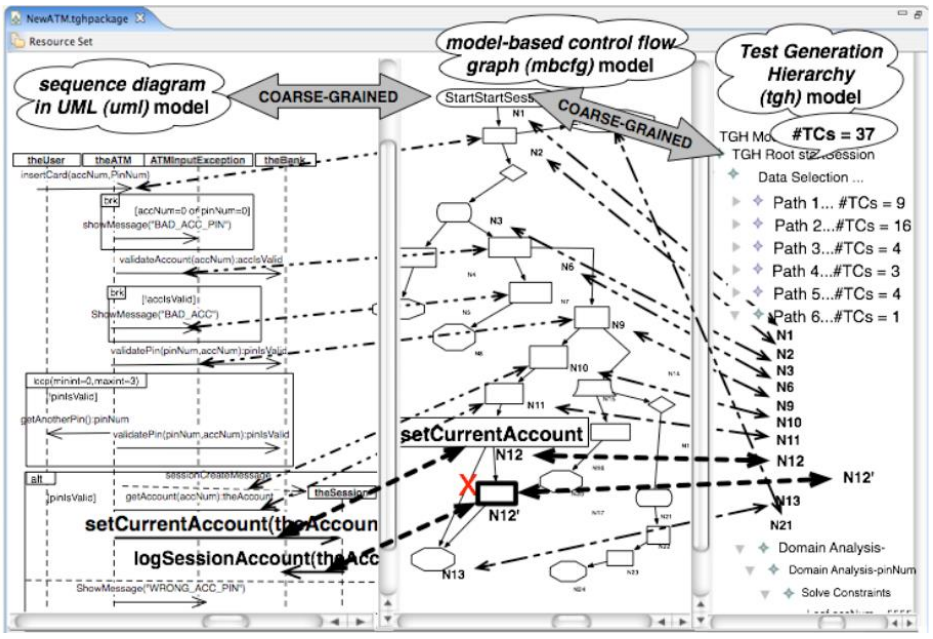
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία

Διερεύνηση της Ιχνηλασιμότητας από την σκοπιά των Τεχνικών Ελέγχου λογισμικού



Της φοιτήτριας **Επιβλέπων καθηγητής**

Κωνσταντινίδου Αγγελικής **Ιγνάτιος Δεληγιάννης**

Αρ. Μητρώου: 04/2617

Θεσσαλονίκη 2010

Πρόλογος

Τα επιχειρησιακά συστήματα λογισμικού είναι ιδιαίτερα σύνθετα και πρέπει να εκπληρώνουν με γρήγορους ρυθμούς τις μεταβαλλόμενες ανάγκες. Αυτές οι αλλαγές αντέχουν τους υψηλούς κινδύνους, όπως οι παρανοημένες αλληλεξαρτήσεις, ελλιπής κατανόηση, ελλιπής εφαρμογή και κάλυψη. Διαδικασίες εξελικτικής ανάπτυξης έχουν δημιουργηθεί για να υποστηρίξουν αυτές τις συχνές αλλαγές στο σύστημα. Μια από αυτές τις έννοιες είναι η ιχνηλασιμότητα. Τα περισσότερα πρότυπα αναπτυξιακής διαδικασίας υποστηρίζουν την ιχνηλασιμότητα και την αναφέρουν ως ένα από τα χαρακτηριστικά γνωρίσματά της [Maeder et al., 2007].

Σήμερα η αποτελεσματικότητα στις πρακτικές ιχνηλασιμότητας διαφέρει αρκετά μεταξύ των ομάδων ανάπτυξης. Μερικά προβλήματα, που εξηγούν αυτήν την κατάσταση, είναι [Ramesh, 1998, Ramesh and Jarke, 2001]: δεν υπάρχουν λεπτομερείς οδηγίες σχετικά με τα είδη από τις πληροφορίες που πρέπει να συγκεντρωθούν για την ιχνηλασιμότητα, το πλαίσιο στο οποίο τέτοιες πληροφορίες πρέπει να χρησιμοποιούνται, και η έλλειψη συνεννόησης για τη σημασία των συνδέσμων μεταξύ των προδιαγραφών.

Σχετικά με τη διαμόρφωση λογισμικού, η UML [OMG, 2002] έχει γίνει γρήγορα η δημοφιλέστερη γλώσσα για την αντικειμενοστραφή διαμόρφωση. Χάρη στον καθορισμό του μεταμοντέλου του και του συμπεριλαμβανόμενου μηχανισμού επέκτασης, η UML προσφέρει μια άριστη ευκαιρία για να θεσπιστεί κοινό πλαίσιο για την προδιαγραφή απαιτήσεων, ανάπτυξης και δοκιμής. Ακόμη, η ανάπτυξη με πρότυπα (model driven development - MDD) οδηγεί στην αυξανόμενη χρήση των προτύπων από κοινού με τον πηγαίο κώδικα στη δοκιμή λογισμικού. Η δοκιμή που βασίζεται στα πρότυπα, εντούτοις, εισάγει τις νέες προκλήσεις για τη δοκιμή των δραστηριοτήτων, οι οποίες περιλαμβάνουν τη δημιουργία και τη συντήρηση των πληροφοριών ιχνηλασιμότητας μεταξύ των τεχνουργημάτων που είναι για δοκιμή. Η ιχνηλασιμότητα απαιτείται για να υποστηρίξει τις δραστηριότητες όπως η εκλεκτική δοκιμή οπισθοδρόμησης. Στην πραγματικότητα, οι βασισμένες σε πρότυπα αυτοματοποιημένες εξεταστικές προσεγγίσεις επικεντρώνονται συχνά στην παραγωγή δοκιμών και σε δραστηριότητες εκτέλεσης, ενώ η υποστήριξη άλλων δραστηριοτήτων περιορίζεται (π.χ. η βασισμένη σε πρότυπα εκλεκτική δοκιμή οπισθοδρόμησης, η ανάλυση συμπεριφοράς και η αξιολόγηση του αποτελέσματος) [Naslavsky and Richardson, 2007].

Περίληψη

Σε αυτήν την εργασία γίνεται μια αναφορά σε μεταμοντέλα για την ιχνηλασιμότητα απαιτήσεων, η οποία είναι βασισμένη στη UML και ενσωματώνει κειμενικές προδιαγραφές σαν στοιχεία UML προτύπων, που διατηρούν μια ομοιογενή αντιπροσώπευση για όλη την ανάπτυξη τεχνουργημάτων λογισμικού και συνδέσμων ιχνηλασιμότητας. Η επιτυχία των διαδικασιών που παρουσιάζονται εξαρτάται από το πόσο καλά είναι καθορισμένες οι σχέσεις μεταξύ των απαιτήσεων και άλλων προδιαγραφών που παράγονται κατά τη διαδικασία λογισμικού. Η ιχνηλασιμότητα απαιτήσεων ορίζεται ως η δυνατότητα να περιγραφεί και να ακολουθηθεί η ζωή μια απαίτησης και στις δύο κατευθύνσεις, προς την προέλευσή της ή προς την εφαρμογή της, που περνά μέσω σχετικών προδιαγραφών. Ένα άλλο αντικείμενο της εργασίας είναι να διευκρινίσει με σαφήνεια στα πρόσθετα διαγράμματα τους συνδέσμους ιχνηλασιμότητας μεταξύ στοιχείων των προτύπων σε μερικά είδη προδιαγραφών. Η ιχνηλασιμότητα περιορίζεται συνήθως στο να συσχετίζεται με τις απαιτήσεις και δεν χρησιμοποιείται στην πράξη. Αλλά οι σύνδεσμοι ιχνηλασιμότητας είναι απαραίτητοι για ολόκληρη την αναπτυξιακή διαδικασία από τις απαιτήσεις έως την εφαρμογή του συστήματος. Η συντήρηση και η διαχείριση πρέπει να πραγματοποιηθεί χωρίς τη βοήθεια τεχνικών μέσων και απαιτεί πολύ μεγάλη προσπάθεια. Η προϋπόθεση για ένα αποτελεσματικό εργαλείο υποστήριξης είναι μια λεπτομερής ενσωμάτωση συνδέσμων ιχνηλασιμότητας στις μεθόδους ανάπτυξης. Προκειμένου να επιτευχθούν τα αποτελέσματα της υψηλής πρακτικής αξίας, μια ευρέως χρησιμοποιημένη μεθοδολογία σχεδίου εφαρμόζεται ως βάση. Έχει επιλεχτεί η ενοποιημένη διαδικασία (RUP) για να γίνει καθορισμός μιας συγκεκριμένης διαδικασίας προτύπου συνδέσμων ιχνηλασιμότητας σε αυτό το έγγραφο. Για την ενοποιημένη διαδικασία, υπάρχουν λεπτομερείς περιγραφές της μεθοδολογίας σχεδιασμού της. Οι επαγγελματίες θεωρούν τη δοκιμή λογισμικού κεντρικής σημασίας ώστε να εξασφαλιστεί ότι ένα σύστημα συμπεριφέρεται όπως αναμένεται. Λόγω της πρόσφατης διαδεδομένης υιοθέτησης της οδηγημένης με πρότυπα ανάπτυξης (model driven development -MDD), ο κώδικας δεν είναι πλέον η μοναδική πηγή για την επιλογή των περιπτώσεων δοκιμής. Περιγράφεται ένα παράδειγμα βασισμένο στη δοκιμή με πρότυπα που χρησιμοποιεί τεχνικές μετασχηματισμού μοντέλων για να δημιουργήσει μια υποδομή που περιλαμβάνει τα βασισμένα σε πρότυπα τεχνουργήματα προς δοκιμή και τις σαφής σχέσεις μεταξύ αυτών των προτύπων. Προτείνεται μια προσέγγιση που επηρεάζει τις τεχνικές του μετασχηματισμού προτύπων ιχνηλασιμότητας για να δημιουργηθούν προσεκτικά μελετημένες σχέσεις μεταξύ των δοκιμών των βασισμένων σε πρότυπο τεχνουργημάτων. Οι σχέσεις δημιουργούνται κατά τη διάρκεια της δοκιμής της διαδικασίας παραγωγής.

Abstract

In this work presented a reference to metamodels for requirements traceability that is based on UML and integrates as much textual specifications as UML model elements, obtaining a homogeneous representation for all the software development artifacts and traceability links among them. The success of processes depends on how well defined the relationships among

requirements and other kinds of specifications generated by the software process are. Requirements traceability is defined as the ability to describe and follow the life of a requirement in both directions, towards its origin or towards its implementation, passing through all the related specifications. Another part of this work is to specify explicitly in additional diagrams the traceability links between model elements, in some kinds of specifications. Traceability is mostly limited to relating requirements and poorly used in practice. But traceability links are necessary for the whole development process from requirements to the implementation of the system. The maintenance and management has currently to be carried out manually and requires an extreme high effort. The precondition for an effective tool support is a detailed integration of traceability links into development methods. In order to achieve results of high practical value, a widely used design methodology is applied as base. The Rational Unified Process (RUP) is chosen for the definition of a process-specific model of traceability links. For the Unified Process, there are rather detailed descriptions of the design methodology. Practitioners regard software testing as the central means for ensuring that a system behaves as expected. Due to the recent widespread adoption of model-driven development (MDD), code is no longer the single source for selecting test cases. Is described an example based on model testing that uses model transformation techniques to create an infrastructure that comprises model-based testing artifacts and fine-grained relationships among these models. This work proposes an approach that leverages model transformation traceability techniques to create fine-grained relationships among model-based testing artifacts. Relationships are created during the test generation process.

Περιεχόμενα

Εισαγωγή

Ο στόχος αυτής της εργασίας είναι να παρουσιαστεί ένα πλαίσιο για τη διερεύνηση της ιχνηλασιμότητας από τη σκοπιά των τεχνικών ελέγχου λογισμικού με την βοήθεια κειμενικών προδιαγραφών και στοιχείων UML προτύπων. Η προσέγγισή μπορεί να εφαρμοστεί σε οποιαδήποτε διαδικασία λογισμικού με βάση τη UML αφού κυρίως βασίζεται στα τεχνουργήματα που παράγονται μέσω αυτής της διαδικασίας και μπορεί να προσαρμοστεί σύμφωνα με τις συγκεκριμένες ανάγκες ιχνηλασιμότητας του προγράμματος.

Αυτό το έγγραφο οργανώνεται σε επτά τμήματα. Μετά από αυτή η εισαγωγή, το πρώτο τμήμα είναι μια εισαγωγή στην ιχνηλασιμότητα απαιτήσεων. Παραθέτονται ορισμοί για την ιχνηλασιμότητα απαιτήσεων, αναφέρονται κάποιοι λόγοι για την ιχνηλασιμότητα απαιτήσεων στις διαφορετικές φάσεις της ανάπτυξης συστημάτων, και επίσης περιγράφονται κάποιοι απαραίτητοι όροι για την ιχνηλασιμότητα. Ακόμη περιγράφεται ένα εννοιολογικό πρότυπο ιχνών και δύο πρότυπα αναφοράς ιχνηλασιμότητας. Το δεύτερο τμήμα αναφέρεται σε τρόπους εφαρμογής της ιχνηλασιμότητας και περιγράφει συγκεκριμένα εργαλεία για τη διαχείριση

απαιτήσεων, από την προοπτική των πλαισίων για τις απαιτήσεις της ιχνηλασιμότητας. Το τρίτο τμήμα παρουσιάζει ένα μεταμοντέλο για την ιχνηλασιμότητα απαιτήσεων και εξηγεί πώς οι κειμενικές προδιαγραφές και οι σύνδεσμοι ιχνηλασιμότητας μπορούν να καθοριστούν στο UML πλαίσιο, διατηρώντας ένα κοινό πλαίσιο για όλες τις πληροφορίες ιχνηλασιμότητας και επεξηγείται αυτή η προσέγγισή χρησιμοποιώντας ένα μικρό πρόγραμμα βασισμένο σε RUP ως παράδειγμα. Στο τέταρτο τμήμα αναλύεται η ενοποιημένη διαδικασία (UP) τεχνουργημάτων ώστε να καθιερωθεί ένα πρότυπο συνδέσμων ιχνηλασιμότητας μεταξύ των τεχνουργημάτων. Επίσης καθορίζεται ένα πρώτο σύνολο κανόνων για επαλήθευση των συνδέσμων στις ενοποιημένες μελέτες αναπτυξιακής διαδικασίας. Στο πέμπτο τμήμα παρουσιάζει τη βασισμένη σε πρότυπα δοκιμή οπισθοδρόμησης με σαφής σχέσεις που καθιερώνονται κατά τη διάρκεια της διαδικασίας παραγωγής δοκιμής. Αρχικά παρουσιάζεται η επισκόπηση της προσέγγισής και μετά αναλυτικά ένα παράδειγμα όπου μετασχηματίζεται το πρότυπο για να δημιουργήσει μια ιχνηλάσιμη υποδομή των σχετικών τεχνουργημάτων που είναι προς δοκιμή. Το έκτο τμήμα περιγράφει μερικές σχετικές εργασίες για τη διαχείριση της ιχνηλασιμότητας απαιτήσεων. Τέλος, το έβδομο τμήμα παρουσιάζει τα συμπεράσματα.

1. Ιχνηλασιμότητα Απαιτήσεων

Η ιχνηλασιμότητα ως γενικός όρος είναι η « ικανότητα να συνδέονται χρονολογικά οι μοναδικά αναγνωρίσιμες οντότητες με τρόπο που έχει κάποια σημασία». Η λέξη χρονολογικά, εδώ, απεικονίζει τη χρήση του όρου όσο αφορά την διαδρομή κάποιου τροφίμου από το χωράφι στο κατάστημα, ή κάποιου φαρμάκου από το εργαστήριο στο ανθρώπινο στόμα. Αυτό που μετράει στην διαχείριση απαιτήσεων δεν είναι τόσο η χρονική εξέλιξη όσο η κατασκευαστική εξέλιξη: ένα ίχνος για το από πού προέρχονται οι απαιτήσεις, πως ικανοποιούνται, πως δοκιμάζονται, και τι επίδραση θα υπάρξει αν αλλάξουν [Wikipedia, 2009].

Η ιχνηλασιμότητα μας επιτρέπει να βρούμε με τρόπο απλούστερο πληροφορίες γύρω από το πρόγραμμα μας. Παραδείγματος χάριν μια υψηλού επιπέδου απαίτηση μπορεί να διασπαστεί σε διάφορες λεπτομερείς απαιτήσεις. Παραδοσιακά η υψηλού επιπέδου απαίτηση θα ήταν σε ένα διαφορετικό έγγραφο από τις λεπτομερείς απαιτήσεις. Για να κρατήσετε τη σχέση ή την ιχνηλασιμότητα μεταξύ αυτών των απαιτήσεων πρέπει να προσθέσετε ένα σύνδεσμο ή ίχνος μεταξύ τους. Στην προδιαγραφή λεπτομερούς απαίτησης θα μπορούσατε να προσθέσετε μια στήλη που παρουσιάζει μια υψηλού επιπέδου απαίτηση που συνδέθηκε με κάθε μια από τις λεπτομερείς απαιτήσεις.

Το τυποποιημένο γλωσσάριο IEEE της ορολογίας τεχνολογίας λογισμικού καθορίζει την ιχνηλασιμότητα ως «βαθμό στον οποίο μια σχέση μπορεί να καθιερωθεί μεταξύ δύο ή

περισσότερων προϊόντων της αναπτυξιακής διαδικασίας, ειδικά στα προϊόντα που έχουν μια σχέση προκάτοχος-διάδοχος ή προϊστάμενος -υφιστάμενος.» [IEEE, 1999].

Η ιχνηλασιμότητα είναι μια από τις ουσιαστικές δραστηριότητες της καλής διαχείρισης απαιτήσεων. Χρησιμοποιείται για να εξασφαλίσει ότι χτίζονται τα σωστά προϊόντα σε κάθε φάση του κύκλου ζωής ανάπτυξης λογισμικού, για να επισημάνουν την πρόοδο της ανάπτυξης και για να μειώσουν την προσπάθεια που απαιτείται να καθοριστούν οι επιδράσεις των ζητούμενων αλλαγών.

Η ιχνηλασιμότητα χρησιμοποιείται για να ακολουθήσει τη σχέση μεταξύ κάθε μοναδικού προϊόντος-επιπέδου απαίτηση και της πηγής της. Παραδείγματος χάριν, μια απαίτηση προϊόντος μπορεί να ανιχνευτεί από μια επιχειρησιακή ανάγκη, ένα αίτημα των χρηστών, έναν επιχειρησιακό κανόνα, μια εξωτερική προδιαγραφή, μια βιομηχανική τυποποίηση ή κανονισμό, έως και από κάποια άλλη πηγή. Χρησιμοποιείται επίσης για να ακολουθήσει τη σχέση μεταξύ κάθε μοναδικού προϊόντος-επιπέδου απαίτηση και των προϊόντων εργασίας στις οποίες εκείνη η απαίτηση διατίθεται. Παραδείγματος χάριν, μια απαίτηση προϊόντος μπορεί να ανιχνευτεί σε ένα ή περισσότερα αρχιτεκτονικά στοιχεία, στοιχεία σχεδίου λεπτομέρειας, αντικείμενα/κλάσεις, μονάδες κώδικα, δοκιμές, θέματα τεκμηρίωσης χρηστών, ή/και ακόμη και σε ανθρώπους ή χειρωνακτικές διαδικασίες που χρησιμοποιούν αυτήν την απαίτηση.

Οι απαιτήσεις προέρχονται από διάφορες πηγές, όπως του ατόμου που παράγει ένα προϊόν, του διευθυντή πωλήσεων και του τελικού χρήστη. Αυτοί οι άνθρωποι έχουν διαφορετικές απαιτήσεις από το προϊόν. Χρησιμοποιώντας την ιχνηλασιμότητα των απαιτήσεων ένα εφαρμοσμένο γνώρισμα μπορεί να οδηγηθεί πίσω στο άτομο ή στην ομάδα που το ήθελε (το ζητούσε) κατά την διάρκεια της εκμαίευσης των απαιτήσεων. Αυτό μπορεί για παράδειγμα να χρησιμοποιηθεί κατά την διάρκεια μιας διαδικασίας ανάπτυξης για να δώσει προτεραιότητα σε κάποια απαίτηση, λαμβάνοντας υπόψη πόσο πολύτιμη είναι η απαίτηση στον κάθε εμπλεκόμενο. Ακόμη μπορεί να χρησιμοποιηθεί μετά την ανάπτυξη, όταν ο χρήστης παρατηρήσει ότι κάποιο γνώρισμα δεν χρησιμοποιείται, για να δει γιατί είχε αρχικά απαιτηθεί [Wikipedia, 2009].

Το μόνο πράγμα που είναι σταθερό σε ένα πρόγραμμα είναι η αλλαγή. Έτσι όταν μια υψηλού επιπέδου απαίτηση αλλάζει θα είναι απαραίτητο να βρεθούν όλες οι παραγόμενες απαιτήσεις και να βεβαιωθεί ότι ισχύουν ακόμα. Εάν θέλετε να βρείτε όλες τις απαιτήσεις προερχόμενες από μια συγκεκριμένη υψηλού επιπέδου απαίτηση πρέπει να ψάξετε μέσω κάθε παραγόμενης προδιαγραφής απαιτήσεων για εκείνες τις λεπτομερείς απαιτήσεις με τη σχετική υψηλού επιπέδου απαίτηση. Εάν είχατε μόνο μια προδιαγραφή λεπτομερών απαιτήσεων αυτό θα ήταν εντάξει, αλλά τι συμβαίνει όταν έχετε δέκα ή περισσότερες παραγόμενες προδιαγραφές;

Η ιχνηλασιμότητα απαιτήσεων παρέχει κρίσιμη υποστήριξη για τους μηχανικούς λογισμικού δεδομένου ότι αυτοί αναπτύσσουν και διατηρούν τα συστήματα λογισμικού. Η ιχνηλασιμότητα μπορεί, παραδείγματος χάριν, να βοηθήσει έναν αναλυτή, να εξασφαλίσει ότι μια υψηλού επιπέδου απαίτηση έχει εκλεπτυνθεί σε χαμηλότερων τμημάτων σχεδιασμού συστατικά, έχει χτιστεί στο εκτελέσιμο σύστημα, και έχει εξεταστεί αποτελεσματικά. Η ιχνηλασιμότητα βοηθά επίσης τους αναλυτές για να καταλάβουν αποτελεσματικά, να διαχειριστούν, και να ελέγξουν τις αλλαγές καθώς εισάγονται. Πολυάριθμα πρότυπα όπως IEEE πρότυπα 830 - 1998, CMMI, και ISO 9001 εξουσιοδοτούν πρακτικές ιχνηλασιμότητας ως απαραίτητο στοιχείο του κύκλου ζωής ανάπτυξης λογισμικού. Δυστυχώς, η άσκηση της ιχνηλασιμότητας είναι πολύ πιο δύσκολη από ότι φαίνεται και πολλές οργανώσεις αποτυγχάνουν να εφαρμόσουν αποτελεσματικές πρακτικές ιχνηλασιμότητας, και έτσι νομίζουν ότι είναι δαπανηρές και ανέφικτες.

Οι απαιτήσεις λογισμικού είναι ευαίσθητες σε αλλαγές, όχι μόνο όταν θα είναι έτοιμο το προϊόν και μετά αλλά και κατά την διάρκεια της επαναληπτικής διαδικασίας ανάπτυξης λογισμικού. Η διαχείριση απαιτήσεων είναι η υπεύθυνη διαδικασία για την εποπτεία των αλλαγών στις απαιτήσεις λογισμικού, και πρέπει ενσωματώνεται ως υποδιαδικασία στη διαδικασία ανάπτυξης λογισμικού, που είναι ο πυρήνας της διαδικασίας λογισμικού [Corriveau, 1996].

Η διαχείριση απαιτήσεων και ειδικά η ιχνηλασιμότητα απαιτήσεων μπορεί να είναι ακριβές δραστηριότητες. Το επίπεδο λεπτομέρειας σε αυτές τις δραστηριότητες και τις συλλεχθείσες πληροφορίες πρέπει να διαμορφώνεται σύμφωνα με τις ιδιαίτερες ανάγκες του προγράμματος, προκειμένου να ληφθεί μια θετική αναλογία κόστους-κέρδους.

Η ιχνηλασιμότητα των απαιτήσεων είναι υπό-τομέας της διαχείρισης απαιτήσεων με την βοήθεια της ανάπτυξης λογισμικού και των μηχανικών συστημάτων. Συσχετίζεται με την καταγραφή της ζωής μιας απαίτησης. Δίνει την ικανότητα να οδηγηθεί πίσω στην προέλευση κάθε απαίτηση και πρέπει συνεπώς κάθε αλλαγή που γίνεται στις απαιτήσεις να καταγράφεται ώστε να επιτευχθεί η ιχνηλασιμότητα. Ακόμη και η χρήση της απαίτησης, μετά την εκπλήρωση του γνωρίσματος που έχει αναπτυχθεί και χρησιμοποιηθεί, πρέπει να είναι ανιχνεύσιμη[Wikipedia, 2009].

