριο αναγνωρίζει τις λογικές υπονοούμενες απαιτήσεις.

Ζ. Έκτη πρόκληση: Επισκόπηση των απαιτήσεων

Η επισκόπηση των απαιτήσεων λογισμικού είναι ένα ζωτικό στοιχείο που παίζει σημαντικό ρόλο στην βεβαίωση ότι το κάθε προϊόν, το οποίο κατασκευάστηκε σύμφωνα με την Διαδικασία Κύκλου Ζωής Λογισμικού IEEE 12207 είναι ολοκληρωμένο, ακριβές και συμφωνεί με τα πρότυπα ποιότητας και ελέγχου. Δεδομένης της σημασίας τους, οι επισκοπήσεις δεν μπορούν να

παραλειφθούν χωρίς την προηγούμενη συμφωνία από το Συμβούλιο Επισκόπησης Ποιότητας. Οι επισκοπήσεις απαιτήσεων περιλαμβάνουν:

1. Τον τύπο των επισκοπήσεων του έργου.
2. Τον προγραμματισμό και την πραγματοποίηση των επισκοπήσεων.
3. Την Βεβαίωση Ποιότητας της Ροής της Διαδικασίας.
4. Τον τύπο της επισκόπησης απαιτήσεων.

Οι επισκοπήσεις απαιτήσεων μπορούν να ταξινομηθούν σε μία από τις παρακάτω κατηγορίες:

1. Επισκοπήσεις ποιότητας
2. Τεχνική επισκόπηση
3. Επισκόπηση χρήστη

Γενικά, τα άτομα που πραγματοποιούν την επισκόπηση της ποιότητας προσλαμβάνονται κατά την έναρξη του έργου και συνεχίζουν την εργασία τους σε όλο τον κύκλο ζωής του. Τα άτομα που πραγματοποιούν την επισκόπηση Ποιότητας, Χρήστη και Τεχνική επισκόπηση, προσλαμβάνονται κατά την διάρκεια του έργου ανάλογα με τις απαιτούμενες ικανότητες τους. Όλοι παίζουν σημαντικό ρόλο στην βεβαίωση της ποιότητας και ακρίβειας του προϊόντος και είναι ιδιαίτερα σημαντικοί.

1. Επισκοπήσεις Ποιότητας: Οι επισκοπήσεις της ποιότητας λαμβάνουν χώρα σε όλες τις φάσεις του κύκλου ζωής του έργου. Ο σκοπός τους είναι ο έλεγχος των εγγράφων παράδοσης πριν αυτή πραγματοποιηθεί.
2. Άτομο Τεχνικής Επισκόπησης: Η κύρια ευθύνη αυτού του ατόμου είναι η βεβαίωση ότι οι τεχνικές λεπτομέρειες των εγγράφων παράδοσης είναι σωστές και δυνατές. Ως αποτέλεσμα, το άτομο αυτό πρέπει να έχει τεχνικό υπόβαθρο και να είναι ικανό να χρησιμοποιήσει τις γνώσεις του για να αξιολογήσει το προϊόν από μία τεχνική οπτική γωνία.
3. Άτομο επισκόπησης χρήστη: Η κύρια ευθύνη αυτού του ατόμου είναι να ασχοληθεί με την διαδικασία Ανάπτυξης Προτιμήσεων. Τα άτομα αυτά πρέπει να έχουν καλές γνώσεις και να κατανοούν τις απαιτήσεις, ενώ πρέπει να μπορούν να τις επικοινωνούν αποδοτικά.

4. Προγραμματισμός και πραγματοποίηση της επισκόπησης: Κάτω από φυσιολογικές συνθήκες, οι επισκοπήσεις πραγματοποιούνται για κάθε ένα από τα προϊόντα της ομάδας του έργου. Η πιο ζωτική επισκόπηση είναι η τελευταία, η οποία παρέχει την έγκριση των εγγράφων. Μετά από την παράδοση του προϊόντος, τα άτομα πρέπει να αναγνωρίσουν τις όποιες πλευρές απαιτούν αλλαγή, επέκταση, διευκρίνιση, διαγραφή ή συζήτηση.

Η. Έβδομη πρόκληση: Η πραγματοποίηση αλλαγής και ο έλεγχος

Πολλά ιδρύματα ανώτατης εκπαίδευσης έχουν αναπτύξει ποικίλες εφαρμογές που χρησιμοποιούνται από την κοινότητα. Αυτές οι εφαρμογές περιέχουν πολλά δεδομένα, τα οποία περιλαμβάνουν κωδικούς πηγών, δεδομένα εισόδου κτλ. Τα δεδομένα αλλάζουν με την πάροδο του χρόνου είτε συστηματικά είτε αυτόματα. Μία καλή κατανόηση του τρόπου που μία εφαρμογή έχει αλλάξει είναι ο μόνος τρόπος αναδόμησης της. Πρέπει λοιπόν να επιλεχθεί μία κατάλληλη τεχνική και εργαλεία ώστε να ελεγχθούν αυτές οι αλλαγές.