Η ιχνηλασιμότητα των απαιτήσεων αφορά την καταγραφή των σχέσεων μεταξύ των απαιτήσεων και άλλων τεχνουργημάτων ανάπτυξης. Ο σκοπός της είναι να διευκολύνει:

- την κατανόηση του προϊόντος υπό ανάπτυξή και των τεχνουργημάτων του και
- την ικανότητα να διαχειρίζεται τις αλλαγές

Πρέπει να εντοπίζονται όχι μόνο οι απαιτήσεις αλλά και οι σχέσεις των απαιτήσεων με όλα τα συσχετιζόμενα τεχνουργήματα, όπως μοντέλα, αποτελέσματα αναλύσεων, περιπτώσεις ελέγχων, διαδικασίες ελέγχων, αποτελέσματα ελέγχων και τεκμηριώσεις όλων των ειδών. Ακόμη και οι άνθρωποι και οι ομάδες χρηστών που συσχετίζονται με τις απαιτήσεις πρέπει να είναι ανιχνεύσιμοι [Wikipedia, 2009].

Αναφορικά στον τρόπο έκφρασης τους, οι απαιτήσεις ήταν παραδοσιακά διευκρινισμένες χρησιμοποιώντας προ πάντων κειμενικές μορφές προδιαγραφής, κυρίως φυσικής γλώσσας. Συνεπώς, η υποστήριξη εργαλείων στη διαχείριση απαιτήσεων έχει στραφεί στον χειρισμό των κειμένων. Αυτές οι απαιτήσεις που εκφράζονται κειμενικά συνδέονται διαμορφώνοντας μια γραφική παράσταση ιχνηλασιμότητας η οποία χρησιμοποιείται για να διαχειριστεί τις απαιτήσεις και την ιχνηλασιμότητα. Σε αυτή τη προσέγγιση, οι προδιαγραφές που παράγονται σε άλλες δραστηριότητες της αναπτυξιακής διαδικασίας μπορεί επίσης να προστεθούν στη γραφική παράσταση ιχνηλασιμότητας, αντιπροσωπεύοντας ένα κείμενο (συνήθως χρησιμοποιείται το όνομα της προδιαγραφής, για παράδειγμα: το όνομα της κλάσης, της ιδιότητας ή της λειτουργίας). Οι προδιαγραφές δοκιμής είναι επίσης κυρίως κειμενικοί, κατά συνέπεια μπορούν να αντιμετωπιστούν με έναν παρόμοιο τρόπο. Ακόμα κι αν οι διάφοροι προμηθευτές εργαλείων τεχνολογίας λογισμικού με τη βοήθεια υπολογιστή (CASE tools) [Pfleeger, 2003] απαιτούν ότι τα προϊόντα τους προσφέρουν μια ολοκληρωμένη πρόταση μεταξύ εφαρμογών για τη διαχείριση απαιτήσεων, τη μοντελοποίηση και τη δοκιμή, η λύση είναι συνήθως βασισμένη σε μηχανισμούς εισαγωγής / εξαγωγής μεταξύ τέτοιων εφαρμογών. Αυτή η στρατηγική δεν είναι η καλύτερη που διαδίδεται δεδομένου ότι αφορά μόνο ένα μέρος της διαδικασίας λογισμικού. Μια άλλη προτεινόμενη εναλλακτική λύση είναι να διευκρινίσει ρητά στα πρόσθετα διαγράμματα τα σημάδια ιχνηλασιμότητας (σύνδεσμοι) μεταξύ στοιχείων των προτύπων, αλλά όχι σε όλα τα είδη

προδιαγραφών, γιατί λόγω της πολυπλοκότητας της γραφικής παράστασης ιχνηλασιμότητας, κάτι τέτοιο δεν θα είναι βιώσιμο, ακόμη και για τα μικρά συστήματα.

1.1 Ορισμοί για την ιχνηλασιμότητα απαιτήσεων

«Η ιχνηλασιμότητα απαιτήσεων μπορεί να οριστεί ως η δυνατότητα να περιγραφεί και να ακολουθηθεί η ζωή μιας απαίτησης, με κατεύθυνση και προς τα εμπρός και προς τα πίσω (δηλ. από την προέλευσή της, μέσω της ανάπτυξης και της εξειδίκευσης της, στην επέκταση, υλοποίηση και τη χρήση της, και μέσω όλων των περιόδων εκλέπτυνσης και επανάληψης σε οποιεσδήποτε από αυτές τις φάσεις)» [Gotel and Finklestein, 1994].

Ενώ αυτός ο ορισμός υπογραμμίζει την παρακολούθηση της ζωής μιας απαίτησης μέσω όλων των φάσεων ανάπτυξης, δεν είναι σαφώς προσδιορισμένο ότι η ιχνηλασιμότητα μπορεί να τεκμηριώσει τις σχέσεις μεταξύ πολλών ειδών τεχνουργημάτων ανάπτυξης, όπως οι απαιτήσεις, οι δηλώσεις προδιαγραφών, τα σχέδια, οι δοκιμές, τα πρότυπα και τα αναπτυσσόμενα συστατικά. Ο επόμενος ορισμός αντιμετωπίζει αυτό το ζήτημα:

«Η ιχνηλασιμότητα απαιτήσεων αναφέρεται στη δυνατότητα να καθοριστούν, να συλληφθούν και να ακολουθηθούν τα ίχνη που αφήνονται από τις απαιτήσεις σε άλλα στοιχεία του περιβάλλοντος ανάπτυξης λογισμικού και του ίχνους που αφήνεται από αυτά τα στοιχεία στις απαιτήσεις» [Pinheiro and Goguen, 1996].

Ο ακόλουθος ορισμός τονίζει τη χρήση της ιχνηλασιμότητας για να τεκμηριώσει το μετασχηματισμό μιας απαίτησης σε διαδοχικά συγκεκριμένα τεχνουργήματα σχεδίου και ανάπτυξης:

«Στον τομέα της μηχανικής απαιτήσεων, η ιχνηλασιμότητα είναι για την κατανόηση του πώς υψηλού επιπέδου απαιτήσεις - στόχοι, σκοποί, φιλοδοξίες, προσδοκίες, ανάγκες - μετασχηματίζονται σε χαμηλού επιπέδου απαιτήσεις. Ενδιαφέρεται επομένως πρώτιστα για τις σχέσεις μεταξύ των στρωμάτων των πληροφοριών» [Hull et al., 2005].

1.2 Τεχνικές της ιχνηλασιμότητας απαιτήσεων

Η ιχνηλασιμότητα απαιτήσεων μπορεί να επιτευχθεί με τη χρησιμοποίηση μιας ή περισσότερων από τις ακόλουθες τεχνικές:

- Παραπομπές. Αυτή η τεχνική περιλαμβάνει την ενσωμάτωση φράσεων όπως «βλέπε στην παράγραφο Χ» σε όλη την τεκμηρίωση προγράμματος (π.χ. επεξηγήσεις, αρίθμηση ή κατηγοριοποίηση των απαιτήσεων, και εξειδικευμένοι πινάκες ή μήτρες που συνοδεύουν τις παραπομπές).
- Ειδικευμένα πρότυπα και έγγραφα ολοκλήρωσης ή μετασχηματισμού. Αυτά χρησιμοποιούνται για να αποθηκεύσουν τους συνδέσμους μεταξύ των εγγράφων που δημιουργούνται στις διαφορετικές φάσεις ανάπτυξης.
- Αναδόμηση. Η τεκμηρίωση αναδομείται από την βοήθεια ενός υπονοούμενου δικτύου ή μιας γραφικής παράστασης για να παρακολουθήσει τις αλλαγές των απαιτήσεων (π.χ., δίκτυα

συντήρησης βασισμένα σε υποθέσεις αλήθειας, αλυσιδωτοί μηχανισμοί, δίκτυα περιορισμού και αναπαραγωγής).

1.3 Λόγοι για την ιχνηλασιμότητα απαιτήσεων

Οι ανάγκες ιχνηλασιμότητας των διαφορετικών συμμετεχόντων - χορηγοί προγράμματος, διευθυντές προγράμματος, αναλυτές, σχεδιαστές, συντηρητές, και τελικοί χρήστες - διαφέρουν λόγω διαφορετικών στόχων και προτεραιοτήτων. Η ιχνηλασιμότητα απαιτήσεων είναι χαρακτηριστικό ενός συστήματος στο οποίο οι απαιτήσεις συνδέονται σαφώς με τις πηγές τους και με τα τεχνουργήματα που δημιουργούνται κατά τη διάρκεια του κύκλου ζωής ανάπτυξης συστημάτων βασιζόμενο σε αυτές τις απαιτήσεις [Ramesh and Jarke, 2001].

Στην εφαρμοσμένη μηχανική απαιτήσεων και στη φάση εκμαίευσης είναι σημαντικό ότι οι λογικές επεξηγήσεις και οι πηγές των απαιτήσεων λαμβάνονται υπόψη προκειμένου να κατανοήσουν την εξέλιξη και επαλήθευση των απαιτήσεων [Ramesh and Jarke, 2001].

Εμφανίζονται τροποποιήσεις κατά τη διάρκεια του σχεδιασμού π.χ. εάν οι απαιτήσεις εξελίσσονται ή εάν το σύστημα είναι αναπτυγμένο επαυξητικά [Knethen, 2002]. Κατά τη διάρκεια της φάσης σχεδίασης των απαιτήσεων η ιχνηλασιμότητα επιτρέπει να παρακολουθήσουμε τι συμβαίνει όταν το αίτημα αλλαγής εφαρμόζεται, πριν ξανασχεδιαστεί το σύστημα. Η ιχνηλασιμότητα μπορεί επίσης να δώσει τις πληροφορίες για τα αποτελέσματα, για σημαντικές αποφάσεις και υποθέσεις των απαιτήσεων [Ramesh and Jarke, 2001].

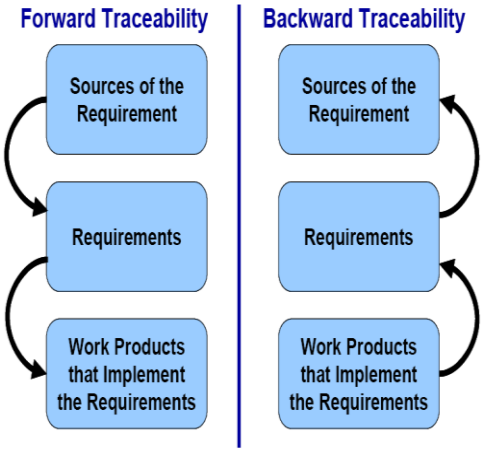
Δοκιμαστικές διαδικασίες μπορούν να ανιχνευτούν στις απαιτήσεις ή στο σχέδιο και αυτό το είδος ιχνηλασιμότητας βοηθά για να σχεδιάσουμε και να τροποποιήσουμε τις διαδικασίες δοκιμής [Ramesh and Jarke, 2001].

Για διάφορους λόγους συμβαίνουν τροποποιήσεις μετά από την παράδοση του συστήματος (π.χ. διόρθωση ελαττωμάτων ή για να προσαρμοστεί το σύστημα σε ένα μεταβαλλόμενο περιβάλλον). Αυτά τα είδη τροποποιήσεων ονομάζονται ανάπτυξη συστημάτων [Lehman and Ramil, 2003]. Οι εμπειρικές μελέτες δείχνουν ότι ακόμη και οι πεπειραμένοι επαγγελματίες λογισμικού προβλέπουν ελλιπή σύνολα επιδράσεων των αλλαγών [Lindvall and Sandahl, 1998].

Οι απαιτήσεις μπορούν να καθοριστούν καλύτερα με πλήρη ιχνηλασιμότητα, ακριβέστερες δαπάνες και προγράμματα αλλαγών, παρά με τον αρμόδιο μηχανικό ή προγραμματιστή που μπορεί να μην ξέρει όλες τις περιοχές που επηρεάζονται από αυτές τις αλλαγές [Ramesh and Jarke, 2001].

1.4 Αμφίδρομη ιχνηλασιμότητα

Οι ορθές πρακτικές ιχνηλασιμότητας επιτρέπουν την αμφίδρομη ιχνηλασιμότητα, που σημαίνει ότι οι αλυσίδες ιχνηλασιμότητας μπορούν να επισημανθούν και στην προς τα εμπρός και στην προς τα πίσω κατεύθυνση όπως διευκρινίζονται στο σχήμα 1.1.



Σχήμα 1.1: Αμφίδρομη (προς τα εμπρός & προς τα πίσω) ιχνηλασιμότητα.

Η προς τα εμπρός ιχνηλασιμότητα εξετάζει:

• Αντιγραφή των πηγαίων απαιτήσεων στο προκύπτον προϊόν ώστε η απαίτηση να εξασφαλίσει πληρότητα σύμφωνη με την προδιαγραφή απαίτησης του προϊόντος .

• Αντιγραφή κάθε μοναδικού προϊόντος απαίτησης προς τα εμπρός, μέσα στο σχέδιο που εφαρμόζει την απαίτηση, στον κώδικα που εφαρμόζει το σχέδιο και στις δοκιμές που επικυρώνουν αυτή την απαίτηση και ούτω καθεξής. Ο στόχος είναι να εξασφαλιστεί ότι κάθε απαίτηση εφαρμόζεται στο προϊόν και ότι κάθε απαίτηση εξετάζεται λεπτομερώς.

Η προς τα πίσω η ιχνηλασιμότητα εξετάζει:

• Αντιγραφή κάθε μοναδικού προϊόντος εργασίας (π.χ., στοιχείο σχεδίου, αντικείμενο/κλάση, μονάδα κώδικα, δοκιμή) πίσω στη σχετική απαίτησή του. Η προς τα πίσω ιχνηλασιμότητα μπορεί να ελέγξει ότι οι απαιτήσεις συσχετίζονται άμεσα με το σχέδιο, τον κώδικα, και τις δοκιμές.

• Αντιγραφή κάθε απαίτησης - πίσω - στην πηγή της.

Το Software Engineering Institute (SEI) Capability Maturity Model Integration (CMMI®) δηλώνει ότι ο σκοπός της διαδικασίας στη «διαχείριση απαιτήσεων είναι να ρυθμιστούν οι απαιτήσεις του προϊόντος του προγράμματος και τα τμήματα προϊόντων και να προσδιοριστούν οι ασυνέπειες μεταξύ εκείνων των απαιτήσεων και των σχεδίων του προγράμματος και των προϊόντων εργασίας» [SEI, 2000]. Μια από τις συγκεκριμένες πρακτικές της διαδικασίας απαιτήσεων είναι «να διατηρηθεί η αμφίδρομη ιχνηλασιμότητα των απαιτήσεων » [SEI, 2000]. Ποιο είναι το όφελος της προσπάθειας να διατηρηθεί η αμφίδρομη ιχνηλασιμότητα;

Η προς τα εμπρός ιχνηλασιμότητα εξασφαλίζει τη σωστή κατεύθυνση του εξελισσόμενου προϊόντος (ότι χτίζουμε το κατάλληλο προϊόν) και δείχνει την πληρότητα της επόμενης εφαρμογής. Παραδείγματος χάριν, εάν ένας επιχειρησιακός κανόνας δεν μπορεί να επισημανθεί

προς τα εμπρός σε μια ή περισσότερες απαιτήσεις προϊόντων τότε η προδιαγραφή απαιτήσεων προϊόντων είναι ελλιπής και το προκύπτον προϊόν μπορεί να μην ικανοποιήσει τις ανάγκες της επιχείρησης. Εάν μια απαίτηση προϊόντων δεν μπορεί να αντιγραφεί προς τα εμπρός στα σχετικά αρχιτεκτονικά στοιχεία σχεδίου της, κατόπιν το αρχιτεκτονικό σχέδιο δεν είναι πλήρες και ούτω καθεξής.

Εάν, από την άλλη, υπάρχουν αλλαγές στο επιχειρησιακό περιβάλλον (π.χ., μια αλλαγή επιχειρησιακού κανόνα ή μια τυποποιημένη αλλαγή), τότε εάν η προς τα εμπρός ιχνηλασιμότητα έχει διατηρηθεί, εκείνη η αλλαγή μπορεί να επισημανθεί προς τα εμπρός στις σχετικές απαιτήσεις και σε όλα τα προϊόντα εργασίας που επηρεάζονται από εκείνη την αλλαγή. Αυτό μειώνει πολύ την προσπάθεια που απαιτείται για να γίνει μια λεπτομερή εργασία της ανάλυσης. Μειώνει επίσης τον κίνδυνο ότι ένα από τα προϊόντα εργασίας έχει ξεχαστεί, με συνέπεια μια ελλιπή εφαρμογή της αλλαγής (δηλαδή μια ατέλεια).

Η προς τα πίσω ιχνηλασιμότητα βοηθά να εξασφαλιστεί ότι το εξελισσόμενο προϊόν παραμένει στη σωστή διαδρομή σε σχέση με τις αρχικές ή/και εξελισσόμενες απαιτήσεις (ότι χτίζουμε το προϊόν σωστά). Ο σκοπός είναι να εξασφαλιστεί ότι δεν επεκτείνουμε το στόχο του προγράμματος με την προσθήκη στοιχείων σχεδίου, κώδικα, δοκιμών ή άλλων προϊόντων εργασίας που δεν απαιτούνται στις απαιτήσεις (όπως λέγεται «χρυσή επένδυση»). Εάν απαιτείται να γίνει μια αλλαγή στην εφαρμογή ή εάν οι υπεύθυνοι για την ανάπτυξη έχουν βρει μια πιο δημιουργική, νέα τεχνική λύση, τότε η αλλαγή ή η λύση πρέπει να επισημανθεί προς τα πίσω στις απαιτήσεις και η επιχείρηση πρέπει να εξασφαλίσει ότι είναι στα πλαίσια του επιθυμητού προϊόντος. Παραδείγματος χάριν, εάν υπάρχει ένα στοιχείο προϊόντων εργασίας που δεν ανιχνεύεται στις προς τα πίσω απαιτήσεις του προϊόντος, ένα, εκ των δύο πραγμάτων, είναι αληθινό. Η πρώτη πιθανότητα είναι ότι υπάρχει μια ελλιπής απαίτηση επειδή το στοιχείο του προϊόντος της εργασίας απαιτείται πραγματικά. Σε αυτήν την περίπτωση, η ιχνηλασιμότητα έχει βοηθήσει να προσδιορίσει την ελλείπουσα απαίτηση και μπορεί επίσης να χρησιμοποιηθεί για να αξιολογήσει τις επιδράσεις της προσθήκης εκείνης της απαίτησης στα σχέδια προγράμματος και σε άλλα προϊόντα εργασίας (μπροστινή ιχνηλασιμότητα). Η δεύτερη πιθανότητα είναι ότι υπάρχει η κατάσταση της «χρυσής επένδυσης» - κάτι έχει προστεθεί που δεν πρέπει να είναι μέρος του προϊόντος. Η χρυσή επένδυση είναι μια δραστηριότητα υψηλού κινδύνου επειδή τα σχέδια προγραμματισμού δεν έχουν διαθέσει το χρόνο ή πόρους στην εργασία και η ύπαρξη αυτού του μέρους του προϊόντος δεν μπορεί να κοινοποιηθεί σε άλλο προσωπικό προγράμματος (π.χ., ο ελεγκτής δεν τον εξετάζει, δεν συμπεριλαμβάνεται στην τεκμηρίωση χρηστών).

Ένα άλλο όφελος της οπίσθιας ιχνηλασιμότητας είναι όταν προσδιορίζεται μια ατέλεια σε ένα από τα προϊόντα εργασίας. Παραδείγματος χάριν, εάν ένα κομμάτι του κώδικα έχει μια ατέλεια, η μήτρα ιχνηλασιμότητας μπορεί να χρησιμοποιηθεί για να βοηθήσει να καθοριστεί η πρωταρχική αιτία της ατέλειας. Παραδείγματος χάριν, είναι αυτό απλά μια ατέλεια του κώδικα ή είναι μια ατέλεια του σχεδιασμού ή των απαιτήσεων; Εάν είναι μια ατέλεια του σχεδιασμού ή των απαιτήσεων, ποια άλλα προϊόντα της εργασίας επηρεάζει αυτή η ατέλεια;

Τα οφέλη της αμφίδρομης ιχνηλασιμότητας απαιτήσεων περιλαμβάνουν τις παρακάτω δυνατότητες:

• Να αναλύει τον αντίκτυπο μιας αλλαγής

− Όλα τα προϊόντα εργασίας που επηρεάζονται από μια αλλαγμένη απαίτηση

− Όλες οι απαιτήσεις που επηρεάζονται από μια αλλαγή ή μια ατέλεια σε ένα προϊόν εργασίας

• Αξιολόγηση της παρούσας κατάστασης των απαιτήσεων και του προγράμματος
− Εύρεσης των απαιτήσεων που λείπουν
− Ανίχνευση κατάστασης χρυσής επένδυσης

1.5 Εννοιολογικό πρότυπο ιχνών

Στο [Knethen, 2001] και [Knethen, 2002] μελετάται μια αλλαγή προσανατολισμένη στην ιχνηλασιμότητα για τα ενσωματωμένα συστήματα και τα αποτελέσματα της προσέγγισης της ανάλυσης αποτελούνται από τρία μέρη:

• προσεκτικά σχεδιασμένο πρότυπο ιχνών

• σύνολο αναλυτικών διαδικασιών που περιγράφουν πώς να καθιερώσουν τα ίχνη και πώς να αναλύσουν τις επιδράσεις των αλλαγών

• υποστήριξη εργαλείων που παρέχουν (ημι-)αυτόματες αναλύσεις και τη συνέπεια στον έλεγχο των εφαρμοσμένων αλλαγών.

Το πρότυπο ιχνών αποφασίζει τους τύπους τεκμηρίωσης οντοτήτων και σχέσεων που θα ανιχνευτούν, για να υποστηρίξουν την επίδραση και την εφαρμογή των αλλαγών σε ένα σύστημα απαιτήσεων. Το εννοιολογικό πρότυπο ιχνών αποτελείται από το εννοιολογικό πρότυπο συστημάτων και το εννοιολογικό πρότυπο τεκμηρίωσης. Αυτά τα πρότυπα περιγράφονται λεπτομερέστερα παρακάτω.

1.5.1 Εννοιολογικό πρότυπο συστημάτων

Ένα εννοιολογικό πρότυπο συστημάτων περιγράφει τους λογικούς τύπους οντοτήτων και τις εξαρτήσεις τους, και τις εκλεπτυσμένες σχέσεις που περιλαμβάνονται σε ένα σύστημα λογισμικού σε διαφορετικά επίπεδα [Palo, 2003].

Οι λογικοί τύποι οντοτήτων και οι σχέσεις τους εξαρτώνται από την περιοχή εφαρμογής που ερευνάται. Το εννοιολογικό πρότυπο συστημάτων διακρίνεται κυρίως μεταξύ των τύπων:

• Στοιχεία σε διαφορετικά επίπεδα αφαίρεσης (π.χ. ελεγχόμενο στοιχείο του περιβάλλοντος).

• Εργασίες σε διαφορετικά επίπεδα αφαίρεσης. Κάθε εργασία του συστήματος πρέπει να αφορά ένα σύνολο σχέσεων εξάρτησης μεταξύ των στοιχείων του περιβάλλοντος που παρακολουθείται και του ενός ελεγχόμενου στοιχείου.

• Σχέσεις εξάρτησης μεταξύ των τύπων οντοτήτων σε ένα επίπεδο αφαίρεσης (π.χ. τα στοιχεία που παρακολουθούνται πρέπει να έχουν σχέσεις επιρροής με τα ελεγχόμενα στοιχεία).

• Οι σχέσεις εκλέπτυνσης μεταξύ τύπων της οντότητας σε διαφορετικά επίπεδα αφαίρεσης (π.χ. το σύστημα εργασιών πρέπει να έχει μια σχέση εκλέπτυνσης με μια λειτουργία του λογισμικού και τουλάχιστον δύο του υλικού).

1.5.2 Εννοιολογικό πρότυπο τεκμηρίωσης

Ένα εννοιολογικό πρότυπο τεκμηρίωσης περιγράφει αντιπροσωπευτικούς τύπους οντοτήτων που περιλαμβάνονται σε διαφορετικά έγγραφα λογισμικού, σε διάφορα επίπεδα αφαίρεσης ενός συστήματος λογισμικού και τις σχέσεις τους. Εκτός από τις σχέσεις εξάρτησης και εκλέπτυνσης που μπορούν να ληφθούν από το εννοιολογικό πρότυπο συστημάτων, το εννοιολογικό πρότυπο τεκμηρίωσης περιλαμβάνει σχέσεις αντιπροσώπευσης. Οι τύποι οντοτήτων αντιπροσώπευσης και οι σχέσεις τους εξαρτώνται από τις τεχνικές προτύπων και περιγραφής προϊόντων που επιλέγονται. Το πρότυπο τεκμηρίωσης επεκτείνει τα στοιχεία περιγραφής για να επιτρέψει το σαφή προσδιορισμό κάθε λογικού τύπου οντοτήτων [Palo, 2003].

Διακρίνει κυρίως μεταξύ των τύπων:

• Οντότητες τεκμηρίωσης (π.χ. περιπτώσεις χρήσης συστημάτων ενός διαγράμματος περίπτωσης χρήσης συστημάτων, ή μέθοδοι σχεδίου λογισμικού ενός διαγράμματος κλάσης σχεδίου λογισμικού).

• Σχέσεις εξάρτησης μεταξύ των τύπων οντοτήτων τεκμηρίωσης στο ίδιο επίπεδο αφαίρεσης (π.χ. κάθε περίπτωση χρήσης συστημάτων πρέπει να έχει μια σχέση επιρροής με τον συσχετιζόμενο δράστη). Αυτές οι σχέσεις προέρχονται από το εννοιολογικό πρότυπο συστημάτων. Κάθε σχέση που περιγράφεται για έναν λογικό τύπο οντοτήτων πρέπει να ισχύει για έναν τύπο οντότητας τεκμηρίωσης που αντιπροσωπεύει το λογικό τύπο οντοτήτων.

• Οι σχέσεις εκλέπτυνσης μεταξύ τύπων οντότητας τεκμηρίωσης σε διαφορετικά επίπεδα αφαίρεσης (π.χ. κάθε περίπτωση χρήσης συστημάτων πρέπει να έχει μια σχέση εκλέπτυνσης σε ένα λογισμικό περίπτωση χρήσης). Αυτές οι σχέσεις εκλέπτυνσης προέρχονται από το εννοιολογικό σύστημα πρότυπων.

• Σχέσεις αντιπροσώπευσης μεταξύ των τύπων οντοτήτων τεκμηρίωσης που αντιπροσωπεύουν τον ίδιο λογικό τύπο οντοτήτων (π.χ. μια περίπτωση χρήσης πρέπει να έχει μια σχέση αντιπροσώπευσης σε μια περιγραφή περίπτωσης χρήσης επειδή και οι δύο αντιπροσωπεύουν έναν στόχο συστημάτων.

1.6 Πρότυπα αναφοράς ιχνηλασιμότητας

Τα πρότυπα αναφοράς γενικά είναι πρωτότυπα μοντέλα κάποιας περιοχής εφαρμογής. Ο σκοπός των προτύπων αναφοράς είναι να μειωθεί σημαντικά η δουλειά της δημιουργίας συγκεκριμένης εφαρμογής πρότυπων και συστημάτων : Ο χρήστης επιλέγει τα σχετικά μέρη του προτύπου αναφοράς, τα προσαρμόζει στο πρόβλημα, και διαμορφώνει μια γενική λύση από τα προσαρμοσμένα μέρη. Δεδομένου ότι η ανάλυση μιας συγκεκριμένης περιοχής χρειάζεται τεράστια προσπάθεια όταν αρχίζει από το μηδέν, η χρήση των προτύπων αναφοράς έχει

αναφερθεί ότι γλιτώνει μέχρι 80 τοις εκατό από τις δαπάνες ανάπτυξης για τα συστήματα σε τυποποιημένες περιοχές [Scheer, 1998].

Τα πρότυπα αναφοράς που περιγράφονται στο [Ramesh and Jarke, 2001] είναι βασισμένα σε διάφορες εμπειρικές μελέτες. Η συλλογή δεδομένων εκτείνεται σε μια περίοδο άνω των τριών ετών. Η κύρια μελέτη αποτελούνταν από 30 ομάδες συζητήσεων σε 26 οργανισμούς, οι οποίοι αφορούσαν μια ευρεία ποικιλία βιομηχανιών συμπεριλαμβανομένης π.χ. της άμυνας, αεροδιαστήματος, φαρμακευτικά είδη, ηλεκτρονική, και τηλεπικοινωνίες. Οι συμμετέχοντες είχαν έναν μέσο όρο 15.5 ετών εμπειρίας σε αρκετές περιοχές κλειδί της ανάπτυξης συστημάτων συμπεριλαμβανομένης π.χ. της τεχνολογίας λογισμικού, διαχείρισης απαιτήσεων, δοκιμή λογισμικού, ολοκλήρωση συστημάτων, ανάλυση συστημάτων, συντήρηση, και εφαρμογή λογισμικού.

Κάνοντας τη μελέτη έγινε προφανές ότι οι συμμετέχοντες θα μπορούσαν να ταξινομηθούν σε δύο ευδιάκριτες ομάδες όσον αφορά την πρακτική ιχνηλασιμότητας τους. Αυτές οι ομάδες αναφέρονται ως Low-end και High-end χρήστες ιχνηλασιμότητας και υπάρχουν χωριστά πρότυπα αναφοράς για αυτές τις ομάδες. Αυτά τα πρότυπα περιγράφονται στα υποκεφάλαια παρακάτω.

1.6.1 Low-End πρότυπο ιχνηλασιμότητας

Οι Low-end χρήστες ιχνηλασιμότητας έχουν τα ακόλουθα χαρακτηριστικά:

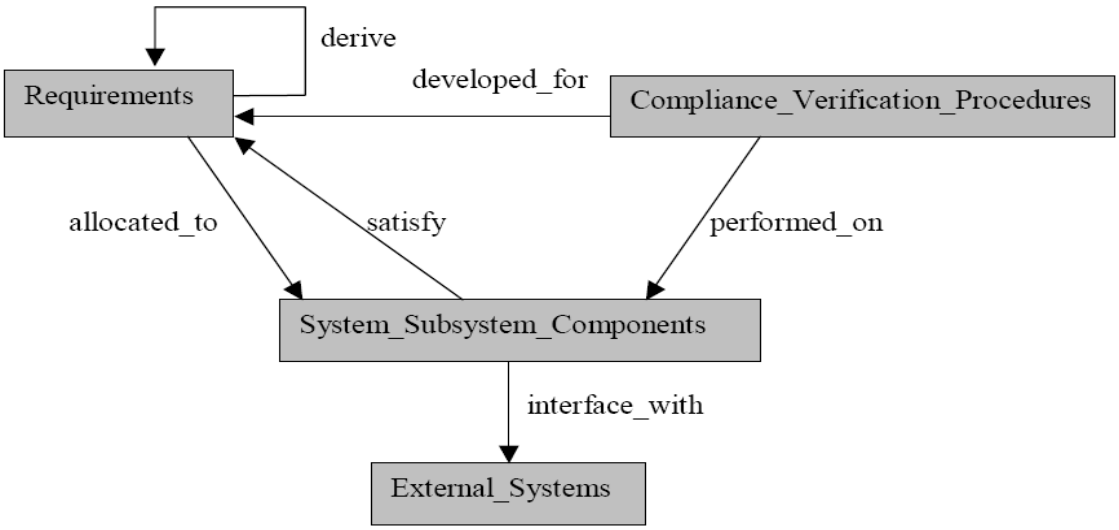
• Η χαρακτηριστική πολυπλοκότητα του συστήματος είναι περίπου 1000 απαιτήσεις

• Το επίπεδο εμπειρίας ιχνηλασιμότητας είναι από μηδέν έως δύο έτη

• Ο ορισμός χρηστών της ιχνηλασιμότητας είναι ο μετασχηματισμός εγγράφων των απαιτήσεων σε σχέδιο.

• Οι κύριες εφαρμογές της ιχνηλασιμότητας είναι αποσύνθεση απαιτήσεων, η κατανομή απαιτήσεων, η επαλήθευση και ο έλεγχος αλλαγών.

Το Low-end πρότυπο ιχνηλασιμότητας μπορείτε να το δείτε στο σχήμα 1.2.



Σχήμα 1.2 : Low-end πρότυπο ιχνηλασιμότητας.

Η χαρακτηριστική άποψη των Low-end χρηστών για την ιχνηλασιμότητα απαιτήσεων είναι ότι προβάλλει έναν σύνδεσμο από αρχικές απαιτήσεις στα πραγματικά τμήματα συστημάτων που ικανοποιούν εκείνες τις απαιτήσεις. Χαμηλότερου επιπέδου καθορισμένες απαιτήσεις προέρχονται από τις υψηλότερου επιπέδου απαιτήσεις συστημάτων. Οι αρχικές και οι παραγόμενες απαιτήσεις κατανέμονται στα τμήματα συστημάτων. Βρίσκοντας ποια συστατικά μέρη ικανοποιούν τις διάφορες απαιτήσεις και ποιες απαιτήσεις χαρτογραφούνται σε διαφορετικά συστατικά μέρη, ο σχεδιαστής είναι σε θέση να ελέγξει ότι όλες οι απαιτήσεις εξετάζονται από το σύστημα. Στη φάση επαλήθευσης της ανάπτυξης συστημάτων, οι Low-end χρήστες χρησιμοποιούν τη βάση δεδομένων απαιτήσεων, η οποία περιέχει την πιο τρέχουσα έκδοση του συστήματος επικυρωμένων απαιτήσεων , για να αναπτύξουν τις διαδικασίες επαλήθευσης συστημάτων όπως δοκιμές ή προσομοιώσεις. Εάν εμφανιζόταν μια αλλαγή στις απαιτήσεις, οι σύνδεσμοι της ιχνηλασιμότητας θα μπορούσαν να προσδιορίσουν τις διαδικασίες επαλήθευσης που πρέπει να τροποποιηθούν ή επαναξιολογηθούν. Οι διαδικασίες επαλήθευσης εκτελούνται στα συστατικά του συστήματος και ελέγχουν ότι το συστατικό ικανοποιεί τις απαιτήσεις. Αποτελέσματα των δοκιμών χρησιμοποιούνται για να ελέγξουν ότι το σύστημα λειτουργεί και ότι καλύπτει όλες τις απαιτήσεις. Το σύστημα των συστατικών μπορεί να εξαρτηθεί από άλλους και μπορεί επίσης να διασυνδεθεί με τα εξωτερικά συστήματα. Αυτές οι πληροφορίες χρησιμοποιούνται στην αξιολόγηση πώς μια απαίτηση ικανοποιείται από ένα συστατικό του συστήματος.

Οι Low-end χρήστες υστερούν ειδικά στον τομέα της σύλληψης της λογικής, για παράδειγμα ένας χρήστης παραθέτει το ακόλουθο: «Συχνά δεν έχουμε καμία ιδέα ποιοι έλαβαν αυτές τις αποφάσεις, και πώς επιδρούν στο υπόλοιπο της προσπάθειας. Απλά προσπαθούν να το κάνουν στο τέλος του προγράμματος ή εφόσον δεν λειτουργήσει. Συχνά οι άνθρωποι που εργάστηκαν σε αυτό, δουλεύουν χωρίς ένα ίχνος αυτών που συμβαίνουν, δεν είναι πειθαρχημένοι αρκετά για να τα τεκμηριώσουν με όλες τις απαιτήσεις τής ομάδας.»

1.6.2 High-End πρότυπο ιχνηλασιμότητας

Οι High-end χρήστες ιχνηλασιμότητας έχουν τα ακόλουθα χαρακτηριστικά:

• Η χαρακτηριστική πολυπλοκότητα του συστήματος είναι περίπου 10000 απαιτήσεις

• Το επίπεδο εμπειρίας ιχνηλασιμότητας είναι από πέντε έως δέκα έτη

• Ο ορισμός των χρηστών για την ιχνηλασιμότητα είναι ότι αυξάνει την πιθανότητα της παραγωγής ενός συστήματος που καλύπτει όλες τις απαιτήσεις πελατών και θα είναι εύκολο να διατηρηθεί

• Οι κύριες εφαρμογές της ιχνηλασιμότητας είναι η ολική κάλυψη του κύκλου ζωής συμπεριλαμβανομένου του χρήστη και του πελάτη, σύλληψη των ζητημάτων των συζητήσεων, σύλληψη λογικών αποφάσεων, εύρεση ιχνών μεταξύ προϊόντων και διαδικασιών.

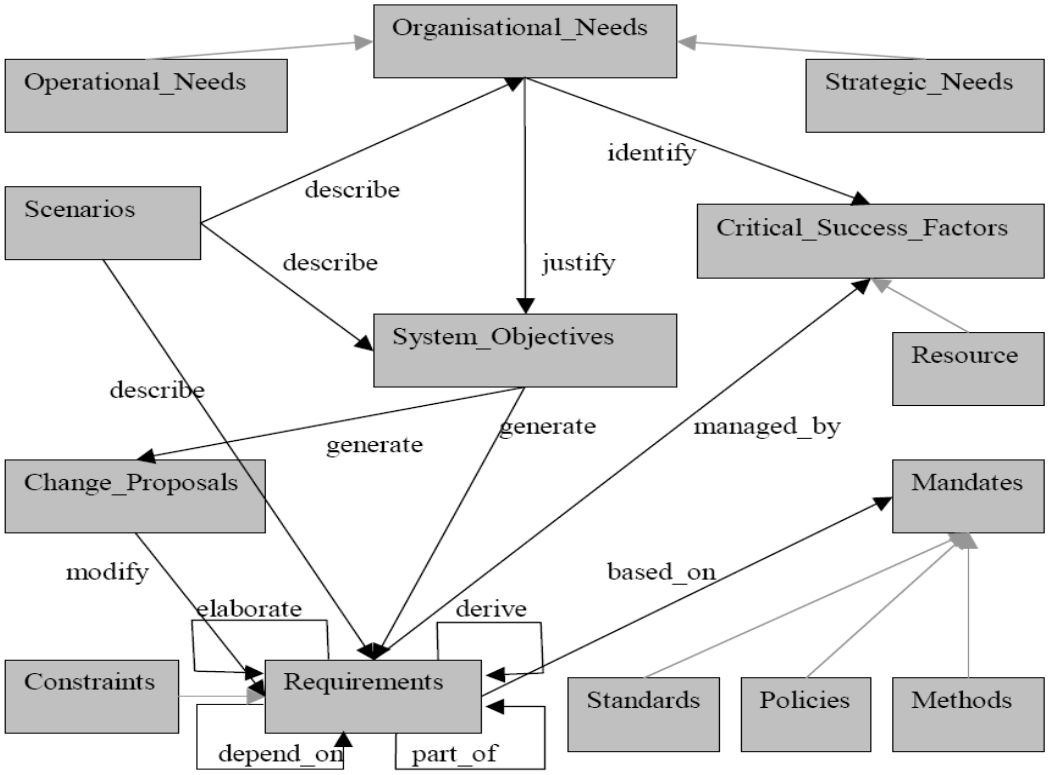
Οι High-end χρήστες ιχνηλασιμότητας υιοθετούν πολύ πλουσιότερα σχέδια ιχνηλασιμότητας από τους Low-end χρήστες και επίσης χρησιμοποιούν πληροφορίες ιχνηλασιμότητας με πολύ πλουσιότερους τρόπους. Επομένως το High-end πρότυπο διαιρείται σε τέσσερα μέρη για σαφήνεια:

• Διοικητικό υποπρότυπο απαιτήσεων
• Υποπρότυπο λογικής
• Υποπρότυπο κατανομής σχεδίου
• Υποπρότυπο επαλήθευσης.

Αυτά τα υποπρότυπα περιγράφονται στα υποκεφάλαια παρακάτω.

1.6.2.1 Διοικητικό υποπρότυπο απαιτήσεων

Με το διοικητικό υποπρότυπο απαιτήσεων οι απαιτήσεις μπορούν να ανιχνευθούν μέσω του κύκλου ζωής για να βοηθήσουν τους συμμετέχοντες να καταλάβουν και να αξιολογήσουν εάν το σύστημα υποστηρίζει τους κρίσιμους παράγοντες επιτυχίας. Το διοικητικό υποπρότυπο απαιτήσεων παρουσιάζεται στο σχήμα 1.3.



Σχήμα 1.3 : Διοικητικό υποπρότυπο απαιτήσεων.

Το διοικητικό υποπρότυπο απαιτήσεων λαμβάνει υπόψη τα ακόλουθα ζητήματα:

• τα συστήματα χτίζονται για να ικανοποιήσουν τις οργανωτικές ανάγκες

• οι οργανωτικές ανάγκες είναι λεπτομερείς στα σενάρια

• οι στόχοι συστημάτων δικαιολογούνται από τις οργανωτικές ανάγκες, οι συμμέτοχοι διευκρινίζουν τους στόχους του συστήματος

• οι απαιτήσεις παράγονται από τους στόχους του συστήματος

• οι οργανωτικές ανάγκες (π.χ. οι συμμέτοχοι) προσδιορίζουν τους κρίσιμους παράγοντες επιτυχίας, π.χ. οι πόροι μπορεί να είναι ένας από τους κρίσιμους παράγοντες επιτυχίας

• οι απαιτήσεις για το σύστημα ρυθμίζονται από τους κρίσιμους παράγοντες επιτυχίας

• οι απαιτήσεις μπορούν επίσης να βασιστούν στα πρότυπα, τις πολιτικές και σε μεθόδους

• οι περιορισμοί μπορούν να αντιμετωπιστούν ως τύπος απαίτησης

• οι χαμηλότερου επιπέδου απαιτήσεις προέρχονται από τις υψηλότερου επιπέδου απαιτήσεις

• μερικές απαιτήσεις διαμορφώνονται από άλλες, που παρέχουν την περαιτέρω εξήγηση ή διευκρίνιση

• απαιτήσεις επίσης εξαρτιούνται από άλλες απαιτήσεις

• οι σύνθετες απαιτήσεις αναλύονται συχνά στα συστατικά τους, προσδιορίζοντας απλούστερες απαιτήσεις που αποτελούν ένα μέρος τους.

1.6.2.2 Υποπρότυπο λογικής

Το υποπρότυπο λογικής διατηρεί τις πληροφορίες για το πώς οι αποφάσεις γίνονται για να επιλύσουν τα ζητήματα ή τις συγκρούσεις καθ' όλη τη διάρκεια του κύκλου της ζωής του συστήματος, για να εξασφαλιστεί ότι οι απαιτήσεις του πελάτη γίνονται κατανοητές και τον ικανοποιούν. Το υποπρότυπο λογικής λαμβάνει υπόψη τα ζητήματα:

- τα αντικείμενα (π.χ. συστατικά, απαιτήσεις, σχέδια) παράγουν τα ζητήματα ή τις συγκρούσεις
- τα ζητήματα επιλύονται με τις αποφάσεις
- οι αποφάσεις μπορούν να έχουν επιπτώσεις στις απαιτήσεις
- εξετάζονται οι διάφορες εναλλακτικές λύσεις που εξετάζουν την ευκρίνεια των ζητημάτων
- μπορούν να προταθούν επιχειρήματα υπέρ και ενάντια σε κάθε εναλλακτική λύση
- πρέπει να επιλεχτεί η ταυτότητα μιας ή περισσότερων εναλλακτικών λύσεων που επηρεάζεται συχνά από κρίσιμους παράγοντες επιτυχίας
- οι υποθέσεις που κρύβονται κάτω από τα διάφορα συστατικά της συζήτησης καταγράφονται επίσης

1.6.2.3 Υποπρότυπο κατανομής σχεδίου

Το υποπρότυπο κατανομής σχεδίου παρουσιάζει τις σχέσεις μεταξύ των απαιτήσεων και των συστατικών του σχεδίου. Το υποπρότυπο κατανομής σχεδίου λαμβάνει υπόψη τα ακόλουθα ζητήματα:

- σχέδιο κίνησης απαιτήσεων
- το σχέδιο είναι συχνά βασισμένο στις απαιτήσεις του περιβάλλοντος (π.χ. πρότυπα, πολιτικές ή μέθοδοι) που ελέγχουν τη δραστηριότητα ανάπτυξης συστημάτων
- τα τμήματα συστημάτων ή υποσυστημάτων είναι οι δομικές μονάδες του συστήματος και είναι καθορισμένα ή δημιουργημένα με τη διαδικασία σχεδίου
- οι απαιτήσεις διατίθενται στα συστατικά που είναι υποτιθέμενα για να τις ικανοποιήσουν
- τα συστατικά εξαρτώνται από άλλα συστατικά
- τα συστατικά μπορούν να είναι μέρος-άλλων συστατικών
- οι πηγές χρησιμοποιούνται από τα συστατικά
- οι λειτουργίες εκτελούνται από τα συστατικά
- οι λειτουργίες απευθύνονται στις απαιτήσεις
- τα συστατικά εξαρτώνται από τα εξωτερικά συστήματα.

1.6.2.4 Υποπρότυπο επαλήθευσης

Το υποπρότυπο επαλήθευσης χρησιμοποιείται για να πιστοποιήσει την πληρότητα και την ακρίβεια του συστήματος και να προσδιορίζει τις αλλαγές που είναι απαραίτητες για να επιτύχουν τους στόχους. Το υποπρότυπο επαλήθευσης λαμβάνει υπόψη τα ακόλουθα ζητήματα:

• η ανάπτυξη των διαδικασιών επαλήθευσης (π.χ. διαμόρφωση πρωτοτύπου, προσομοίωση, η δοκιμή και η επιθεώρηση) διαχειρίζεται από τη χρήση πόρων τους

• οι απαιτήσεις του περιβάλλοντος (π.χ. πρότυπα, πολιτικές ή μέθοδοι) είναι συνήθως η βάση της διαδικασίας επαλήθευσης και καθορίζουν ποιες διαδικασίες απαιτούνται και πώς αυτές θα εκτελεστούν

• οι διαδικασίες επαλήθευσης ελέγχουν πώς τα συστατικά ικανοποιούν τις απαιτήσεις ή βοηθούν για να παράγουν προτάσεις αλλαγής για τις απαιτήσεις, ή για το σχέδιο ή την εφαρμογή.

2. Εφαρμογή της ιχνηλασιμότητας

Ένας τεχνικός ελέγχου λογισμικού είχε μια εφαρμογή που την δούλευε για μερικά έτη. Την σταμάτησε και ξανάρχισε αρκετές φορές. Μετά από δύο έτη, είχε ολοκληρωθεί αρκετά, αλλά οι απαιτήσεις ήταν προχειροδουλεμένες. Μερικές είχαν αλλάξει προτού να αρχίσει η ανάπτυξη, μερικές είχαν προστεθεί ή διαγραφεί, και μερικές είχαν αλλάξει κατά τη διάρκεια της ανάπτυξης. Το σύστημα βασίστηκε σε μια σύνθετη νομοθεσία που είχε περάσει από μια αλλαγή κατά τη διάρκεια εκείνης της περιόδου.

Όταν αποφασίστηκε να ολοκληρωθεί η ανάπτυξη, κανένας δεν ήξερε τους σύνθετους επιχειρησιακούς κανόνες που ενσωματώνονταν στο λογισμικό. Κανένας δεν ήξερε εάν κάλυψε πραγματικά τις απαιτήσεις της τρέχουσας νομοθεσίας. Κανένας δεν ήξερε εάν οι επιχειρησιακοί κανόνες εφαρμόστηκαν με συνέπεια σε όλες τις καταστάσεις. Παραδείγματος χάριν, εάν τα οφέλη ήταν μόνο διαθέσιμα σε ανθρώπους μιας συγκεκριμένης κλίμακας ηλικιών, και η

νομοθεσία της κλίμακας ηλικιών είχε αλλάξει κατά τη διάρκεια του περασμένου χρόνου, η εφαρμογή εξασφάλισε ποιοι πρέπει να είναι στη νέα κλίμακα; Το πρόβλημα επιδεινώθηκε από το γεγονός ότι κανένας δεν ήξερε που εφαρμόστηκαν αυτοί οι κανόνες στον κώδικα. Ο μόνος τρόπος να το ανακαλύψουν ήταν αν διάβαζαν τα σχετικά μέρη του κώδικα γραμμή προς γραμμή.

Στο τέλος, μια ομάδα υπεύθυνων για την ανάπτυξη εξέτασε κάθε γραμμή κώδικα, και αναζήτησε τους επιχειρησιακούς κανόνες. Ο κώδικας ξαναγράφηκε σε πολλά σημεία για να αφαιρεθεί τη σκληρή κωδικοποίηση των κανόνων. Το κόστος ήταν ουσιαστικό και υπήρξε καθυστέρηση στο πρόγραμμα.

Εάν ένα προσχέδιο εργασίας είχε παρουσιαστεί και υπαγόταν σε διάφορες αλλαγές, ο συντάκτης θα του έδινε αυτόματα έναν αριθμό έκδοσης. Με τις απαιτήσεις, που υπόκεινται επίσης στις αλλαγές, η έκδοση συχνά δεν εφαρμόζεται. Μπορεί να είχαν υπάρξει απαιτήσεις που καταχωρήθηκαν στο έγγραφο τον Ιανουάριο, και άλλες το Φεβρουάριο, αλλά τι ακριβώς έχει αλλάξει; Το τελικό έγγραφο απαιτήσεων αντιπροσωπεύει τι δημιουργείται;

Ένα πολύ χαρακτηριστικό πρόβλημα είναι ότι υπάρχει ένας τεχνικός λογισμικού αρμόδιος για τη συλλογή και τη διαχείριση των απαιτήσεων. Εάν ο αναλυτής αφήσει τη δουλειά του σε κάποιον άλλο, που πρέπει να διαχειριστεί το πρόγραμμα, είναι αδύνατο να γνωρίζει από πού προήλθαν οι απαιτήσεις. Ένα έγγραφο των εκατοντάδων απαιτήσεων θα αναπτυχθεί και κανένας δεν θα ξέρει στα σίγουρα από πού προήλθαν ορισμένες απαιτήσεις.