6.3. Συμπεράσματα

Η ανάπτυξη επιχειρηματικών εφαρμογών είναι περίπλοκη, ιδιαίτερα για το περιβάλλον ανώτατης εκπαίδευσης, διότι έχει πολλές οπτικές γωνίες, στόχους και σκοπούς. Το προσωπικό πρέπει να έχει μία ξεκάθαρη εικόνα και να διαθέτει στρατηγικές για την ανάπτυξη μίας επιχειρηματικής εφαρμογής, διότι οι απαιτήσεις αυτού του περιβάλλοντος δεν μοιάζουν με αυτές των άλλων επιχειρήσεων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Pressman, R. S., (2000), *Software Engineering . A Practitioners Approach*, McGraw.Hill 4th edition.
- Σκορδαλάκης, Μ. (1991), *Εισαγωγή στην Τεχνολογία Λογισμικού*, Αθήνα, εκδόσεις ΣΥΜΜΕΤΡΙΑ
- Κόλλιας, Γ. (1986), *Βάσεις Δεδομένων*, τόμος ΙΙ, Αθήνα, εκδόσεις ΣΥΜΜΕΤΡΙΑ
- Pfleeger, Shari Lawrence, (2003), *Τεχνολογία Λογισμικού*, εκδόσεις ΚΛΕΙΔΑΡΙΘΜΟΣ, Αθήνα
- Sommerville, (2009), *Βασικές Αρχές Τεχνολογίας Λογισμικού,* εκδόσεις ΚΛΕΙΔΑΡΟΘΜΟΣ, Αθήνα
- Γιακουμάκης Μαν., Διαμαντίδης Νικ. (2009), *Τεχνολογία Λογισμικού,* εκδόσεις ΣΤΑΜΟΥΛΗ, Αθήνα
- *IEEE Guide To Software Requirements Specification*, ANSI/IEEE,Std 830.1993, 1984.

- *IEEE Recommended Practice for Software Design Descriptions*, ANSI/IEEE, Std 1016.1987.
- Barry W. Boehm. Software Engineering Economics. Advances in Computing Science and Technology. Prentice Hall, 1981.
- Mitch Lubars, Colin Potts, and Charles Richter. A review of the state of practice in requirements modelling. In Proceedings of the IEEE International Symposium on Requirements Engineering (RE93), pages 2–14, San Diego, California, USA, January 1993.
- Susan Gerhart, Dan Craigen, and Ted Ralston. Experience with formal methods in critical systems. IEEE Software, pages 21–39, January 1994.
- Victor R. Basili, Scott Green, Oliver Laitenberger, Filippo Lanubile, Forrest Shull, Sivert Sorumgard, and Marvin V. Zelkowitz. The empirical investigation of perspective-based reading. Journal of Empirical Software Engineering, 1(2):133– 164, 1996.
- Erik Kamsties and H. Dieter Rombach. A framework for evaluating system and software requirements specification approaches. In Manfred Broy and Bernhard Rumpe, editors, Proceedings of the RTSE'97 - Workshop on Requirements Targeting Software and Systems Engineering, pages 281–296, Munich, Germany, April 1998. Technical University of Munich (TUM), Technical Report TUM-I9807.
- Watts H. Humphrey. A Discipline for Software Engineering. Addison-Wesley, 1995.
- Ian Alexander, "The Basics of Requirements Management: Why Projects Have Requirements", White paper, January 2002. Ian Alexander's Home Page. Available: http://easyweb.easynet.co.uk/~iany/index.htm
- Ian Alexander, "Are There Requirements for BPS?" Proceedings of Workshop on Requirements Engineering for Business Process Support (REBPS'03) Austria, June 17, 2003

- Donaldson, Scott E and Siegel, Stanley G (2001), Successful Software Development, 2nd edi. NJ, USA: Prentice Hall.
- Ian Sommerville (1995). Software Engineering. 5th ed. United States of America: Addison Wesley.
- Thomas J. Crowe, Krishnakant Rathi, Joseph D. Rolfes, Applying a Taxonomy of Business Processes to Identify Reengineering Opportunities: Conference paper. Department of Industrial Engineering, University of Missouri, Columbia, USA
- IEEE Std 802-1981 (1981), Recommended Practice for Functional Requirement Document, Software Engineering Standards Committee of the IEEE Computer Society, New York.
- IEEE Std 830-1998 (1998), Recommended Practice for Software Requirements Specifications, Software Engineering Standards Committee of the IEEE Computer Society, New York.
- Lauesen, Soren (2002), Software Requirements: Style and Technique. Great Britain: Addison Wesley.
- Roger S. Pressman (2001). Software Engineering A Practitioner's Approach. 5th ed. United States of America: McGraw-Hill Companies.
- Peter Henderson, Yvonne Howard and Bob Walters, "A tool for evaluation of the Software Development Process"**,** *Journal of Systems and Software*, Vol 59, No 3, pp 355-362 (2001)
- Peter Henderson, **"**Business Processes, Legacy Systems and a Fully Flexible Future", appears in *Systems Engineering for Business Process Change*, Springer Verlag, 2000
- California Software, "Application Servers and Enterprise Application Development", White Paper. Available: http://www.californiasw.com/techcenter/javaapplicationdevelopment.html
- ClearNova, Inc. "Enterprise Application Development Challenges: An Overview for Managers" White Paper, Available: http://itsolutions.forbes.com/forbes/