Εάν οι αρχικοί στόχοι καθορίζονταν και δίνονταν σε έναν δημιουργό, θα μπορούσανε να χαρτογραφηθούν τα χαρακτηριστικά γνωρίσματα εκείνων των στόχων και των απαιτήσεων εκείνου του χαρακτηριστικού γνωρίσματος. Ο δημιουργός θα ξέρει πάντα από που προήλθε η απαίτηση και ποιον γενικό στόχο ικανοποιεί. Έτσι οι υπεύθυνοι για την ανάπτυξη θα μπορούν πάντα να ξέρουν τι κάνουν και η ομάδα προγράμματος να καταλαβαίνει τι δημιουργεί ο υπεύθυνος για την ανάπτυξη. Η ιχνηλασιμότητα απαιτήσεων είναι ένας καλός τρόπος να μετρηθεί η πληρότητα και να συμπληρωθούν οποιαδήποτε χάσματα.

2.1 Καθιέρωση της ιχνηλασιμότητας απαιτήσεων

Η ιχνηλασιμότητα είναι σημαντική για διάφορους λόγους. Αυτοί περιλαμβάνουν:

- Γνώση των τρεχόντων απαιτήσεων
- Γνώση των αλλαγών των απαιτήσεων
- Τεκμηριωμένη βάση δοκιμών
- Κατανόηση των απαιτήσεων που έχουν χτιστεί μέσα στο σύστημα
- Δημιουργία μιας βάσης για την τεκμηρίωση τρεχόντων συστημάτων

Ενώ είναι ιδανικό να υπάρχει ένα εργαλείο για να διαχειριζόμαστε τις απαιτήσεις, η μη κατοχή ενός τέτοιου εργαλείου δεν είναι λόγος να μην ανιχνεύουμε τις απαιτήσεις. Μπορούμε να δημιουργήσουμε εύκολα ένα απλό χειρωνακτικό σύστημα ώστε να καταγράφουμε τις διευκρινίσεις μας.

2.2 Μήτρα ιχνηλασιμότητας

Η υποστήριξη για την ιχνηλασιμότητα λογισμικού επιτυγχάνεται με τη διαχείριση των σχέσεων μεταξύ των τεχνουργημάτων λογισμικού. Η διαχείριση αυτών των σχέσεων στοχεύει στην υποστήριξη (αυτόματα, ημιαυτόματα ή με το χέρι) της δημιουργίας, της συνέχισης, της συντήρησης, και της καταστροφής των σημαντικών σχέσεων στα τεχνουργήματα λογισμικού [Alexander, 2002], [Spanoudakis and Zisman, 2005]. Στοχεύει επίσης στην περιγραφή και την ταξινόμηση αυτών των σχέσεων [Alexander, 2002], [Lindval and Sandahl, 1996], [Spanoudakis and Zisman, 2005]. Η καθιέρωση και η διατήρηση των σχέσεων μεταξύ των τεχνουργημάτων λογισμικού είναι σημαντική επειδή οι σχέσεις μπορούν να χρησιμοποιηθούν σε διάφορες δραστηριότητες τεχνολογίας λογισμικού όπως η ανάλυση αντίκτυπου αλλαγής λογισμικού και η επικύρωση, η επαλήθευση και η δοκιμή λογισμικού [Spanoudakis and Zisman, 2005]. Μια ομοιότητα μεταξύ της βασισμένης σε πρότυπο δοκιμής και της ιχνηλασιμότητας είναι η ανάγκη να ρυθμιστούν οι σχέσεις μεταξύ των τεχνουργημάτων.

Όσον αφορά την ιχνηλασιμότητα των τεχνουργημάτων δοκιμής, οι διαθέσιμες λύσεις τυπικά χρησιμοποιούν μια μήτρα ιχνηλασιμότητας για να υποστηρίξουν τη διαχείριση σχέσεων μεταξύ των απαιτήσεων και των περιπτώσεων δοκιμής. Ένα παράδειγμα από μια ημιαυτόματη λύση [Bouquet et al., 2003] δημιουργεί μια μήτρα ιχνηλασιμότητας από τις απαιτήσεις στις περιπτώσεις δοκιμής κατά τη διάρκεια της διαδικασίας παραγωγής δοκιμής. Όταν οι περιπτώσεις δοκιμής παράγονται από τα πρότυπα, τα προσδιοριστικά χρησιμοποιούνται για να δημιουργήσουν τη μήτρα ιχνηλασιμότητας που συσχετίζει τα προσδιοριστικά των απαιτήσεων με τα προσδιοριστικά των περιπτώσεων δοκιμής. Η προσέγγισή μας προτείνει επίσης τη δημιουργία σχέσεων μεταξύ δοκιμαστικών τεχνουργημάτων κατά τη διάρκεια της διαδικασίας παραγωγής δοκιμής. Εντούτοις, δεν συσχετίζει τις απαιτήσεις με τις περιπτώσεις δοκιμής, αλλά με τις οντότητες που συνθέτουν τα πρότυπα σε μια ιεραρχία δοκιμής, και δεν απαιτεί προηγούμενο σχολιασμό πρότυπων. Επιπλέον, το επίπεδο καταμερισμού των σχέσεων στις μήτρες ιχνηλασιμότητας είναι υψηλό και σταθερό, ενώ η προσέγγιση ιχνηλασιμότητας που υιοθετούμε επιτρέπει το επίπεδο καταμερισμού να ποικίλει. Αυτό θα μπορούσε να είναι ένας σοβαρός παράγοντας όταν ο στόχος είναι να υποστηριχτούν οι δραστηριότητες εκτός από την παραγωγή (π.χ. βασισμένη δοκιμή οπισθοδρόμησης).

Σχετικά με τις σχέσεις, οι αυτοματοποιημένες βασισμένες σε πρότυπο δοκιμαστικές προσεγγίσεις εκμεταλλεύονται τις υπονοούμενες ή/και σαφής σχέσεις. Οι υπονοούμενες σχέσεις ενσωματώνονται στους αλγορίθμους και τα πρότυπα του εργαλείου, ενώ οι σαφώς προσδιορισμένες σχέσεις είτε δημιουργούνται αυτόματα και γίνονται ακριβής από το εργαλείο, είτε δημιουργούνται από τους χρήστες. Μερικές προσεγγίσεις χρησιμοποιούν τις υπονοούμενες σχέσεις για να υποστηρίξουν την παραγωγή, την εκτέλεση και την αξιολόγηση δοκιμής [Fraikin and Leonhardt, 2002], [Nebut et al., 2006], [Wittevrongel et al., 2001] ενώ άλλοι χρησιμοποιούν τις υπονοούμενες σχέσεις για να υποστηρίξουν τη δοκιμή οπισθοδρόμησης [Briand and Labiche, 2001]. Οι πρόσθετες προσεγγίσεις χρησιμοποιούν τις σαφώς προσδιορισμένες σχέσεις για να υποστηρίξουν την παραγωγή δοκιμής [Basanieri et al., 2002], την εκτέλεση και την αξιολόγηση δοκιμής [Grieskamp et al., 2003], [Hartman and Nagin, 2004], ή την ανάλυση κάλυψης [Hartman and Nagin , 2004].

Η μήτρα ιχνηλασιμότητας απαιτήσεων είναι το κλειδί για να είσαι σε θέση να γνωρίζεις οπτικά τις απαιτήσεις που επισημαίνονται. Η μήτρα παρουσιάζει τον πίνακα των

χαρακτηριστικών γνωρισμάτων, και για κάθε χαρακτηριστικό γνώρισμα και μπορείς να δεις εάν υπάρχει προς τα πίσω και μπροστά ιχνηλασιμότητα – δηλαδή κάθε χαρακτηριστικό γνώρισμα χαρτογραφείται πίσω από έναν στόχο. Είναι κάθε χαρακτηριστικό γνώρισμα πλήρες, σύμφωνα με τις απαιτήσεις; Όταν βλέπετε ένα χαρακτηριστικό γνώρισμα χωρίς οποιαδήποτε ιχνηλασιμότητα πρέπει να επισημανθεί στην ομάδα προγράμματος ότι οι απαιτήσεις είναι ελλιπείς.

2.2.1 Εκτέλεση ιχνηλασιμότητας

Ο κλασικός τρόπος να εκτελεσθεί η ιχνηλασιμότητα είναι με την κατασκευή μιας μήτρας ιχνηλασιμότητας. Όπως διευκρινίζεται στον πίνακα 2.1, μια μήτρα ιχνηλασιμότητας συνοψίζει με μορφή μητρών, την ιχνηλασιμότητα από τις αρχικές ανάγκες των συμμέτοχων με τις απαιτήσεις του σχετικού προϊόντος τους και έπειτα προς άλλα στοιχεία προϊόντων εργασίας. Προκειμένου να κατασκευαστεί μια μήτρα ιχνηλασιμότητας, κάθε απαίτηση, κάθε πηγή απαιτήσεων και κάθε στοιχείο προϊόντων εργασίας πρέπει να έχουν ένα μοναδικό προσδιοριστικό που μπορεί να χρησιμοποιηθεί ως αναφορά στη μήτρα. Η μήτρα απαιτήσεων έχει το πλεονέκτημα ότι λειτουργεί ως αποθήκη για να τεκμηριώσει και την προς τα εμπρός και την προς τα πίσω ιχνηλασιμότητα σε όλα τα προϊόντα εργασίας

Requirement Source	Product Requirements	HLD Section #	LLD Section #	Code Unit	UTS Case #	STS Case #	User Manual
Business Rule #1	R00120 Credit Card Types	4.1 Parse Mag Strip	4.1.1 Read Card Type	Read_Card_Type.c Read_Card_Type.h	UT 4.1.032 UT 4.1.033 UT 4.1.038 UT 4.1.043	ST 120.020 ST 120.021 ST 120.022	Section 12
			4.1.2 Verify Card Type	Ver_Card_Type.c Ver_Card_Type.h Ver_Card_Types.dat	UT 4.2.012 UT 4.2.013 UT 4.2.016 UT 4.2.031 UT 4.2.045	ST 120.035 ST 120.036 ST 120.037 ST 120.037	Section 12
Use Case #132 step 6	R00230 Read Gas Flow	7.2.2 Gas Flow Meter Interface	7.2.2 Read Gas Flow Indicator	Read_Gas_Flow.c	UT 7.2.043 UT 7.2.044	ST 230.002 ST 230.003	Section 21.1.2
	R00231 Calculate Gas Price	7.3 Calculate Gas price	7.3 Calculate Gas price	Cal_Gas_Price.c	UT 7.3.005 UT 7.3.006 UT 7.3.007	ST 231.001 ST 231.002 ST 231.003	Section 21.1.3

Πίνακας 2.1: Παράδειγμα μιας μήτρας ιχνηλασιμότητας.

Μια απλή μήτρα ιχνηλασιμότητας απαιτήσεων να μοιάσει με αυτήν:

ID	User Requirements	System Reference
UF1	Add new customers	S1, S2
UBR4	Cannot add a user if they already exist	S1, S55
UD5	User surname is mandatory	S1
Etc		

Πίνακας 2.2: Μια απλή μήτρα ιχνηλασιμότητας.

Σε αυτό το παράδειγμα, υπάρχουν διάφορα IDs
- UF είναι «λειτουργία χρηστών»
- UBR είναι «επιχειρησιακός κανόνας χρηστών»
- UD είναι «δεδομένα χρηστών»

Τα πραγματικά IDs καθορίζονται από την τεχνική που χρησιμοποιείται για να τεκμηριώσει απαιτήσεις. Παραδείγματος χάριν μπορείτε να αναφερθείτε στις περιπτώσεις χρήσης χρησιμοποιώντας τον αριθμό τους.

Η αναφορά συστημάτων θα καθοριστεί από το πώς το σύστημα είναι δομημένο. Εάν η απαίτηση χρησιμοποιείται σε περισσότερα από ένα σημεία, πρέπει να σημειωθεί. Κατά αυτόν τον τρόπο, οι ελεγκτές ξέρουν ποια μέρη του συστήματος να ελέγξουν για τις ιδιαίτερες περιπτώσεις, και εάν μια αλλαγή γίνει, πρέπει να γίνει σε όλα τα μέρη - π.χ. δυνατότητα να προστεθεί ένας χρήστης εάν υπάρχουν ήδη άλλοι.

Συμβατότητα προς τα πίσω

Μπορούμε να πάρουμε τις απαιτήσεις και να κοιτάξουμε προς τα εμπρός αλλά θέλουμε επίσης να κοιτάξουμε και πίσω στο σύστημα, και να καθορίσουμε ποιοι κανόνες προσκρούουν σε κάθε μέρος του συστήματος.

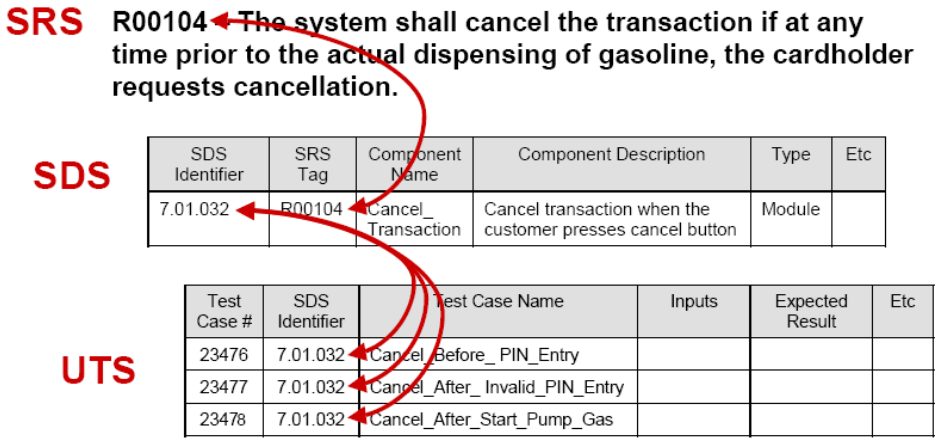
Μια μήτρα προς τα πίσω μοιάζει με αυτήν:

System Reference	Functional Requirement	ID
S1	The system shall enable customers to be added	UF1, UBR4, UD5

Πίνακας 2.3: Μια μήτρα προς τα πίσω.

Κατά αυτόν τον τρόπο, ο υπεύθυνος για την ανάπτυξη μπορεί να δει τους κανόνες και τις απαιτήσεις για κάθε μονάδα κώδικα. Ο ελεγκτής ξέρει επίσης ποιες λειτουργίες να ελέγξει όταν εστιάζει σε μια ιδιαίτερη αποτύπωση κώδικα.

2.3 Τοποθέτηση Συνδέσμων Ιχνηλασιμότητας



Σχήμα 2.1: Παράδειγμα τοποθέτησης συνδέσμων ιχνηλασιμότητας.

Ένας ακόμη μηχανισμός για την ιχνηλασιμότητα είναι η τοποθέτηση συνδέσμων ιχνηλασιμότητας. Πάλι, κάθε απαίτηση, κάθε πηγή απαίτησης και κάθε στοιχείο προϊόντων εργασίας πρέπει να έχει ένα μοναδικό προσδιοριστικό. Στην τοποθέτηση συνδέσμων ιχνηλασιμότητας εντούτοις, τα προσδιοριστικά χρησιμοποιούνται ως ετικέτες στα επόμενα προϊόντα εργασίας για να προσδιορίσουν προς τα πίσω, στο έγγραφο προκατόχων. Όπως φαίνεται στο σχήμα 2.1 παραδείγματος χάριν, η Software Design Specification (SDS) περιλαμβάνει τους συνδέσμους που προσδιορίζουν τις απαιτήσεις που εφαρμόζονται από κάθε μοναδικά προσδιορισμένο στοιχείο σχεδιασμού και η Unit Test Specification (UTS) περιλαμβάνει τους συνδέσμους ιχνηλασιμότητας που ανιχνεύουν πίσω στο σχέδιο τα στοιχεία που αφορούν κάθε περίπτωση δοκιμής. Η τοποθέτηση συνδέσμων ιχνηλασιμότητας διαδίδεται μέσω του προϊόντος εργασίας που τίθεται από τις μονάδες πηγαίου κώδικα, τη συμπερίληψη συνδέσμων ιχνηλασιμότητας πίσω στα στοιχεία σχεδίου και στις περιπτώσεις δοκιμής , με τους συνδέσμους πίσω στα στοιχεία αρχιτεκτονικής και στις περιπτώσεις δοκιμής συστημάτων, με συνδέσμους ιχνηλασιμότητας πίσω στις απαιτήσεις ανάλογα με την περίπτωση και ανάλογα με την ιεραρχία των προϊόντων εργασίας που χρησιμοποιούνται στο προϊόν. Οι σύνδεσμοι ιχνηλασιμότητας έχουν το πλεονέκτημα ότι είναι κομμάτι των προϊόντων εργασίας και έτσι δεν χρειάζεται να διατηρείται μια χωριστή μήτρα. Εντούτοις, ενώ η προς τα πίσω να επισήμανση είναι εύκολη με συνδέσμους ιχνηλασιμότητας, η προς τα εμπρός επισήμανση χρησιμοποιώντας αυτόν τον μηχανισμό είναι πολύ δύσκολη.

2.4 Εφαρμογή της ιχνηλασιμότητας με εργαλεία

Κάθε σημαντικός κατασκευαστής ηλεκτρονικών εργαλείων δίνει τη δυνατότητα υπολογισμών με λογιστικά φύλλα (spreadsheet) ή/και βάσεις δεδομένων που μπορούν να βοηθήσουν στην επισήμανση ιχνών απαιτήσεων. Υπάρχουν τουλάχιστον δέκα εμπορικά προϊόντα που εμπίπτουν στην κατηγορία workbench και υποστηρίζουν κάποιο επίπεδο ιχνηλασιμότητας απαιτήσεων. Τουλάχιστον, παρέχουν :

- αμφίδρομες απαιτήσεις που συνδέονται με τα στοιχεία συστημάτων
- σύλληψη της λογικής, υπευθυνότητα, και δοκιμή/ επικύρωση
- προσδιορισμός των ασυνεπειών
- ικανότητες σύνδεσης ιχνών
- επαλήθευση των απαιτήσεων
- ιστορικό των αλλαγών των απαιτήσεων

Ερευνώνται διαφορετικοί τύποι οργάνωσης της υποστήριξης της ιχνηλασιμότητας απαιτήσεων μέσα από τις απαιτήσεις που συλλέγονται από τη φάση του κύκλου ζωής λογισμικού. Περιλαμβάνουν την ανάπτυξη μιας κοινής γλώσσας, μιας μεθόδου, ενός προτύπου, και της δομής βάσεων δεδομένων, καθώς επίσης και μηχανισμών για να παρέχουν την ανταλλαγή στοιχείων μεταξύ διαφορετικών εργαλείων στο περιβάλλον. Τα πρωτότυπα υπάρχουν και τουλάχιστον ένα εμπορικό προϊόν παρέχει την υποστήριξη για την ανταλλαγή στοιχείων μέσω των αντικειμενοστραφών εγκαταστάσεων βάσεων δεδομένων.

Υπάρχουν πολλά διαθέσιμα εργαλεία για να διαχειριστούν την ιχνηλασιμότητα. Για τα λιγότερα πολυσύνθετα συστήματα, αρκεί κάποια βασική τεκμηρίωση. Παραδείγματος χάριν, στο Word, η χρήση του κρυμμένου κειμένου σε ένα έγγραφο είναι χρήσιμη. Χρησιμοποιείτε Format, Font, Hidden. Μπορείτε να κρύψετε κάποιες πληροφορίες, και να προσθέσετε μια υποσημείωση. Να είστε βέβαιος ότι έχετε εισαγάγει την ημερομηνία της αλλαγής. Μια άλλη τεχνική εάν χρησιμοποιείται το Excel είναι να κρυφτεί η παλαιά σειρά ή η στήλη και να προστεθεί μια σημείωση στο κελί με τους λόγους της αλλαγής. Εάν χρησιμοποιείται μια προσωπική βάση δεδομένων όπως το MS-Access, μπορεί να αναπτυχθεί μια σχετικά απλή βάση δεδομένων που κρατά τις προγενέστερες εκδόσεις των απαιτήσεων.

Τα εργαλεία διαχείρισης απαιτήσεων προσφέρουν μια ικανοποιητική επεξεργασία για τις κειμενικές προδιαγραφές αλλά έχουν δυσχέρειες κατά την ενσωμάτωση των προδιαγραφών με άλλες που δεν εκφράζονται κειμενικά. Αυτή η εφαρμογή είναι βασισμένη στους μηχανισμούς εισαγωγών που συνδέονται με εργαλεία τεχνολογίας λογισμικού με τη βοήθεια υπολογιστή (CASE tools). Συνήθως, σε αυτή την προσέγγιση, τα ονόματα των στοιχείων διαμόρφωσης καθιερώνονται στα πλαίσια των εργαλείων διαχείρισης απαιτήσεων.

Ένα πολύ διαδεδομένο εργαλείο είναι το TOOR (Traceability of Object-Oriented Requirements - ιχνηλασιμότητα από τις αντικειμενοστραφείς απαιτήσεις), που παρουσιάζουν οι

Pinheiro και Goguen [Pinheiro and Goguen, 1996] αυτό είναι βασισμένο σε FOOPS, μια επίσημη αντικειμενοστραφή γλώσσα (formal object-oriented language). Περιέργως, δεν υπάρχει οποιεσδήποτε εργασία για το TOOR πέρα από το αναφερθέν έγγραφο. Εντούτοις, η επίσημη προσέγγιση και η λειτουργία που περιγράφει παραμένει ενδιαφέρουσα.

Στο Rational RequisitePro (www.rational.com) οι κειμενικές προδιαγραφές μπορούν να συνδέονται με τα στοιχεία UML προτύπου στη βάση του Rational Rose . Αλλά αυτό είναι δυνατό μόνο για τις περιπτώσεις χρήσεις, άλλα πρότυπα δεν είναι άμεσα προσιτά από το Rational RequisitePro. Ο μόνος τύπος συνδέσμου του RequisitePro είναι το traceTo.

Άλλο γνωστό εργαλείο διαχείρισης απαιτήσεων είναι το Telelogic DOORS (www.telelogic.com). Αυτό το εργαλείο μπορεί να είναι συνδεμένο με τα περισσότερα δημοφιλή εργαλεία τεχνολογίας λογισμικού με τη βοήθεια υπολογιστή (CASE tools). Η χρήση του είναι παρόμοια με τους μηχανισμούς εισαγωγής του Rational RequisitePro, αλλά παρέχει περισσότερες λειτουργίες και ικανότητα στην αλλαγή από το πλαίσιο DOORS στο πλαίσιο εργαλείων τεχνολογίας λογισμικού με τη βοήθεια υπολογιστή (CASE tools). Εν πάση περιπτώσει, ο χρήστης πρέπει να εργαστεί με δύο χωριστά περιβάλλοντα, και πρέπει να επιλέξει το περιβάλλον ανάλογα με το εάν θέλει για να κάνει τη διαχείριση απαιτήσεων ή τη διαμόρφωση λογισμικού. Όλα τα παραπάνω εργαλεία έχουν δυσκολίες μέχρι τη διαμόρφωση της ιχνηλασιμότητας στις ανάγκες του προγράμματος. Δεν είναι προσανατολισμένοι σε μια συγκεκριμένη διαδικασία λογισμικού και αν και μερικοί απ' αυτούς ορίζουν τύπους απαιτήσεων, δεν προσφέρουν ένα πλαίσιο για τη διαμόρφωση απαιτήσεων ιχνηλασιμότητας. Τελικά, όλοι οι ορισμοί και η ερμηνεία για τις πληροφορίες της ιχνηλασιμότητας αφήνονται στο χρήστη του εργαλείου.

Το SeDiTeC (Κέντρο δοκιμής διαγραμμάτων ακολουθίας - Sequence Diagram Test Center) [Fraikin and Leonhardt, 2002] υποστηρίζει την εκτέλεση των διαγραμμάτων ακολουθίας, έτσι χρησιμοποιεί τα διαγράμματα ακολουθίας για να εξετάσει τις συσχετίσεις μεταξύ των αντικειμένων. Ο ελεγκτής είναι υπεύθυνος για την περιγραφή των συγκεκριμένων παραμέτρων και των αναμενόμενων αποτελεσμάτων των διαγραμμάτων ακολουθίας. Ο χρήστης δημιουργεί με το χέρι τις συγκεκριμένες παραμέτρους και τα αναμενόμενα αποτελέσματα. Η αξιολόγηση αποτελέσματος είναι βασισμένη στην οργάνωση πηγαίου κώδικα για να καταγράψει τις επιτευχθείσες κλήσεις μεθόδου, τις συγκεκριμένες παραμέτρους και τα επιτευχθέντα αποτελέσματα.

Το SCENTOR [Wittevrongel and Maurer, 2001] υποστηρίζει έγγραφα παραγωγής μέσω της κλάσης extending JUnit framework για να παραγάγει τα συγκεκριμένα χειρόγραφα δοκιμής. Ομοίως με το SeDiTeC, ο ελεγκτής είναι υπεύθυνος για τη διευκρίνιση των συγκεκριμένων τιμών παραμέτρου των μεθόδων και των αναμενόμενων αποτελεσμάτων. Αυτή η προδιαγραφή γράφεται σε Java, η οποία είναι η γλώσσα που χρησιμοποιείται από το πλαίσιο JUnit. Ο χρήστης διευκρινίζει επίσης τις συγκεκριμένες τιμές παραμέτρου των μεθόδων και τα αναμενόμενα αποτελέσματα, ενώ η υπόθεση της άμεσης χαρτογράφησης μεταξύ των διαγραμμάτων ακολουθίας και της εφαρμογής υποστηρίζει τη συγκεκριμένη παραγωγή χειρογράφων δοκιμής.

Το COWtest pluS UIT Environment (COW_SUITE) [Basanieri et al., 2002] είναι ένα εργαλείο που παρέχει συστηματική και αυτοματοποιημένη υποστήριξη για τον προγραμματισμό δοκιμών και την παραγωγή περιπτώσεων δοκιμής, βασισμένο στα διαγράμματα UML(περίπτωσης χρήσης, ακολουθίας, και διαγράμματα κλάσεων). Είναι κατά ένα μεγάλο μέρος βασισμένο στο χώρισμα των κλάσεων. Αναλύει τα διαγράμματα ακολουθίας και αλληλεπιδρά με τους χρήστες για να συλλέξει τις τιμές επιλογών, και τους περιορισμούς για την περίπτωση των

αντιφατικών συνδυασμών επιλογής. Οι σχέσεις μεταξύ των προτύπων UML γίνονται σαφής όταν το πρότυπο φορτώνεται στο εργαλείο.

Το TOTEM (Δοκιμή των αντικειμενοστραφών συστημάτων με την ενοποιημένη γλώσσα διαμόρφωσης) [Briand et al., 2002] υποστηρίζει την παραγωγή δοκιμής, ενώ μια επέκταση του υποστηρίζει τη δοκιμή οπισθοδρόμησης βασισμένη στα διαγράμματα UML και τις σχέσεις μεταξύ αυτών των διαγραμμάτων. Το TOTEM αντίθετα με την προσέγγιση αυτής της εργασίας, χρησιμοποιεί μη σαφώς προσδιορισμένες και υπονοούμενος σχέσεις. Τα πειράματα με το εργαλείο έχουν δείξει ότι η ακρίβεια της δοκιμής οπισθοδρόμησης ήταν περιορισμένη, και απαιτήθηκε περαιτέρω έρευνα που να συμπεριλαμβάνει όρους «φρουρούς».

Το AGEDIS (Automated Generation and Execution of test suites for Distributed component-based Software) [Hartman, 2004] [Hartman and Nagin, 2004] χρησιμοποιεί τις ακριβής σχέσεις που δημιουργούνται από το χρήστη για να εκτελέσει και να αξιολογήσει τις δοκιμές. Σε αυτήν την περίπτωση, οι σχέσεις χαρτογραφούν τα αφηρημένα ερεθίσματα από τις κλήσεις των μεθόδων, και τις αφηρημένες παρατηρήσεις από τον έλεγχο της αξιολόγησης. Το AGEDIS εξωτερικεύει επίσης τις σχέσεις μεταξύ των αφηρημένων ακολουθιών δοκιμής και των αποτελεσμάτων ιχνών δοκιμής κατά τη διάρκεια της εκτέλεσης δοκιμής. Υποστηρίζει την απεικόνιση των ιχνών δοκιμής και της αφηρημένης ακολουθίας δοκιμής που τα παρήγαγε, η οποία υποστηρίζει τη χειρωνακτική ανάλυση κάλυψης.

Η AsmL (Abstract state machine Language) [Grieskamp et al., 2003] επίσης χρησιμοποιεί τις ρητές σχέσεις για να εκτελέσει και να αξιολογήσει τα αφηρημένα χειρόγραφα δοκιμής. Υποστηρίζει την παράλληλη εκτέλεση του προτύπου και της εφαρμογής του.

Στο [Dinh-Trong and Ghosh, 2006], μια γραφική παράσταση που ενσωματώνει τις πληροφορίες διαγραμμάτων κλάσεων και ακολουθίας χρησιμοποιείται ως βάση για την παραγωγή περιπτώσεων δοκιμής. Αυτή η προσέγγιση υιοθετεί μια συμβολική προσέγγιση εκτέλεσης για να αντλήσει τους περιορισμούς εισαγωγής δοκιμής από την παρουσιασμένη γραφική παράσταση και λύνει αυτούς τους περιορισμούς.

Ομοίως με αυτές τις προσεγγίσεις, προτείνουμε τη χρήση σαφώς προσδιορισμένων σχέσεων μεταξύ των δοκιμών των τεχνουργημάτων. Διαφορετικά, εξωτερικεύουμε τις σαφώς προσδιορισμένες σχέσεις από πρότυπο σε μια ιεραρχία παραγωγής περιπτώσεων δοκιμής, η οποία επιτρέπει την υποστήριξη για την ακριβέστερη δοκιμή οπισθοδρόμησης, την βασισμένη σε πρότυπο ανάλυση κάλυψης και την αξιολόγηση αποτελέσματος.

2.5 Επίλογος

Γενικά, η εφαρμογή τεχνικών ιχνηλασιμότητας απαιτήσεων μέσα σε μια επιχείρηση διευκολύνει την επαναχρησιμοποίηση και τη συντηρησιμότητα του συστήματος. Εντούτοις, θα απαιτηθούν πρόσθετοι πόροι για να εφαρμοστούν οι διαδικασίες ιχνηλασιμότητας. Στη μελέτη μιας περίπτωσης διαπιστώθηκε ότι το κόστος ήταν περισσότερο από το διπλό κανονικό κόστος τεκμηρίωσης που σχετίζεται με την ανάπτυξη ενός συστήματος του ίδιου μεγέθους. Εντούτοις, αυτό είναι μια δαπάνη που γίνεται μια φορά μόνο και οι γενικές δαπάνες για να διατηρήσουν το σύστημα λογισμικού αναμένονται για να μειωθούν. Σχεδόν η άμεση επιστροφή παρατηρήθηκε σε μικρό χρονικό διάστημα που χρησιμοποιήθηκε το βελτιωμένο υλικό.

Όλες οι τεχνικές εφαρμογής ιχνηλασιμότητας απαιτούν την δημιουργία μιας λειτουργικής ομάδας συμμετεχόντων για να δημιουργήσουν και να διατηρήσουν τους συνδέσμους μεταξύ των απαιτήσεων, της πηγής και της κατανομής στα επόμενα προϊόντα εργασίας. Ο αναλυτής απαιτήσεων πρέπει να αρχίσει την ιχνηλασιμότητα απαιτήσεων και να τεκμηριώσει την αρχική επισήμανση των απαιτήσεων των προϊόντων στην πηγή τους. Δεδομένου ότι οι αρχιτέκτονες συστημάτων και λογισμικού δημιουργούν υψηλού επιπέδου σχέδιο, αυτοί οι επαγγελματίες προσθέτουν τις πληροφορίες τους στην τεκμηρίωση ιχνηλασιμότητας. Οι υπεύθυνοι για την ανάπτυξη που κάνουν τη χαμηλού επιπέδου δοκιμή σχεδίου, κώδικα και μονάδων και προσθέτουν τις πρόσθετες πληροφορίες ιχνηλασιμότητας για τα στοιχεία που δημιουργούν, όπως και ολοκληρώνουν, συστήματα και ελεγκτές αποδοχής. Για τα μικρά προγράμματα, μερικοί από αυτούς τους ρόλους μπορούν να μην υπάρξουν ή μπορούν να γίνουν από τον επαγγελματία δειγμάτων, το οποίο περιορίζει τον αριθμό διαφορετικών ανθρώπων που εργάζονται με τις πληροφορίες ιχνηλασιμότητας. Για τα μεγαλύτερα προγράμματα, όπου οι πληροφορίες ιχνηλασιμότητας προέρχονται από πολλούς διαφορετικούς επαγγελματίες, μπορεί να είναι απαραίτητο να υπάρξει κάποιος που συντονίζει, τεκμηριώνει και εξασφαλίζει περιοδικούς λογιστικούς ελέγχους των πληροφοριών ιχνηλασιμότητας από όλες τις διάφορες πηγές του για να επιτύχει την πληρότητα και τη συνέπεια.

3. Πλαίσιο για την ιχνηλασιμότητα απαιτήσεων

3.1 Ένα μεταμοντέλο για την ιχνηλασιμότητα απαιτήσεων

Πριν την αναφορά του μεταμοντέλου για τις απαιτήσεις ιχνηλασιμότητας, συνοψίζονται οι ανάγκες πληροφοριών για τη διαχείριση απαιτήσεων. Δηλαδή τα είδη πληροφοριών που συνδέονται με την ιχνηλασιμότητα απαιτήσεων και τις πιθανές χρήσεις τους [Dömges and Pohl, 1998]:

1. Σύνδεσμοι ιχνηλασιμότητας μεταξύ των διαφορετικών τύπων προδιαγραφών εξασφαλίζουν ότι οι λειτουργίες του συστήματος καλύπτουν τις προσδοκίες των συμμέτοχων, ότι δεν γίνεται εφαρμογή περιττών λειτουργιών, και εκτελείται η ανάλυση αντίκτυπου όταν οι απαιτήσεις αλλάζουν.

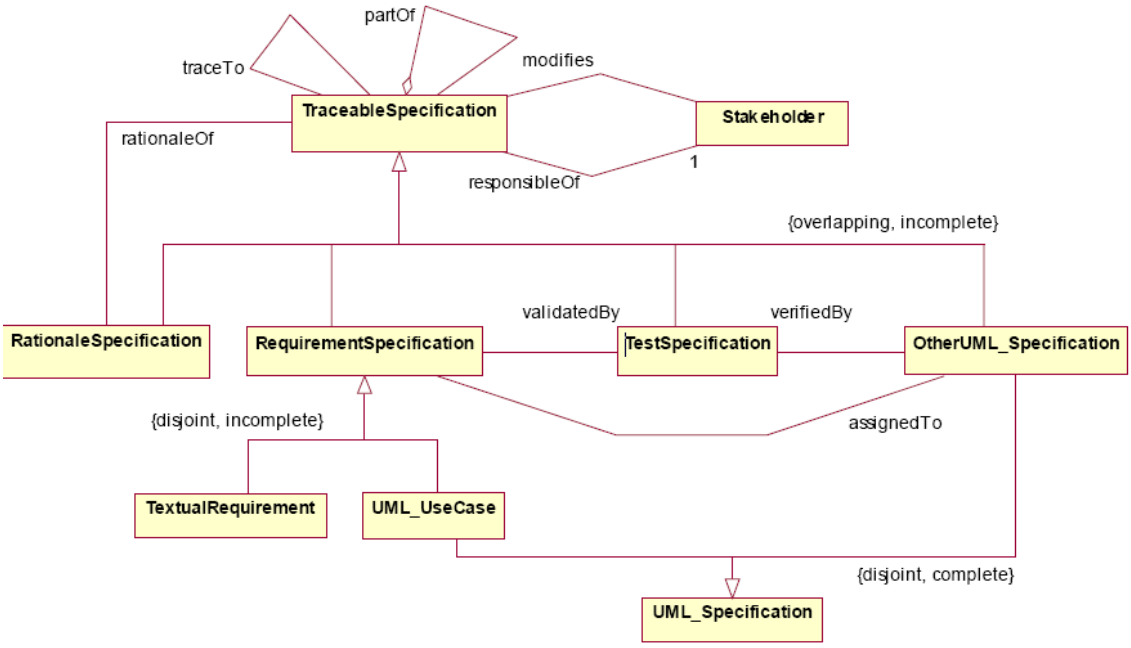
2. Δομές συνεισφοράς δηλαδή οι σύνδεσμοι μεταξύ των συμμέτοχων και των προδιαγραφών που βοηθούν στη βελτίωση της επικοινωνίας και της συνεργασίας μεταξύ των συμμέτοχων, και εγγυώνται ότι η συνεισφορά κάθε ατόμου εξετάζεται και καταχωρείται.

3. Λογική εξήγηση που συνδέεται με τις προδιαγραφές, συμπεριλαμβάνει εναλλακτικές λύσεις, αποφάσεις, υποθέσεις, κ.λπ. που συμβάλλουν στη βελτίωση της κατανόησης και της αποδοχής του συστήματος από τους συμμέτοχους, και για να βελτιώσει τη διοίκηση αλλαγών αποφεύγοντας να μελετήσει πάλι τις εκτιμήσεις που αποκλείστηκαν ήδη. Αυτό είναι δυνατό αν γίνουν προσιτές οι λύσεις, τα θεμελιώδη στοιχεία τους, και οι εναλλακτικές λύσεις που έχουν αποκλειστεί.

Στο σχήμα 3.1, με τη βοήθεια ενός διαγράμματος κλάσης, παρουσιάζεται ένα μεταμοντέλο για την ιχνηλασιμότητα των απαιτήσεων. Οι κλάσεις αντιπροσωπεύουν τους τύπους οντοτήτων και οι ενώσεις αντιπροσωπεύουν τους τύπους συνδέσμων ιχνηλασιμότητας. Χρησιμοποιούνται ονόματα που είναι σχετικά με τις ενώσεις για να γίνουν περισσότερο ευανάγνωστοι οι τύποι συνδέσμων ιχνηλασιμότητας.

Γενικά, ενδιαφερόμαστε για δύο τύπους οντοτήτων: την TraceableSpecification και την Stakeholder. Η κλάση Stakeholder είναι αρμόδια της δημιουργίας και των τροποποιήσεων των προδιαγραφών. Η TraceableSpecification είναι μια προδιαγραφή λογισμικού σε ένα συγκεκριμένο επίπεδο, όπου αυτό μπορεί να είναι ένα έγγραφο, ένα πρότυπο, ένα διάγραμμα, ένα τμήμα σε ένα έγγραφο, ένα κείμενο που διευκρινίζει μια μη λειτουργική απαίτηση, μια περίπτωση χρήσης, μια κλάση, μια ιδιότητα, κ.λπ. Το συγκεκριμένο επίπεδο για την TraceableSpecification καθορίζεται σύμφωνα με τον ρόλο που θα δοθεί στην έννοια partOf.

Ο τύπος οντότητας TraceableSpecification είναι μια γενίκευση της RationaleSpecification, της RequirementSpecification, της TestSpecification, και της OtherUML_Specification. Το TraceableSpecification μπορεί να ανήκει σε περισσότερες της μιας τέτοιων υποκατηγοριών, για παράδειγμα, όταν ένα έγγραφο περιλαμβάνει διάφορους τύπους προδιαγραφών (με πολλαπλή χρήση της partOf). Η RequirementSpecification είναι μια απαίτηση ή ομάδα απαιτήσεων. Οι απαιτήσεις, σύμφωνα με το πώς εκφράζονται, μπορεί να ταξινομηθούν ως TextualRequirements (απαιτήσεις που εκφράζονται χρησιμοποιώντας ένα κομμάτι κειμένου) ή UML_UseCase (το αντίστοιχο UML πρότυπο για την αντιπροσώπευση μιας λειτουργικής απαίτησης). Η RationaleSpecification καθιερώνει, για παράδειγμα: θεμελιώδη στοιχεία, εναλλακτικές λύσεις ή υποθέσεις που συνδέονται με την TraceableSpecification. Τέλος, η TestSpecification καθορίζει μια δοκιμή για την επικύρωση μιας απαίτησης ή την επαλήθευση ενός στοιχείου UML προτύπου (για παράδειγμα: ελέγχοντας ένα αρχείο πηγαίου κώδικα που είναι η εφαρμογή μιας κλάσης ή ενός συστατικού). Γενικεύσεις όπως ποιανού γονικές κλάσεις είναι η TraceableSpecification και η RequirementSpecification ορίζονται ως «ελλιπής» επιτρέποντας τον καθορισμό άλλων τύπων προδιαγραφών που μπορούν να είναι ενδιαφέρουσες για την ιχνηλασιμότητα. Για παράδειγμα, μερικές προδιαγραφές που δεν είναι κειμενικές ούτε γλώσσας UML είναι: το βίντεο, οι εικόνες, η φωνή, κ.λπ. Αυτές οι προδιαγραφές αντιστοιχούν συνήθως στα θεμελιώδη στοιχεία, και είναι χρήσιμες κατά τη διάρκεια της αναθεώρησης και της αξιολόγησης από τα πρότυπα ανάλυσης και σχεδίου [Haumer et al., 2000]. Εντούτοις, θα μπορούσαν επίσης να είναι ένα ιδιαίτερο μέσο για την RequirementSpecification, για παράδειγμα, καταχώρηση σε ένα βίντεο.



Σχήμα 3.1: Μεταμοντέλο για την ιχνηλασιμότητα απαιτήσεων.

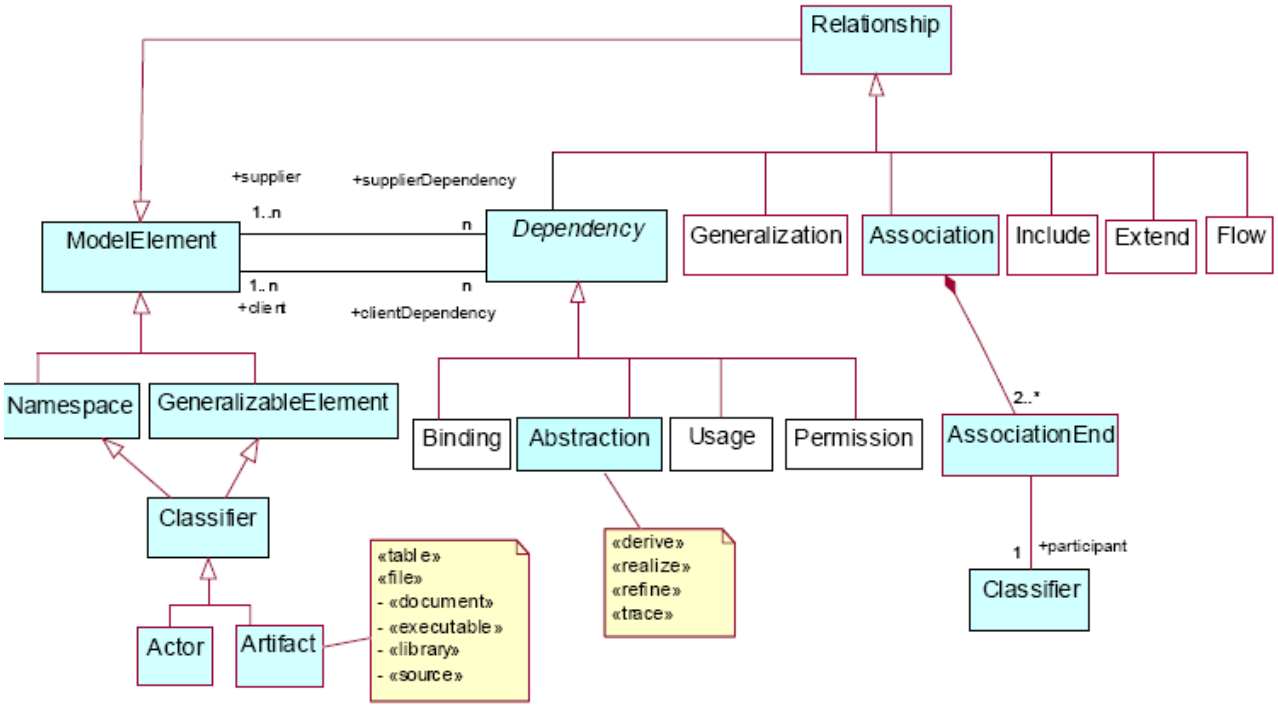
Ο γενικότερος τύπος συνδέσμου ιχνηλασιμότητας αντιπροσωπεύεται από την ένωση με το όνομα traceTo ο όποιος επιτρέπει την τοποθέτηση συνδέσμων ιχνηλασιμότητας μεταξύ οποιοδήποτε TraceableSpecification. Οι υπόλοιποι τύποι συνδέσμων ιχνηλασιμότητας (modifies, responsibleOf, rationaleOf, validatedBy, verifiedBy και assignedTo) είναι πιο συγκεκριμένοι. Ο τύπος συνδέσμου που ονομάζεται modifies ιδρύει μια σχέση μεταξύ της Stakeholder και της TraceableSpecification που τις επιτρέπει να τροποποιούν. Με έναν παρόμοιο τρόπο, ο responsibleOf καθορίζει την Stakeholder ως αρμόδια του καθορισμού και της συντήρησης της TraceableSpecification. Ο τύπος συνδέσμου που ονομάζεται validatedBy αφορά την RequimementsSpecification αντιστοιχίζεται με την TestSpecification και την επικυρώνει. Ο τύπος συνδέσμου verifiedBy καθορίζει το TestSpecifications που ελέγχει μια προδιαγραφή UML. Τέλος, ο τύπος συνδέσμου assignedTo καθορίζει τα στοιχεία UML προτύπων που πραγματοποιούν ορισμένες απαιτήσεις, για παράδειγμα, οι κλάσεις που πραγματοποιούν μια περίπτωση χρήσης.

Το μεταμοντέλο που παρουσιάζεται στο σχήμα 3.1 καλύπτει τις τέσσερις αρχές των πληροφοριών ιχνηλασιμότητας που περιλαμβάνονται στις εργασίες των Ramesh και Jarke [Ramesh and Jarke, 2001]: απαιτήσεις, λογική, κατανομή των απαιτήσεων στα μοντέλα και στα στοιχεία εφαρμογής, και τέλος, δοκιμή. Επιπλέον, το μεταμοντέλο, ενσωματώνει πτυχές προ-ιχνηλασιμότητας και μετα-ιχνηλασιμότητας [Jarke, 1998], [Pohl, 1996]. Η προ-ιχνηλασιμότητα επιτρέπει τη διαδρομή από την προέλευση των απαιτήσεων μέχρι τη σαφή προδιαγραφή τους στο έγγραφο απαιτήσεων προδιαγραφών λογισμικού (Software Requirements Specification - SRS) ή αντίστροφα. Η μετά-ιχνηλασιμότητα επιτρέπει τη διαδρομή από το SRS στο μεταγενέστερο λογισμικό και στις προδιαγραφές δοκιμής ή αντίστροφα. Και στα δύο είδη ιχνηλασιμότητας το μεταμοντέλο μας παρέχει τους τύπους συνδέσμων responsibleOf και modifies για να καθορίσουν την Stakeholder, δηλαδή τους μετόχους που συμμετέχουν. Για την προ-ιχνηλασιμότητα ο τύπος συνδέσμου traceTo είναι διαθέσιμος μεταξύ απαιτήσεων που εκφράζονται σε διαφορετικά επίπεδα αφαίρεσης και ο rationaleOf για τη λογική που συσχετίζονται οι προδιαγραφές

απαιτήσεων. Η μετα-ιχνηλασιμότητα υποστηρίζεται από τους τύπους συνδέσμων traceTo, validatedBy, verifiedBy, assignedTo και rationaleOf.

3.2 Το πλαίσιο UML για την ιχνηλασιμότητα του μεταμοντέλου

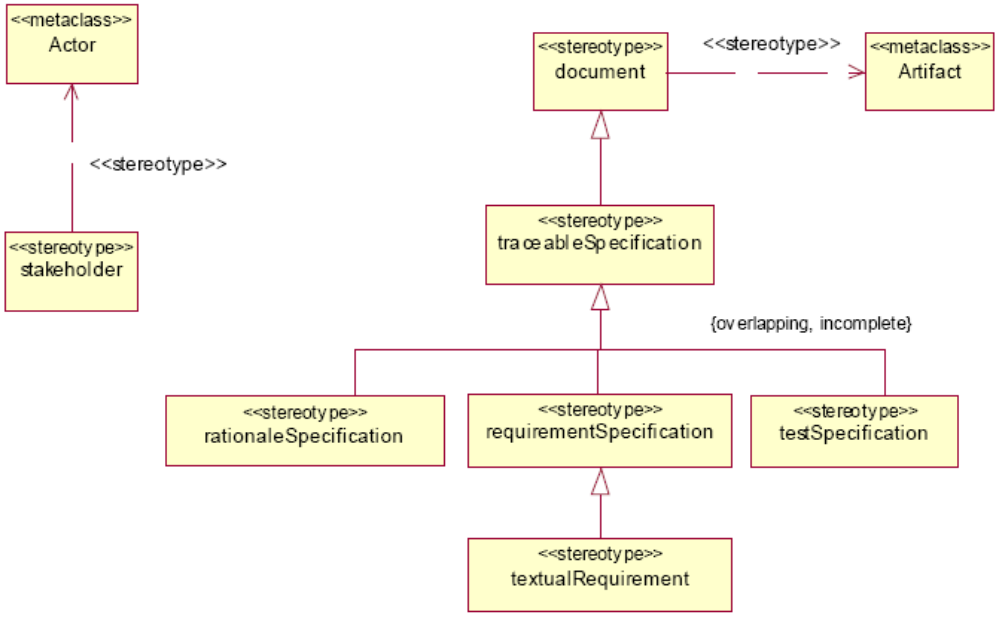
Για να γίνει πιο απλή και πρακτική η εφαρμογή του μεταμοντέλου θα ήταν καλό να ενσωματωθούν όλοι οι τύποι οντοτήτων και συνδέσμων σε ένα κοινό πλαίσιο. Λαμβάνοντας υπόψη ότι : (α) οι προδιαγραφές UML καθορίζονται ακριβέστερα και είναι πιο αποδεκτές από άλλες προδιαγραφές, (β) η UML παρέχει μηχανισμό επέκτασης (στερεότυπα, καθορισμένες τιμές και περιορισμοί) για να ενσωματωθούν νέοι τύποι προδιαγραφών, και (γ) οι προδιαγραφές UML υποστηρίζονται από τα περισσότερα εργαλεία τεχνολογίας λογισμικού με τη βοήθεια υπολογιστή (CASE tools). Άρα είναι εμφανές ότι θα ήταν αρμόζον να ενσωματωθούν όλοι τύποι προδιαγραφών του μεταμοντέλου στο πλαίσιο της UML. Κατά συνέπεια, για κάθε τύπο οντότητας και για κάθε τύπο συνδέσμου θα υπάρχει αντιστοιχία με ένα στοιχείο UML προτύπου. Για να γίνει αυτό, οι UML μετακλάσεις επιλέγονται ως βασικές κλάσεις για να καθιερώσουν τα νέα στερεότυπα. Όταν ο τύπος οντότητας ή τύπος συνδέσμου ταιριάζει σημασιολογικά με μια UML μετακλάση, αυτές οι μετακλάσεις χρησιμοποιούνται άμεσα χωρίς τον καθορισμό ενός νέου στερεοτύπου. Το αποτέλεσμα αυτής της ανάλυσης είναι ένα σχεδιάγραμμα UML (σχήμα 3.2) για αυτό το μεταμοντέλο ιχνηλασιμότητας. Παρακάτω δίνουμε τις λεπτομέρειες για το πώς εκτελείται μια τέτοια εφαρμογή.



Σχήμα 3.2: Πλαίσιο UML για το μεταμοντέλο ιχνηλασιμότητας.

Οντότητες ιχνηλασιμότητας στο πλαίσιο της UML. Για την οντότητα Stakeholder η επιλογή είναι απλή. Το πρότυπο στοιχείο Actor είναι η μετακλάση που χρησιμοποιείται ως

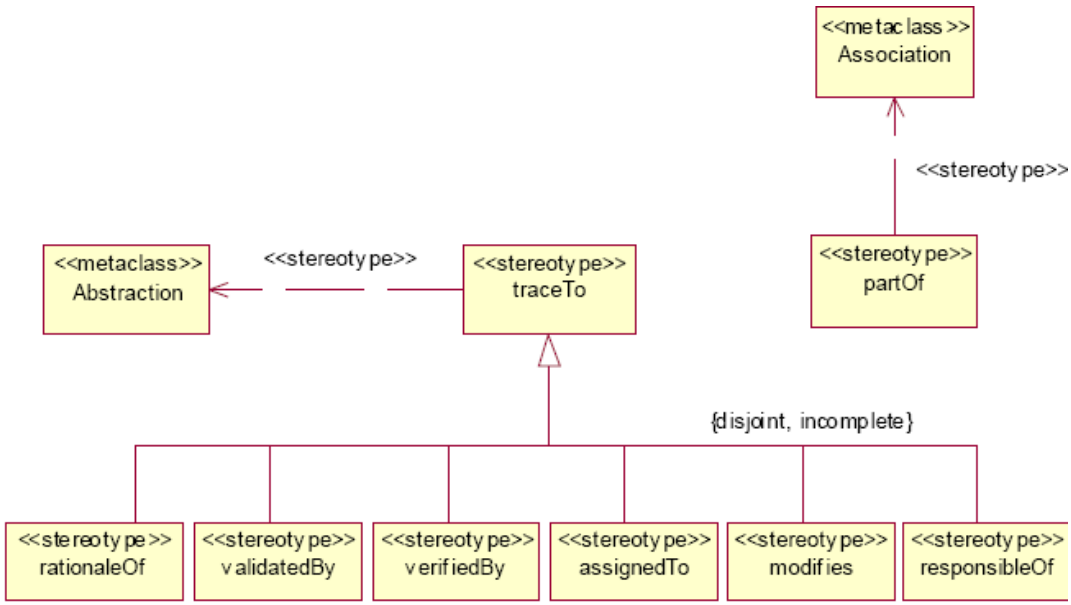
βασική κλάση η οποία καθορίζει το αντίστοιχο στερεότυπο. Άλλοι τύποι οντοτήτων πρέπει να έχουν τη δυνατότητα να επιτρέπουν τους συσχετισμούς προκειμένου να καθιερωθούν οι σχέσεις συσχέτισης μεταξύ τους. Σύμφωνα με αυτό, η επιλεγμένη μετακλάση πρέπει να είναι μεταξύ των στοιχείων UML που είναι κλάση παιδί του Classifier. Για τις οντότητες που δεν υποστηρίζουν τις τυποποιημένες προδιαγραφές UML, έχει επιλεγεί ο Classifier που ονομάζεται Artifact -τεχνούργημα- (έχει προστεθεί στην έκδοση UML 1.4). Το τεχνούργημα έχει μερικά προκαθορισμένα στερεότυπα, μεταξύ των οποίων το «document-έγγραφο», το οποίο είναι αυτό που θα χρησιμοποιήσουμε για να αντιπροσωπεύσουμε τα έγγραφα και τα τμήματα τους. Για τους τύπους οντοτήτων που ταιριάζουν άμεσα με τα στοιχεία του UML προτύπου (UML_UseCase και OtherUML_Specification) θα χρησιμοποιηθεί το ίδιο στοιχείο UML προτύπου. Για να ομαδοποιηθούν και να οργανωθούν τα τεχνουργήματα UML και η Stakeholder χρησιμοποιούνται κάποια πρότυπα της UML, δηλαδή Packages-πακέτα και προαιρετικά θα προσθέτονται τα προκαθορισμένα στερεότυπα «model-πρότυπο» ή «subsystem-υποσύστημα», ανάλογα με το εάν καθορίζουμε ένα πρότυπο ενός συστήματος/ υποσυστήματος ή διαιρούμε το σύστημα σε υποσυστήματα.



Σχήμα 3.3: Στερεότυπα για την Stakeholder και τις κειμενικές προδιαγραφές.

Σύνδεσμοι ιχνηλασιμότητας στο πλαίσιο UML. Οι τύποι συνδέσμων αναπαριστούνται ως στοιχεία UML προτύπου από τον τύπο της αφαίρεσης (Abstraction), εκτός από τη σχέση partOf, η οποία αντιπροσωπεύεται από τη συσχέτιση ή τη σύνθεση μεταξύ των προδιαγραφών χρησιμοποιώντας την μετακλάση Association σαν βασική κλάση. Αν και οι διαφορετικοί τύποι συνδέσμων διαμορφώνονται από τις διαφορετικές ενώσεις στο μεταμοντέλο ιχνηλασιμότητας, δεν είναι ανεξάρτητοι, στην πραγματικότητα, ο τύπος συνδέσμου traceTo είναι μια γενίκευση όλων των άλλων τύπων συνδέσμων. Ο τύπος συνδέσμου traceTo θα συμπίπτει με το στερεότυπο «trace-ίχνος», που προκαθορίζεται μέσα στη UML. Στη UML μια εξάρτηση ιχνών δείχνει ένα ιστορικό ή μία διαδικαστική σχέση μεταξύ δύο στοιχείων που αντιπροσωπεύουν την ίδια έννοια χωρίς διευκρινιστικούς κανόνες μεταξύ τους [OMG, 2002]. Με εξαίρεση τον τύπο συνδέσμου

traceTo, οι άλλοι τύποι συνδέσμων θα είναι παιδιά-στερεότυπα του προκαθορισμένου στερεότυπου «ίχνος».



Σχήμα 3.4: Στερεότυπα για τους συνδέσμους ιχνηλασιμότητας.

Στο σχήμα 3.2 παρουσιάζεται το πλαίσιο της UML για τις έννοιες που εξετάστηκαν. Σύμφωνα με προηγούμενες εξηγήσεις, το σχήμα 3.3 και το σχήμα 3.4 παρουσιάζουν την UML αναπαράσταση για τους τύπους οντοτήτων και τους τύπους συνδέσμων που συμπεριλαμβάνονται στο μεταμοντέλο ιχνηλασιμότητας. Αυτή η αναπαράσταση αποτελεί ένα ουσιαστικό σχεδιάγραμμα UML για τις απαιτήσεις ιχνηλασιμότητας.

3.3 Διαμορφώνοντας την ιχνηλασιμότητα

Στην ιχνηλασιμότητα απαιτήσεων προσδιορίζονται δύο δραστηριότητες:

(α) η διαμόρφωση της ιχνηλασιμότητας σύμφωνα με τις ανάγκες του προγράμματος, και

(β) η διευκρίνιση και εκμετάλλευση των πληροφοριών ιχνηλασιμότητας κατά τη διάρκεια της ανάπτυξης και συντήρησης λογισμικού.

Θα εστιάσουμε στη διαμόρφωση δραστηριοτήτων που εφαρμόζει το συγκεκριμένο UML σχεδιάγραμμα για την ιχνηλασιμότητα. Το σχεδιάγραμμα θα ενεργήσει ως πλαίσιο για να τοποθετηθούν οι τύποι των τεχνουργημάτων (artifacts) από δω και πέρα χρησιμοποιείται ο όρος «τεχνούργημα» υπό μια ευρύτερη έννοια, πέρα από τον ορισμό που παρέχεται στη UML, και με τον ίδιο τρόπο όπως κάνουν, οι περισσότερες διαδικασίες λογισμικού (για παράδειγμα RUP). Κατά συνέπεια, θα θεωρούμε ως τεχνουργήματα όλα τα έγγραφα, αρχεία και άλλα φυσικά στοιχεία που παράγονται ή που χρησιμοποιούνται κατά τη διάρκεια της διαδικασίας ανάπτυξης λογισμικού, επιπλέον, θα ονομάζεται τεχνούργημα οποιοδήποτε UML πρότυπο στοιχείο για την ιχνηλασιμότητα και οι τύποι συνδέσμων. Συνεπώς, οι σύνδεσμοι ιχνηλασιμότητας που καθιερώνονται κατά τη διάρκεια της ανάπτυξης ή συντήρησης λογισμικού ελέγχονται κατά τη

διάρκεια διαμόρφωσης της ιχνηλασιμότητας (για παράδειγμα, για μια συγκεκριμένη εφαρμογή, ορισμένοι τύποι συνδέσμων ισχύουν μόνο μεταξύ ορισμένων τύπων τεχνουργημάτων).

Η διαμόρφωση ιχνηλασιμότητας σε ένα πρόγραμμα περιλαμβάνει τις ακόλουθες διαδικασίες:

1. Επιλογή τύπων τεχνουργημάτων που ενδιαφέρουν την προοπτική ιχνηλασιμότητας. Αποτελούν ένα υποσύνολο όλων των τύπων τεχνουργημάτων που χρησιμοποιούνται από την εφαρμογή. Κάθε επιλεγμένο τεχνούργημα θα έχει ένα αντίστοιχο στερεότυπο ως κλάση-παιδί ενός στερεότυπου που καθορίζεται στο σχεδιάγραμμα ιχνηλασιμότητας (εκτός από εκείνους τους τύπους τεχνουργημάτων που είναι τυποποιημένα στοιχεία UML πρότυπου). Όταν γίνει αυτό, θα γίνει επέκταση του σχεδιαγράμματος με νέα στερεότυπα για τις οντότητες (αυτό περιλαμβάνει τη δυνατότητα καθορισμού νέων στερεοτύπων ως ειδικεύσεις από το συμμέτοχο).

2. Καθορισμός σχέσεων συσχέτισης μεταξύ των τεχνουργημάτων. Αυτός ο στόχος μπορεί να μην είναι απαραίτητος για όλους τους τύπους τεχνουργημάτων εάν τέτοιες σχέσεις προκαθορίζονται και συμπεριλαμβάνονται στην περιγραφή των τύπων τεχνουργημάτων.

3. Δημιουργία τύπων συνδέσμων ιχνηλασιμότητας που συσχετίζονται με την εφαρμογή. Οι τύποι συνδέσμων δημιουργούνται μεταξύ ζευγαριών τύπων τεχνουργημάτων που επιλέγονται από τη διαδικασία 1. Σε αυτήν την περίπτωση μπορεί επίσης να είναι απαραίτητο να επεκταθεί το σχεδιάγραμμα ιχνηλασιμότητας συμπεριλαμβάνοντας νέους τύπους συνδέσμων ως ειδικεύσεις στερεοτύπων.

4. Καθορισμός κριτηρίων για να αντληθούν οι σύνδεσμοι ιχνηλασιμότητας και ποιοι τύποι συνδέσμων (που καθιερώνονται στη διαδικασία 3) θα χρησιμοποιήσουν αυτά τα κριτήρια.

3.4 Διαμορφώνοντας την ιχνηλασιμότητα σε μια εφαρμογή RUP

Το μεταμοντέλο που παρουσιάζεται και το αντίστοιχο σχεδιάγραμμα είναι ανεξάρτητα από τη διαδικασία ανάπτυξης λογισμικού, το κοινό είναι ότι η διαδικασία βασίζεται στη UML. Εντούτοις, για να επεξηγηθεί η εφαρμογή της προσέγγισής επιλέγεται η RUP ως διαδικασία, κυρίως επειδή προσφέρει αρκετές λεπτομέρειες και ποικιλία σχετικά με τα διαθέσιμα τεχνουργήματα. Η RUP είναι μια μεθοδολογία αντικειμενοστραφούς ανάλυσης και σχεδίασης (Rational Unified Process) [Χατζηγεωργίου[1]], 2005], [Jacobson et al., 1999]. Η RUP παρέχει πρότυπα για Word και για HTML για τεχνουργήματα όπως έγγραφα, και χρησιμοποιεί τη UML για τη διαμόρφωση λογισμικού. Έπειτα επεξηγούνται οι διαδικασίες για τις απαιτήσεις ιχνηλασιμότητας σε ένα μικρό πρόγραμμα RUP.

Υποεργασία 1. Επιλογή τύπων τεχνουργημάτων που είναι ανιχνεύσιμες προδιαγραφές. Σε αυτό το παράδειγμα χρησιμοποιούνται τα τεχνουργήματα της RUP που παρουσιάζονται στον παρακάτω πίνακα:

RUP Type of Artifact	Parent Class Stereotype
Vision	«traceableSpecification»
Software Feature	«textualRequirement»
Supplementary Spec.	«traceableSpecification»
Non-Functional Req.	«textualRequirement»
Assumption	«rationaleSpecification»
Use Case Specification	«traceableSpecification»
Use Case Precondition	«traceableSpecification»
Use Case Model	«model»
Use Case	
Analysis & Design Model	«model»
Class	
Implementation Model	«model»
Component	
Data Model	«model»
Test Case	«testSpecification»

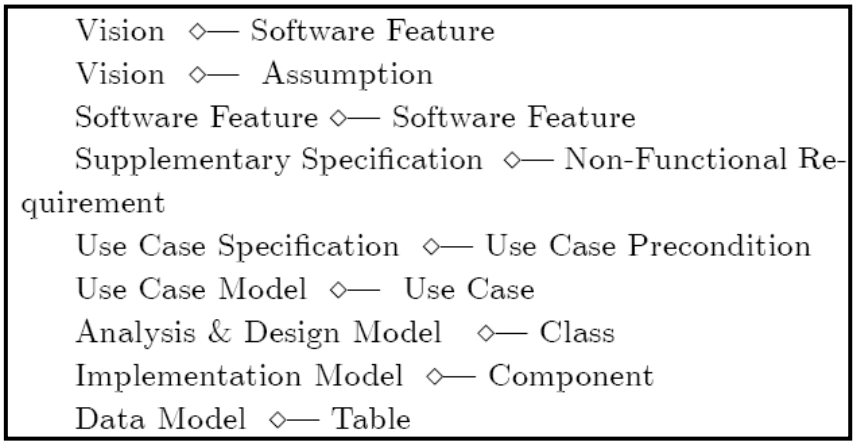
Πίνακας 3.1 (α): Τεχνουργήματα της RUP.

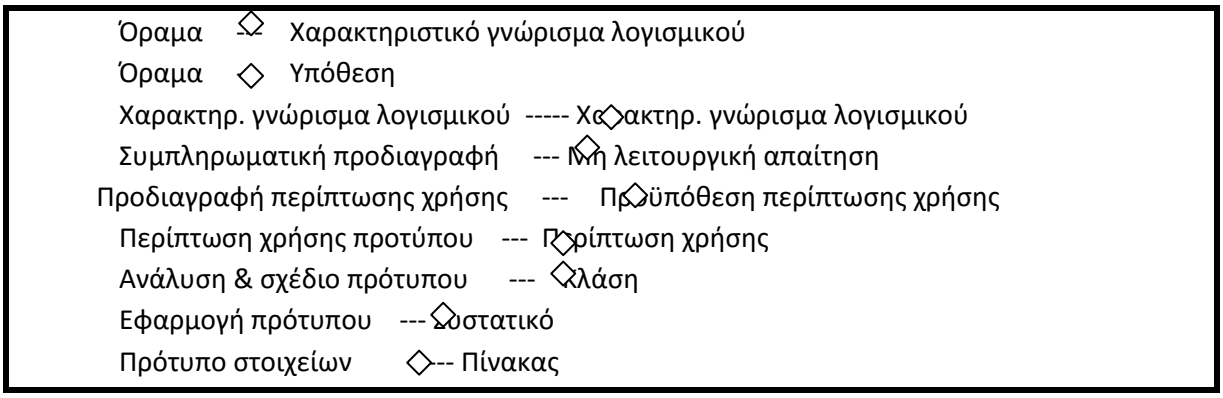
Τύπος RUP τεχνουργήματος	**Στερεότυπο γονικής κλάσης**
Όραμα	«traceableSpecification»
Χαρακτηριστικό γνώρισμα λογισμικού	«textualRequirement»
Συμπληρωματική προδιαγραφή	«traceableSpecification»
Μη λειτουργική απαίτηση	«textualRequirement»
Υπόθεση	«rationaleSpecification»
Περίπτωση χρήσης προδιαγραφής	«traceableSpecification»
Περίπτωση χρήσης προϋπόθεσης	«traceableSpecification»
Περίπτωση χρήσης πρότυπου	«model»
Περίπτωση χρήσης	
Πρότυπο ανάλυσης & σχεδίου	«model»
Κλάση	
Πρότυπο εφαρμογής	«model»
Συστατικό	
Πρότυπο στοιχείων	«model»
Περίπτωση δοκιμής	«testSpecification»

Πίνακας 3.1 (β): Τεχνουργήματα της RUP.

Όταν, στον παραπάνω πίνακα, δεν υπάρχει ένα στερεότυπο γονικής κλάσης αυτό είναι επειδή χρησιμοποιεί άμεσα το αντίστοιχο στοιχείο UML προτύπου. Όπως αναφέρεται παραπάνω, το στερεότυπο «model» είναι στο πακέτο της UML με ένα τέτοιο στερεότυπο.

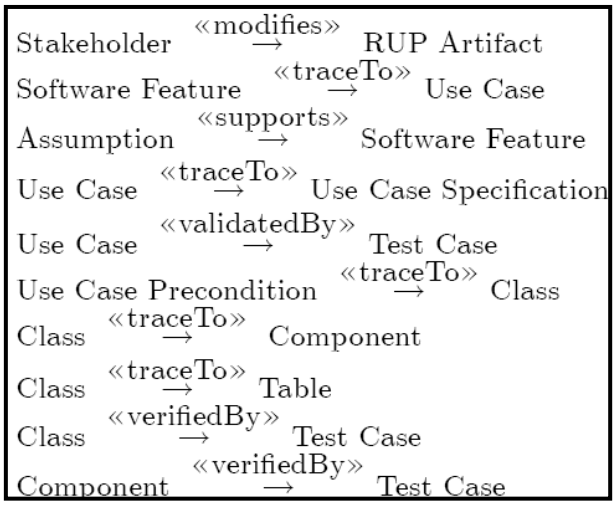
Υποεργασία 2. Καθορίζει τις σχέσεις συσχέτισης μεταξύ τύπων τεχνουργημάτων. Για τα επιλεγμένα τεχνουργήματα οι σχέσεις συσχέτισης (συσσωμάτωσης) [Χατζηγεωργίου[2]], 2005]είναι:





Πίνακες 3.2: Υποεργασία 2.

Υποεργασία 3. Καθορισμός των τύπων συνδέσμων ιχνηλασιμότητας τα οποία έχουν άμεση σχέση με την εφαρμογή. Στο παράδειγμά αυτοί είναι:



Συμμέτοχος	→ «responsibleOf» →	Τεχνούργημα RUP
Συμμέτοχος	→ «modifies» →	Τεχνούργημα RUP
Χαρ. γνώρισμα λογισμικού	→ «traceTo» →	Περίπτωση χρήσης
Υπόθεση	→ «supports» →	Χαρακτηρ. γνώρισμα λογισμικού
Περίπτωση χρήσης	→ «traceTo» →	Προδιαγραφή περίπτωσης χρήσης
Περίπτωση χρήσης	→ «validatedBy» →	Περίπτωση δοκιμής
Προϋπόθεση περ. χρήσης	→ «traceTo» →	Κλάση
Κλάση	→ «traceTo» →	Συστατικό
Κλάση	→ «traceTo» →	Πίνακας
Κλάση	→ «verifiedBy» →	Περίπτωση δοκιμής
Συστατικό	→ «verifiedBy» →	Περίπτωση δοκιμής

Πίνακες 3.3: Υποεργασία 3.

Χρησιμοποιούνται τα τεχνουργήματα της RUP (RUP Artifacts) για να αναφερθούμε σε οποιοδήποτε RUP τεχνούργημα που επιλέγεται στην υποεργασία 1. Το στερεότυπο «supports» έχει εισαχθεί ως νέα στερεότυπη ειδίκευση του στερεότυπου «rationaleOf».

Υποεργασία 4 Για να καταγραφεί και η υπονοούμενη ιχνηλασιμότητα θα χρησιμοποιείται η μεταβατικότητα, σχέσεις συσχέτισης και ακριβές όνομα που να ταιριάζει μεταξύ των τύπων των τεχνουργημάτων. Σε αυτήν την τελευταία περίπτωση το κριτήριο της ισότητας ονόματος θα εφαρμοστεί μόνο στους ακόλουθους τύπους συνδέσμων:

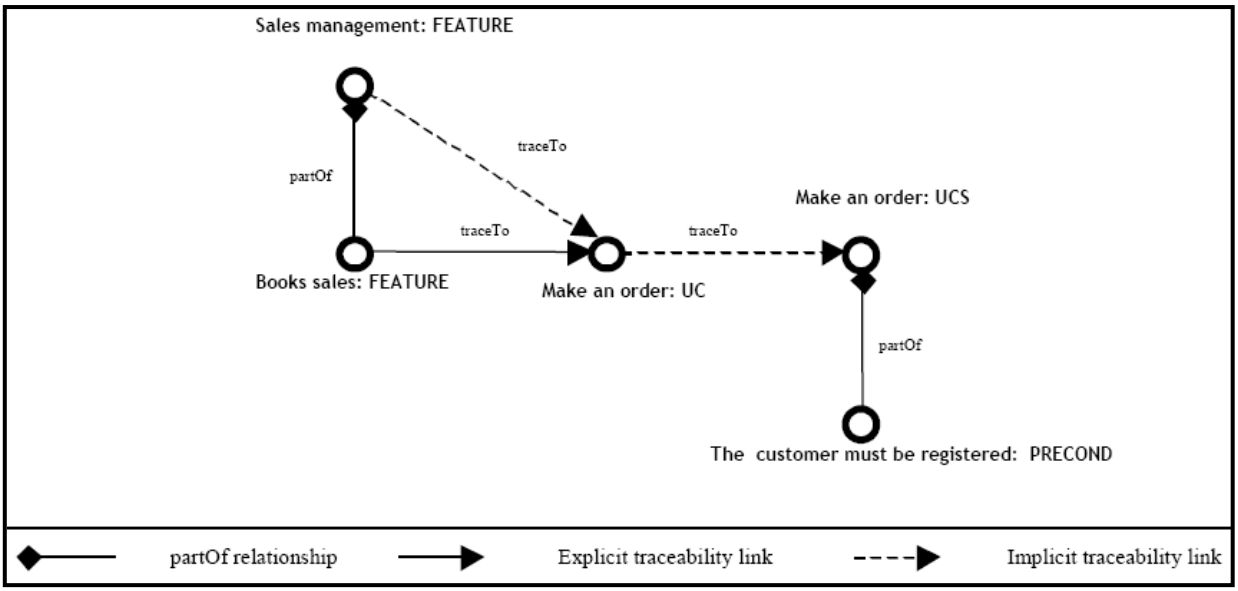
Use Case	$\xrightarrow{\text{«traceTo»}}$	Use Case Specification
Use Case	$\xrightarrow{\text{«validatedBy»}}$	Test Case
Class	$\xrightarrow{\text{«traceTo»}}$	Table
Class	$\xrightarrow{\text{«verifiedBy»}}$	Test Case
Component	$\xrightarrow{\text{«verifiedBy»}}$	Test Case

Περίπτωση χρήσης	→ «traceTo» →	Προδιαγραφή περίπτωσης χρήσης
Περίπτωση χρήσης	→ «validatedBy» →	Περίπτωση δοκιμής
Κλάση	→ «traceTo» →	Πίνακας
Κλάση	→ «verifiedBy» →	Περίπτωση δοκιμής
Συστατικό	→ «verifiedBy» →	Περίπτωση δοκιμής

Πίνακες 3.4: Υποεργασία 4.

Τελικά, σύμφωνα με την διαμόρφωση ιχνηλασιμότητας, θα μπορούσαμε να έχουμε τη γραφική παράσταση ιχνηλασιμότητας που παρουσιάζεται στο σχήμα 3.5. FEATURE (χαρακτηριστικό γνώρισμα) είναι ο τύπος χαρακτηριστικού γνωρίσματος λογισμικού τεχνουργημάτων, UC είναι η περίπτωση χρήσης τύπων, UCS είναι ένας τύπος προδιαγραφής

περίπτωσης χρήσης, και το PRECOND αντιστοιχεί στη χρήση τύπων Προϋπόθεση περίπτωσης. Ο σύνδεσμος : Sales management : FEATURE → traceTo → Make an order: UC, παράγεται από τη σχέση συσχέτισης μεταξύ του λογισμικού χαρακτηριστικού γνωρίσματος Sales management- διαχείριση πωλήσεων και Books sales- πωλήσεις βιβλίων. Ο σύνδεσμος: Make an order : UC → traceT o → Make an order: UCS, παράγεται από τα ταιριαστά κριτήρια ονόματος. Άλλοι υπονοούμενοι σύνδεσμοι μεταβατικότητας ή συσχέτισης αγνοούνται επειδή δεν ενδιαφέρουν ως τύποι συνδέσμων (καθορίζονται στην υποεργασία 3).



Σχήμα 3.5: Ένα παράδειγμα γραφικής παράστασης ιχνηλασιμότητας.

3.5 Επίλογος

Σε αυτήν την ενότητα παρουσιάστηκε ένα μεταμοντέλο ιχνηλασιμότητας ενσωματώνοντας κειμενικές προδιαγραφές (για τις απαιτήσεις, τον ορθολογισμό και τις δοκιμές) με την τυποποιημένη UML προδιαγραφή, χρησιμοποιώντας το ίδιο το UML πλαίσιο. Κατά συνέπεια, από την άποψη της ιχνηλασιμότητας απαιτήσεων, το μεταμοντέλο μας προσφέρει ένα κεντρικό πλαίσιο για τους τύπους οντοτήτων και τους τύπους των συνδέσμων ιχνηλασιμότητας που μπορούν προσαρμοστούν σε μια συγκεκριμένη εφαρμογή χρησιμοποιώντας τους μηχανισμούς επέκτασης που παρέχονται από UML. Η ιχνηλασιμότητα μεταμοντέλου έχει ενσωματωθεί σε εφαρμογή UML και επιτρέπει μια ευκολότερη εφαρμογή με την υποστήριξη εργαλείων τεχνολογίας λογισμικού με τη βοήθεια υπολογιστή (CASE tools) της UML.

Επιπλέον, σκιαγραφείται μια διαδικασία διαμόρφωσης για την ιχνηλασιμότητα απαιτήσεων βασισμένη στην εφαρμογή της UML για την ιχνηλασιμότητα απαιτήσεων. Η προσέγγισή, συμπεριλαμβανομένου του μεταμοντέλου, η αντιστοιχία της UML εφαρμογής και η διαδικασία διαμόρφωσης έχουν μόνο την υπόθεση της χρησιμοποίησης μιας βασισμένης σε UML διαδικασίας, αλλά αυτό είναι ανεξάρτητο από οποιαδήποτε ιδιαίτερη διαδικασία. Εντούτοις, για να επεξηγηθεί η προσέγγισή, παρουσιάστηκε ένα παράδειγμα χρησιμοποιώντας την RUP ως αναπτυξιακή διαδικασία.

Μια σημαντική πτυχή κατά τη διαμόρφωση της ιχνηλασιμότητας για ένα πρόγραμμα είναι να τοποθετήσει τις ιδιότητες της ιχνηλασιμότητας (και τις πιθανές τιμές τους) για κάθε έναν τύπο τεχνουργημάτων. Υποτίθεται ότι το εργαλείο διαχείρισης απαιτήσεων προσφέρει ένα σύνολο προκαθορισμένων ιδιοτήτων για κάθε τύπο τεχνουργήματος και ο χρήστης είναι ικανός να επιλέξει ή να καθορίσει νέους. Για παράδειγμα, στη RUP μερικές ιδιότητες για τα χαρακτηριστικά γνωρίσματα λογισμικού είναι: κατάσταση (προτεινόμενη, εγκεκριμένη ή που ενσωματώνεται), όφελος (κρίσιμο, σημαντικό ή χρήσιμο), κατ' εκτίμηση προσπάθεια, κίνδυνος και σταθερότητα (αυτές οι ιδιότητες έχουν συνήθως τις τιμές όπως υψηλή, μεσαία ή χαμηλή).

4. Σύνδεσμοι Ιχνηλασιμότητας

Οι σύνδεσμοι ιχνηλασιμότητας γίνονται ευρέως αποδεκτοί ως αποδοτικά μέσα για να υποστηριχθεί μια εξελικτική ανάπτυξη λογισμικού [Maeder et al., 2007]. Εντούτοις, η χρήση τους στην ανάλυση και στο σχεδιασμό είναι σπάταλη προσπάθεια και λάθος τάση λόγω έλλειψης μεθόδων και εργαλείων για τη δημιουργία τους, την αναπροσαρμογή και την επαλήθευση τους .

Σε αυτή την ενότητα αναλύεται και ταξινομείται η ενοποιημένη διαδικασία (UP) τεχνουργημάτων ώστε να καθιερωθεί ένα πρότυπο συνδέσμων ιχνηλασιμότητας για αυτή τη διαδικασία. Αυτό το πρότυπο καθορίζει όλες τις απαραίτητες συνδέσεις μεταξύ των τεχνουργημάτων. Επιπλέον, παρέχει μια βάση για (ημι) - αυτόματη καθιέρωση και επαλήθευση

των συνδέσμων στις ενοποιημένες μελέτες ανάπτυξης διαδικασίας. Επίσης καθορίζεται ένα πρώτο σύνολο κανόνων ως βήμα προς έναν αποδοτικό τρόπο διαχείρισης των συνδέσμων. Στην τρέχουσα ενότητα το σύνολο κανόνων επεκτείνεται για να θεσπίσει ολόκληρο πλαίσιο μεθόδων και κανόνων.

4.1 Σύνδεσμοι ιχνηλασιμότητας προσαρμοσμένοι στη UP μέθοδο

Οι σύνδεσμοι ιχνηλασιμότητας είναι απαραίτητοι για ολόκληρη την αναπτυξιακή διαδικασία από τις απαιτήσεις έως την εφαρμογή του συστήματος. Σήμερα, η ιχνηλασιμότητα περιορίζεται συνήθως στο να συσχετίζεται με τις απαιτήσεις και κακώς δεν χρησιμοποιείται στην πράξη. Οι σύνδεσμοι ιχνηλασιμότητας πρέπει να κρατηθούν σε σωστή και πλήρη τάξη και πρέπει να καθοριστούν σε σωστά καθορισμένο επίπεδο για να είναι χρήσιμοι. Αυτό απαιτεί ένα πολύ μεγάλο αριθμό των συνδέσμων που ρυθμίζονται και που διατηρούνται ακόμη και για τα μικρά συστήματα. Η συντήρηση πρέπει να πραγματοποιηθεί χωρίς τη βοήθεια τεχνικών μέσων και η διαχείριση απαιτεί πολύ μεγάλη προσπάθεια. Η προϋπόθεση για ένα αποτελεσματικό εργαλείο υποστήριξης είναι μια λεπτομερής ενσωμάτωση συνδέσμων ιχνηλασιμότητας στις μεθόδους ανάπτυξης.

Ο καθορισμός των δραστηριοτήτων, των σχέσεων και των τεχνουργημάτων των περισσότερων μεθόδων σχεδιασμού είναι πάρα πολύ ανακριβείς και ασαφείς για να καθορίσουν κανόνες για συνδέσμους ιχνηλασιμότητας βασισμένους σε αυτούς και για να υποστηρίξουν την έννοια της ιχνηλασιμότητας [Maeder et al., 2007].

Προκειμένου να επιτευχθούν σωστά αποτελέσματα έχει επιλεχτεί και πάλι η ενοποιημένη διαδικασία (UP) για τον καθορισμό προτύπων συνδέσμων ιχνηλασιμότητας. Για την UP, υπάρχουν λεπτομερείς περιγραφές της μεθοδολογίας σχεδιασμού της. Στην προηγούμενη ενότητα παρουσιάστηκε η εφαρμογή του Letelier [Letelier, 2002] για το δικό του μοντέλο UP, οι ορισμοί του δεν είναι αρκετά λεπτομερείς για να παράγουν κανόνες για συνδέσμους ιχνηλασιμότητας. Δεν υπάρχει καμία λεπτομερής περιγραφή για το πώς και μεταξύ ποιων τεχνουργημάτων οι σύνδεσμοι ιχνηλασιμότητας πρέπει να τοποθετηθούν, αν και η UP περιγράφει την ιχνηλασιμότητα ως ένα από τα χαρακτηριστικά της γνωρίσματα. Επιπλέον, ένας συντακτικός και σημασιολογικός καθορισμός των συνδέσμων ιχνηλασιμότητας είναι απαραίτητος.

Σαν συμβολή αυτής της ενότητας ταξινομούνται και αναλύονται τα UP τεχνουργήματα που συσχετίζονται με την ιχνηλασιμότητα. Με βάση αυτό, όλοι οι απαραίτητοι σύνδεσμοι μεταξύ των τεχνουργημάτων των δραστηριοτήτων UP της εφαρμοσμένης μηχανικής απαιτήσεων και του σχεδιασμού καθορίζονται με την παροχή ενός πρότυπου συνδέσμου ιχνηλασιμότητας.

Επιπλέον, ένας συντακτικός και σημασιολογικός ορισμός των συνδέσμων ιχνηλασιμότητας καθιερώνεται προσαρμοσμένος στη UP μέθοδο. Η ανάλυση της UP και η προσαρμογή της έννοιας ιχνηλασιμότητα εκτελείται κατά τη διάρκεια των πρακτικών μελετών ανάπτυξης. Σαν αποτέλεσμα αυτής της ενότητας, έχουν καθιερωθεί κανόνες για την επαλήθευση των συνδέσμων ιχνηλασιμότητας.

4.2 Τύποι συνδέσμων ιχνηλασιμότητας

Υπάρχει ένα ευρύ φάσμα από τους τύπους και τις έννοιες που αφορούν την ιχνηλασιμότητα, π.χ. η έννοια των εξαρτήσεων σε UML και την επέκτασή του σε SysML [Weilkiens, 2006]. Για λόγους κατηγοριοποίησης εισάγεται η έννοια ενός τύπου συνδέσμων ιχνηλασιμότητας. Ένας τύπος συνδέσμων ιχνηλασιμότητας ταξινομεί τη σχέση μεταξύ δύο συνδεόμενων στοιχείων ή/και τη δραστηριότητα ανάπτυξης, για την παραγωγή του στοιχείου προορισμού από το στοιχείο πηγής. Η χρησιμοποίηση των τύπων συνδέσμων ιχνηλασιμότητας, έχει στόχο να ελαχιστοποιηθεί ο αριθμός διαφορετικών κανόνων για τη καθιέρωση και τον έλεγχο των συνδέσμων. Δεν είναι απαραίτητο να βρεθούν κανόνες για κάθε πιθανό συνδυασμό στοιχείων πηγής και προορισμού, κανόνες βασισμένοι σε τύπους μπορούν να εφαρμοστούν αντ' αυτού. Καθιερώνονται τέσσερις τύποι συνδέσμων ιχνηλασιμότητας ως βασικοί [Maeder et al, 2007]:

• Εκλέπτυνση - Refinement (καθαρίζω- εκλεπτύνω- βελτιώνω) - σύμφωνα με το επίπεδο λεπτομέρειας των συνδεδεμένων αντικειμένων (π.χ. μεταξύ ενός αντικείμενου ανάλυσης και ενός αντικείμενου σχεδιασμού),

• Κατανόηση - Realization (κατανοώ- πραγματοποιώ) - το εξαρτώμενο αντικείμενο αντιπροσωπεύει ένα μέρος της λύσης, στο πρόβλημα που περιγράφεται με το ανεξάρτητο αντικείμενο (π.χ. μεταξύ μιας περίπτωση χρήσης και μιας κλάσης ανάλυσης),

• Επαλήθευση - Verification (ελέγχω) - της συμπεριφοράς και των ιδιοτήτων από την αναπτυγμένη λύση ή τα μέρη της (π.χ. μεταξύ μιας περίπτωσης χρήσης και μιας περίπτωσης δοκιμής) και

• Καθορισμός - Definition (καθορίζω) - των αντικειμένων (π.χ. μεταξύ ενός στοιχείου του γλωσσάριου και τη χρήση του σε ένα από τα πρότυπα).

4.3 Απεικόνιση συνδέσμων ιχνηλασιμότητας

Στη UML τα ίχνη ορίζονται σαν ένα ειδικό είδος εξαρτήσεων. Επομένως χρησιμοποιείται η ίδια γραφική απεικόνιση: ένα βέλος, ομοιοκατευθυνόμενο με το στερεότυπο «ίχνος». Για μια απλή εξάρτηση το βέλος κατευθύνεται από το εξαρτώμενο (προορισμός) στο ανεξάρτητο (πηγή) στοιχείο π.χ. ένα αντικείμενο ανάλυσης συνδέεται προς μια περίπτωση χρήσης. Η γραφική κατεύθυνση του συνδέσμου της ιχνηλασιμότητας δεν αποκλείει τη χρήση της και στις δύο κατευθύνσεις, προς τα εμπρός και προς τα πίσω.

4.4 Η ενοποιημένη διαδικασία UP

Οι διαδικασίες ανάπτυξης λογισμικού αποτελούνται από τις δραστηριότητες και τα τεχνουργήματα που οδηγούν από τις απαιτήσεις στην εφαρμογή του συστήματος. Ο χειρισμός των συνδέσμων ιχνηλασιμότητας μπορεί να είναι εργαλείο υποστήριξης στις περισσότερες πράξεις ενός υπεύθυνου για την ανάπτυξη που ασχολείται με τις δραστηριότητες μιας μεθόδου. Έτσι είναι δυνατόν να εφαρμόσει τους κανόνες ιχνηλασιμότητας σε αυτές τις δραστηριότητες.

Όσο καλύτερη και πιο λεπτομερής είναι η διαδικασία περιγραφής τόσο είναι ευκολότερος ο καθορισμός των κανόνων για τη δημιουργία και την ενημέρωση των συνδέσμων ιχνηλασιμότητας. Επομένως, η προσέγγιση για την ιχνηλασιμότητα που προτείνεται σε αυτή την ενότητα είναι προς με την βοήθεια της ενοποιημένης διαδικασίας, επειδή είναι συγκεκριμένη, ευρέως χρησιμοποιημένη και καλά περιγραμμένη [Maeder et al, 2007].

Στην UP διάφορες προηγούμενες μέθοδοι όπως η αντικειμενοστραφής (προσανατολισμένη στο αντικείμενο) τεχνολογία λογισμικού OOSE [Jacobson, 1992] ήταν βασισμένη στις καλύτερες πρακτικές και την εμπειρία. Η UP είναι διαθέσιμη εμπορικά και ως έκδοση ανοικτού κώδικα. Το UP πρότυπο διαδικασίας, οι δραστηριότητες των μεθόδων και η σύνθεση των τεχνουργημάτων περιγράφονται λεπτομερώς για το κάθε επίπεδο υποστήριξης. Η UP μπορεί να προσαρμοστεί και να συγκεκριμενοποιηθεί σε ιδιαίτερα προγράμματα και ανάγκες επιχειρήσεων. Η UP είναι μια επαυξητική και επαναληπτική διαδικασία βασισμένη στην περίπτωση χρήσης και στην αρχιτεκτονική ανάπτυξης του λογισμικού. Η επαυξητική, επαναληπτική προσέγγιση μπορεί να δει το σχέδιο σε δύο διαστάσεις όπως περιγράφεται στο [Arlow and Neustadt, 2005].

4.4.1. Δραστηριότητες ανάπτυξης και σχέσεις μεταξύ των στοιχείων του προτύπου

Σε αυτή την υποενότητα προτείνεται ένα πρότυπο χρήσιμων συνδέσμων ιχνηλασιμότητας για τη UP. Καταρχάς κάθε μια από τις UP δραστηριότητες ανάπτυξης εξηγούνται εν συντομία, και είναι τοποθετημένοι οι απαραίτητοι σύνδεσμοι ιχνηλασιμότητας μεταξύ των περιληφθέντων τεχνουργημάτων. Για να δοθεί μια επισκόπηση, η σημαντικότερη ιχνηλασιμότητα συσχετίζεται με τις δραστηριότητες ανάπτυξης της UP, που παρουσιάζονται στο σχήμα 4.1 από ένα διάγραμμα δραστηριότητας. Πρέπει να επισημάνουμε ότι χρησιμοποιείται μια διαδοχική αντιπροσώπευση των δραστηριοτήτων για καλύτερη απεικόνιση. Εντούτοις, στην πράξη οι δραστηριότητες πραγματοποιούνται αυξητικά σε διάφορες επαναλήψεις.

4.4.1.1 Δραστηριότητες ανάπτυξης κατά τη διάρκεια των ροών των απαιτήσεων

Επεξεργασία του εγγράφου οράματος.

Με βάση τη φυσική γλώσσα του έγγραφου των απαιτήσεων των συμμέτοχων (δηλαδή τις ανάγκες τους), πρέπει να καθοριστούν τα χαρακτηριστικά γνωρίσματα του συστήματος. Οι ανάγκες και η κατανόηση των χαρακτηριστικών γνωρισμάτων συνδέονται με σαφής συνδέσμους ιχνηλασιμότητας του τύπου <κατανόηση>

Δημιουργία του γλωσσαρίου και του πρότυπου αντικειμένου(περιοχής).

Παράλληλα με το έγγραφο οράματος, πρέπει να αρχίσει η επεξεργασία του γλωσσάριου με τον καθορισμό και την είσοδο όλων των σχετικών όρων. Κάθε νέος όρος που προσδιορίζεται κατά τη διάρκεια μιας δραστηριότητας πρέπει να καθοριστεί, προτού να μπορέσει χρησιμοποιηθεί. Ο υπεύθυνος για την ανάπτυξη πρέπει να εξασφαλίσει ότι δεν υπάρχει ήδη ένας άλλος όρος που καθορίζεται για το ίδιο ζήτημα. Εάν ο νέος όρος έχει σχέσεις με άλλους όρους πρέπει να διαμορφωθεί στο DOM επίσης. Επιπλέον, κάθε όρος πρέπει να ταξινομηθεί σε έναν από τους ακόλουθους τύπους: άτομο, αντικείμενο ή διαδικασία. Αυτές οι κατηγορίες αναφέρονται στον τύπο του όρου που χρησιμοποιείται από τα πρότυπα και για την ονομασία των πρότυπων στοιχείων. Γνωρίζοντας τον τύπο ενός όρου, είναι δυνατόν να ελέγξουμε τη σωστή του χρήση μέσα σε ένα πρότυπο κειμένων ή μέσα σε ένα προσδιοριστικό πρότυπο αντικείμενο.

Ανάπτυξη του προτύπου περίπτωσης χρήσης.

Ως πρώτο βήμα πρέπει να διευκρινιστούν τα όρια του συστήματος και τα άτομα που θα αλληλεπιδράσουν. Τα άτομα πρέπει να καθοριστούν στο γλωσσάριο επίσης. Το επόμενο βήμα είναι να βρεθούν οι περιπτώσεις χρήσης από τα προηγούμενα χαρακτηριστικά γνωρίσματα. Μεταξύ των περιπτώσεων χρήσης και των χαρακτηριστικών γνωρισμάτων μπορούν υπάρχουν σχέσεις μ:ν, το οποίο σημαίνει ότι διάφορες περιπτώσεις χρήσης μπορούν να εκλεπτύνουν ένα χαρακτηριστικό γνώρισμα ή ότι αρκετά χαρακτηριστικά γνωρίσματα εκλεπτύνονται με μια περίπτωση χρήσης. Χαρακτηριστικά γνωρίσματα και περιπτώσεις χρήσης συνδέονται με ένα σαφή σύνδεσμο ιχνηλασιμότητας, του τύπου < εκλέπτυνση > . Η σύνδεση μεταξύ ενός ατόμου και μιας προκαλούμενης περίπτωσης χρήσης μπορεί να οδηγήσει σε έναν υπονοούμενο σύνδεσμο ιχνηλασιμότητας. Η προδιαγραφή περίπτωσης χρήσης πρέπει να ενισχυθεί με τις προδιαγραφές περιπτώσεων δοκιμής για την επαλήθευση της κατανόησης του. Οι περιπτώσεις χρήσης και οι περιπτώσεις δοκιμής πρέπει να συνδεθούν από ένα σαφή σύνδεσμο ιχνηλασιμότητας του τύπου < επαλήθευση >. Η σχέση είναι πολλαπλότητας μ: ν.

Ανάπτυξη της περιγραφής διεπαφών.

Τα έγγραφα, τα GUI-πρότυπα ή τα πρότυπα- μοντέλα μπορούν να χρησιμοποιηθούν για τις περιγραφές διεπαφών. Η περιγραφή μιας διεπαφής περιέχει τις ενώσεις μεταξύ των ατόμων και των περιπτώσεων χρήσης, στις οποίες μια διεπαφή χρησιμοποιείται, αναπαριστώντας ένα σαφή σύνδεσμο ιχνηλασιμότητας του τύπου < εκλέπτυνση>.

4.4.1.2 Δραστηριότητες ανάπτυξης της αντικειμενοστραφούς ανάλυσης

Προσδιορισμός των κλάσεων ανάλυσης.

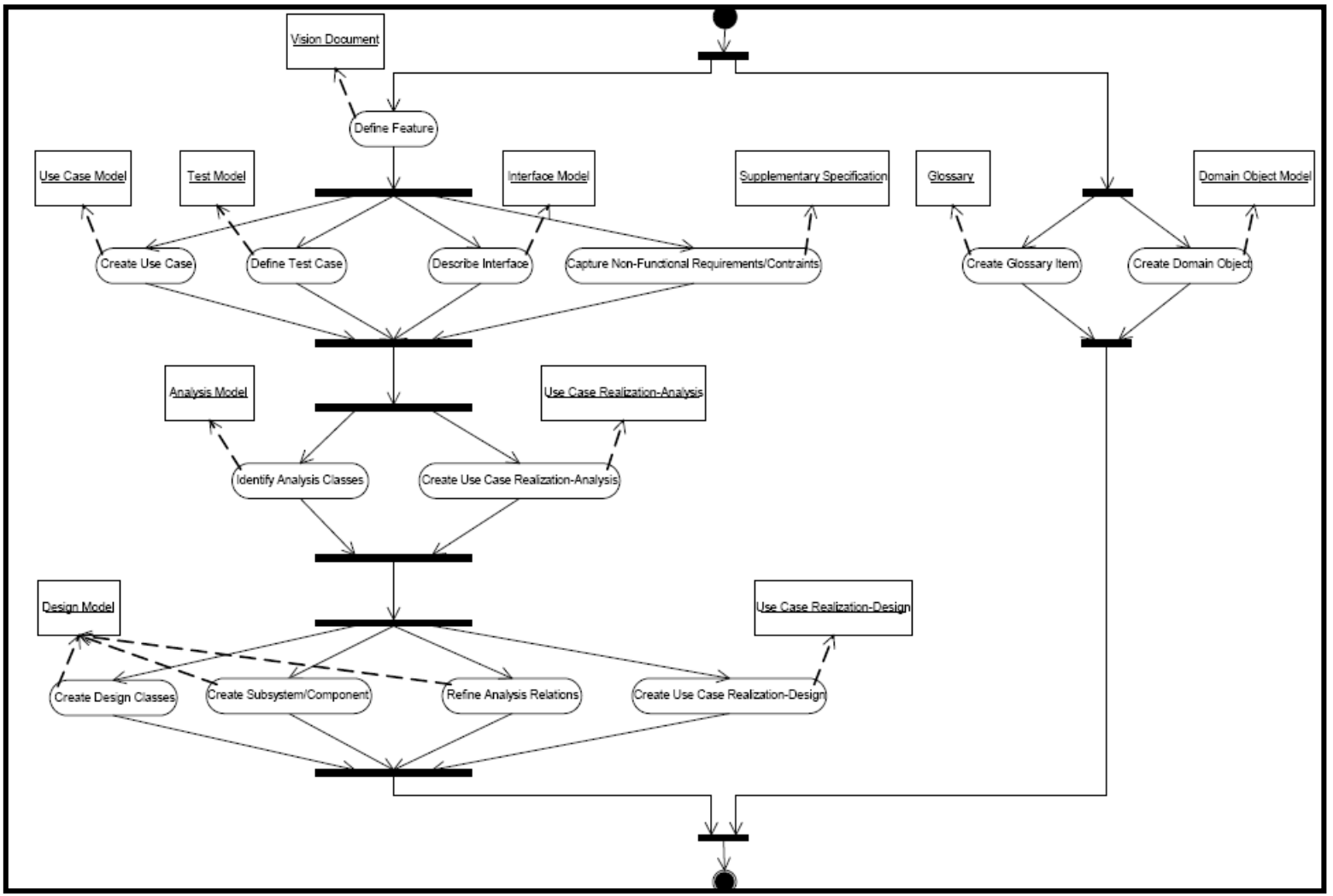
Στην φάση της ανάλυσης οι κλάσης και τα πακέτα χρησιμοποιούνται για τη διαμόρφωση της δομής του συστήματος. Στην UP ανάλυση οι κλάσης διακρίνονται ως κλάσης διεπαφών,

οντοτήτων ή ελέγχου. Υπάρχουν διαφορετικές προσεγγίσεις για την εύρεση των κλάσεων ανάλυσης. Η εξέταση των ουσιαστικών και των ρημάτων περιγραφών περιπτώσεων χρήσης είναι μια ευρέως αποδεκτή τεχνική. Τα ουσιαστικά είναι υποψήφια για τις κλάσης ή τα χαρακτηριστικά και τα ρήματα είναι υποψήφια για τις μεθόδους. Ένας άλλος τρόπος βρείτε τις κλάσης είναι η μέθοδος CRC-καρτών. Η επιλογή για μια μέθοδο καθορίζεται από το πρόγραμμα. Κάθε περίπτωση χρήσης συνδέεται με ένα σαφή σύνδεσμο ιχνηλασιμότητας στις κλάσης ανάλυσης, οι οποίες πραγματοποιούν τη ροή της. Κάθε κλάση μπορεί να συνδεθεί με περισσότερες ή μόνο μια περίπτωση χρήσης και αντίστροφα. Αυτός σημαίνει ότι μια κλάση μπορεί να πραγματοποιήσει περισσότερο από μια περιπτώσεις χρήσης.

Διαδικασία εκλέπτυνσης-ανάλυσης της περίπτωσης χρήσης.

Σε αυτό το βήμα πρέπει να περιγράφει με τη βοήθεια της UML, η συνεργασία μεταξύ των διαφορετικών κλάσεων ανάλυσης, μέσω αλληλεπιδραστικών διαγραμμάτων. Για κάθε περίπτωση χρήσης υπάρχει τουλάχιστον ένα διάγραμμα, αντιπροσωπεύοντας την επικοινωνία και τα μηνύματα μεταξύ των περιπτώσεων.

Τα διαγράμματα αλληλεπίδρασης πρέπει να συνδεθούν με τη συσχετιζόμενη περίπτωση χρήσης, που χρησιμοποιεί ένα σαφή σύνδεσμο ιχνηλασιμότητας από τον τύπο <κατανόηση>. Είναι επίσης δυνατό να συνδεθούν υπονοούμενα με τη χρήση των ονομάτων των διαγραμμάτων. Σχεδιάζοντας μηνύματα μεταξύ των κλάσεων στα διαγράμματα αλληλεπίδρασης, καθιερώνεται υπονοούμενη σύνδεση μεταξύ των αντίστοιχων κλάσεων. Αυτή η σύνδεση μπορεί να χρησιμοποιηθεί για να ελέγξει τις σχέσεις των κλάσεων.



Σχήμα 4.1: Δραστηριότητες ανάπτυξης των UP ροών: Απαιτήσεις και ανάλυση/σχέδιο.

4.4.1.3 Δραστηριότητες ανάπτυξης κατά τη διάρκεια του σχεδίου

Δημιουργία των κλάσεων σχεδίου (πρότυπο σχεδίου).

Το πρότυπο σχεδίου είναι μια εκλέπτυνση του προτύπου ανάλυσης. Σαν πρώτο βήμα όλα τα στοιχεία του προτύπου ανάλυσης πρέπει να αντιγραφούν. Τα αντιγραμμένα στοιχεία θεωρούνται ως αρχικά πρότυπα σχεδίου. Είναι δυνατό να συνδεθεί η ανάλυση και στοιχεία σχεδίου αυτόματα αντιγράφοντας ένα σαφή σύνδεσμο ιχνηλασιμότητας του τύπου <εκλέπτυνση>.

Κατά τη διάρκεια της φάσης σχεδιασμού σχεδόν όλα τα στοιχεία του αρχικού πρότυπου σχεδίου είναι λεπτομερή και σαφώς προσδιορισμένα. Κάνοντας αυτό, οι σύνδεσμοι ιχνηλασιμότητας μεταξύ των στοιχείων πρέπει να αλλαχτούν ή να επεκταθούν. Τα νέα προστιθέμενα στοιχεία σχεδίου πρέπει να συνδεθούν με τα στοιχεία ανάλυσης. Τελικά, κάθε πακέτο ανάλυσης πρέπει να συνδεθεί με ένα ή περισσότερα υποσυστήματα σχεδίου, κάθε κλάση ανάλυσης πρέπει να συνδεθεί με μια ή περισσότερες κλάσης σχεδίου ή/και διεπαφές και κάθε πραγματοποίηση-κατανόηση-ανάλυση περίπτωσης χρήσης πρέπει να συνδεθεί με ένα πραγματοποίηση- κατανόηση σχέδιο περίπτωσης χρήσης.

Εκλέπτυνση των σχέσεων ανάλυσης.

Κατά τη διάρκεια του σχεδιασμού οι σχέσεις που καθιερώνονται μεταξύ των αντικειμένων ανάλυσης πρέπει να εκλεπτυνθούν (αποσαφηνιστούν) περαιτέρω και να τοποθετηθούν στην επιλεγμένη γλώσσα προγραμματισμού. Είναι απαραίτητο να συνδεθεί η αρχική σχέση στο πρότυπο ανάλυσης και τα στοιχεία που αντικαταστάθηκαν στο πρότυπο σχεδίου, με ένα σαφή σύνδεσμο ιχνηλασιμότητας του τύπου <εκλέπτυνση>. Εάν μια κλάση ανάλυσης πραγματοποιείται στο πρότυπο σχεδίου από μια ιδιότητα μιας κλάσης ή αντίστροφα, αυτή η δραστηριότητα πρέπει να τεκμηριωθεί από ένα σύνδεσμο ιχνηλασιμότητας επίσης.

Καθιέρωση των υποσυστημάτων και των (συστατικών) επιμέρους αντικειμένων.

Η λειτουργική αποσύνθεση του συστήματος σε πακέτα αρχίζει στη φάση ανάλυσης και ολοκληρώνεται κατά τη διάρκεια της φάσης σχεδιασμού. Τα μέρη του συστήματος, χωρισμένα σε υποσυστήματα, και τα επιμέρους αντικείμενα τους επικοινωνούν μόνο χρησιμοποιώντας τις καθορισμένες διεπαφές. Τα υποσυστήματα εκλεπτύνονται σε πακέτα ανάλυσης τα οποία συνδέονται με τα αρχικά πακέτα με ένα σαφή σύνδεσμο ιχνηλασιμότητας του τύπου <εκλέπτυνση>. Νέα εισαχθέντα επιμέρους αντικείμενα και υποσυστήματα στο πρότυπο σχεδίου για να ικανοποιήσουν τις μη λειτουργικές απαιτήσεις ή τους περιορισμούς που συνδέεται με τους συνδέσμους του τύπου <κατανόηση >.

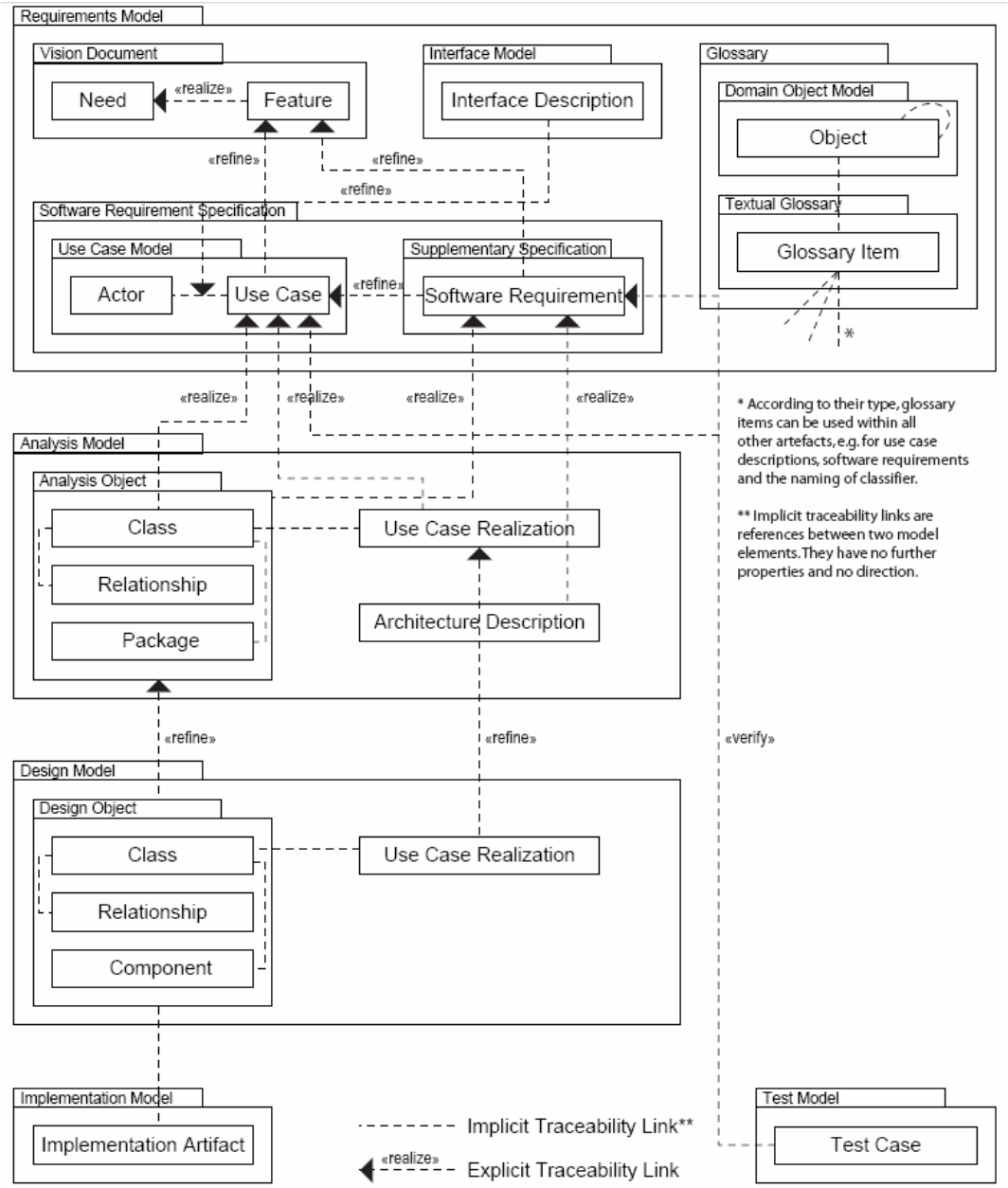
Καθιέρωση του σχεδίου κατανόησης των περιπτώσεων χρήσης.

Κατά τη διάρκεια της ανάλυσης η κατανόηση των περιπτώσεων χρήσης χρησιμοποιείται για να απαντήσει στο ερώτημα, τι πρέπει να κάνει το σύστημα, για να πραγματοποιήσει μια περίπτωση χρήσης. Κατά τη διάρκεια του σχεδιασμού αυτά τα διαγράμματα είναι περαιτέρω εκλεπτυσμένα για να επιδείξουν πώς πρόκειται να γίνει. Τα διαγράμματα πρέπει να συνδεθούν με ένα σαφή σύνδεσμο ιχνηλασιμότητας του τύπου <εκλέπτυνση> , με το αντίστοιχο διάγραμμα μέσα στο πρότυπο ανάλυσης. Επιπλέον καθιερωμένα διαγράμματα πρέπει να συνδεθούν με ένα σαφή σύνδεσμο ιχνηλασιμότητας του τύπου <κατανόηση> με τη σχετική περίπτωση χρήσης.

4.4.2 Δραστηριότητες της εφαρμογής

Το μοντέλο σχεδιασμού μετασχηματίζεται στον εκτελέσιμο κώδικα κατά τη διάρκεια της εφαρμογής. Εάν είναι δυνατό να παραχθεί ο πηγαίος κώδικας αυτόματα ή ένας υπεύθυνος για την ανάπτυξη πρέπει να τον εφαρμόσει , εξαρτάται από το επίπεδο λεπτομέρειας του μοντέλου σχεδιασμού [Maeder et al., 2007].

Εάν ο πηγαίος κώδικας παράγεται αυτόματα, καμία πρόσθετη δυνατότητα της ιχνηλασιμότητας δεν είναι απαραίτητη. Το εργαλείο που χρησιμοποιείται για αυτό συνήθως προσφέρει όλες τις λειτουργίες απαραίτητες να ακολουθήσουν ένα αντικείμενο σχεδιασμού στην εφαρμογή. Εάν ένας υπεύθυνος για την ανάπτυξη κάνει το μετασχηματισμό χειρωνακτικά, είναι δυνατόν να χρησιμοποιήσει τη συνήθη ιχνηλασιμότητα με τη βοήθεια των ονομάτων των αντικειμένων της εφαρμογής, ειδάλλως μπορούν να χρησιμοποιηθούν οι σαφής σύνδεσμοι ιχνηλασιμότητας. Οι σύνδεσμοι ιχνηλασιμότητας αποθηκεύονται στον πηγαίο κώδικα ως σχόλια.



* Σύμφωνα με τον τύπο τους, τα στοιχεία του γλωσσάριου μπορούν να χρησιμοποιηθούν μέσα σε όλα τα τεχνουργήματα, π.χ. για την περιγραφή περίπτωσης χρήσης, απαιτήσεων λογισμικού και την ονομασία των κλάσεων.

** Οι υπονοούμενοι σύνδεσμοι ιχνηλασιμότητας είναι αναφορές μεταξύ δύο στοιχείων του προτύπου. Δεν έχουν περαιτέρω ιδιότητες και καμία κατεύθυνση.

Σχήμα 4.2: Πρότυπο συνδέσμων ιχνηλασιμότητας για τις UP απαιτήσεις, την αντικειμενοστραφή ανάλυση και την ακολουθία των ροών.

4.4.3 Περιγραφή της ροής από διαγράμματα δραστηριότητας και μηχανές κατάστασης (διαγράμματα κατάστασης)

Τα διαγράμματα δραστηριότητας και οι μηχανές κατάστασης επιτρέπουν τη διαμόρφωση διαδικασιών, χωρίς έναν προγενέστερο καθορισμό της δομής του συστήματος. Τα διαγράμματα δραστηριότητας χρησιμοποιούνται ειδικά για να περιγράψουν τις ροές, π.χ. περιπτώσεις χρήσεις, ροές πληροφοριών μεταξύ των περιπτώσεων χρήσης (ως διάγραμμα αλληλεπίδρασης) ή μέθοδοι και αλγόριθμοι στο μοντέλο σχεδιασμού. Οι μηχανές κατάστασης επιτρέπουν να διαμορφωθούν τα αντιδραστικά αντικείμενα, όπως οι κλάσης, οι περιπτώσεις χρήσης, τα υποσυστήματα ή ολόκληρα συστήματα. Και οι δύο τύποι διαγραμμάτων μπορούν να χρησιμοποιηθούν στις διάφορες καταστάσεις μέσα στην αναπτυξιακή διαδικασία, γι' αυτό υπάρχει διάκριση μεταξύ τους [Χατζηγεωργίου[3], 2005].

Εάν ένα διάγραμμα δραστηριότητας ή μια μηχανή κατάστασης χρησιμοποιείται για να περιγράψει μια περίπτωση χρήσης, μια κλάση ή ένα άλλο στοιχείο, κατόπιν και τα δύο, το διάγραμμα και το στοιχείο του προτύπου έχουν συνδεθεί με ένα σαφή σύνδεσμο ιχνηλασιμότητας του τύπου <εκλέπτυνση>. Εναλλακτικά, μια σαφής ονομασία του διαγράμματος και του αντίστοιχου στοιχείου μπορεί να χρησιμοποιηθεί για την ιχνηλασιμότητα που υπονοείται.

4.5 Επαλήθευση των συνδέσμων ιχνηλασιμότητας με κανόνες επαλήθευσης

Οι καθιερωμένοι σύνδεσμοι ιχνηλασιμότητας πρέπει να ελεγχθούν για την πληρότητα και την ακρίβεια τους. Αυτό είναι απαραίτητο για να εξασφαλιστεί η δυνατότητα χρησιμοποίησης τους και για την αποφυγή της αποσύνθεσης των πληροφοριών ιχνηλασιμότητας μετά από τις αλλαγές στα συνδεδεμένα πρότυπα. Παρακάτω, καθορίζονται οι κανόνες επαλήθευσης[Maeder et al., 2007].

Προς το παρόν αυτό το σύνολο κανόνων, ως πρώτο βήμα, καλύπτει την επαλήθευση για την ύπαρξη των συνδέσμων ιχνηλασιμότητας. Φθάνοντας σε αυτόν τον στόχο η ανάλυση των όρων που χρησιμοποιούνται στα προσδιοριστικά, η αξιολόγηση των σχέσεων στις κλάσης των μοντέλων ή η ανάλυση των περιγραφών περιπτώσεων χρήσης είναι απαραίτητη. Ένα παράδειγμα για ένα σύνολο κανόνων που εισάγεται στον πίνακα 1 είναι: κάθε περίπτωση χρήσης πρέπει να κατανοείται από τουλάχιστον μια κλάση ανάλυσης. Αυτός ο κανόνας επαληθεύει την ύπαρξη τουλάχιστον ενός συνδέσμου ιχνηλασιμότητας και μεταξύ των δύο πρότυπων στοιχείων. Αλλά αυτό δεν είναι απολύτως ικανοποιητικό για την επαλήθευση της ορθότητας. Προσεγγίσεις για περαιτέρω επικυρώσεις προσφέρουν τη χρήση όρων στο πρότυπο και την επικύρωση της ευλογοφάνειας μεταξύ διαγραμμάτων. Παραδείγματος χάριν, η ανάλυση των όρων σημαίνει, την αναζήτηση για στοιχεία του γλωσσάριου του τύπου του αντικειμένου, στη περιγραφή της περίπτωσης χρήσης, και προσπάθεια να συσχετιστεί το προσδιοριστικό τους από τις συνδεμένες κλάσης ανάλυσης και τις ιδιότητές τους. Διαφορές μεταξύ των δύο σημαίνει ότι πρέπει να ειδοποιηθεί ο υπεύθυνος για την ανάπτυξη.

Έλεγχος ευλογοφάνειας μεταξύ των διαφορετικών διαγραμμάτων, σημαίνει ότι για κάθε περίπτωση χρήσης που προκαλείται από ένα άτομο, πρέπει να καθοριστεί η κλάση ανάλυσης του

τύπου της διεπαφής. Άλλη περίπτωση εξετάζει την κατανόηση των περιπτώσεων χρήσης, κλάσεις από όλες τις περιπτώσεις μέσω της κατανόησης των περιπτώσεων χρήσης πρέπει να συνδέονται με την περίπτωση χρήσης, επειδή την κατανοούν. Παρατίθενται οι εξής μέχρι τώρα γνωστοί κανόνες:

Ανάγκη - «πραγματοποίηση» - Χαρακτηριστικό γνώρισμα (μ: ν)	
1.	Κάθε ανάγκη πραγματοποιείται από τουλάχιστον ένα χαρακτηριστικό γνώρισμα.
2.	Κάθε χαρακτηριστικό γνώρισμα πραγματοποιεί τουλάχιστον μια ανάγκη.
Χαρακτηριστικό γνώρισμα - «εκλέπτυνση» - Περίπτωση Χρήσης (μ: ν)	
1.	Κάθε χαρακτηριστικό γνώρισμα εκλεπτύνεται από τουλάχιστον μια περίπτωση χρήσης.
2.	Κάθε περίπτωση χρήσης εκλεπτύνει τουλάχιστον ένα χαρακτηριστικό γνώρισμα.
Συσχέτιση Περίπτωση Χρήσης – Δράστης - «εκλέπτυνση» - Περιγραφή Διεπαφής (μ: ν)	
1.	Κάθε συσχέτιση μεταξύ μιας περίπτωσης χρήσης και ενός δράστη εκλεπτύνεται από τουλάχιστον μια περιγραφή διεπαφής.
2.	Κάθε περιγραφή διεπαφής εκλεπτύνει τουλάχιστον μια συσχέτιση μεταξύ του δράστη περίπτωσης χρήσης.
Δράστης - - - - - Περίπτωση Χρήσης (μ: ν)	
1.	Κάθε δράστης συνδέεται τουλάχιστον σε μια περίπτωση χρήσης.
2.	Ο συσχετιζόμενος δράστης είναι ο ίδιος με τους δράστες που χρησιμοποιούνται στην περιγραφή της περίπτωσης χρήσης.
Περίπτωση Χρήσης - «εκλέπτυνση» - Απαίτηση Λογισμικού (μ: ν)	
1.	Κάθε απαίτηση λογισμικού (μη λειτουργική απαίτηση, περιορισμός) εκλεπτύνει τουλάχιστον μια περίπτωση χρήσης.
Περίπτωση χρήσης/ Απαίτηση Λογισμικού - «επαλήθευση» - Περίπτωση Δοκιμής (μ: ν)	
1.	Κάθε απαίτηση λογισμικού επαληθεύεται από τουλάχιστον μια περίπτωση δοκιμής.
2.	Κάθε περίπτωση δοκιμής επαληθεύει τουλάχιστον μια περίπτωση χρήσης ή απαίτηση λογισμικού.
Γλωσσάριο - - - - - DOM (1: 0,1)	
1.	Κάθε αντικείμενο περιοχής καθορίζεται στο γλωσσάριο.
Περίπτωση Χρήσης - «πραγματοποίηση» - Κλάση Ανάλυσης (μ: ν)	
1.	Κάθε περίπτωση χρήσης πραγματοποιείται από τουλάχιστον μια κλάση ανάλυσης.
2.	Κάθε κλάση ανάλυσης πραγματοποιεί τουλάχιστον μια περίπτωση χρήσης.
Περίπτωση Χρήσης - «πραγματοποίηση» - Ανάλυση Πραγματοποίησης Περίπτωσης Χρήσης (1: ν)	
1.	Κάθε πραγματοποίηση περίπτωσης χρήσης πραγματοποιεί μια περίπτωση χρήσης.

Πίνακας 4.1 : Κανόνες επαλήθευσης συνδέσμων ιχνηλασιμότητας.

Εφαρμόζοντας τους καθορισμένους κανόνες, πρέπει να σημειωθεί ότι η UP είναι μία επαυξητική και επαναληπτική διαδικασία. Αυτό σημαίνει ότι αυτοί οι κανόνες θα αυξήσουν τις προειδοποιήσεις αν το πρότυπο δεν είναι πλήρως ολοκληρωμένο. Εντούτοις είναι δυνατόν να ελέγξει όλες τις αλυσίδες των τεχνουργημάτων στο τελευταίο υπάρχον τεχνούργημα και όλα τα πιο χαλαρά τεχνουργήματα. Ένα παράδειγμα για ένα χαλαρό τεχνουργήματα είναι μια περίπτωση χρήσης, η οποία πραγματοποιείται από μια κλάση ανάλυσης, αλλά δεν εκλεπτύνει οποιοδήποτε χαρακτηριστικό γνώρισμα της. Αυτό πρέπει να οδηγήσει σε μια προειδοποίηση για τον υπεύθυνο για την ανάπτυξη.

4.6 Επίλογος

Οι σύνδεσμοι ιχνηλασιμότητας βελτιώνουν τη συντηρησιμότητα και τις εξελικτικές αναπτυξιακές διαδικασίες υποστήριξης, π.χ. με την ανάκτηση προηγούμενων δραστηριοτήτων ανάπτυξης, ειδικά για τη περίπτωση των μεταβαλλόμενων απαιτήσεων. Σε αυτή την ενότητα οι δραστηριότητες ενός προτύπου διαδικασίας λογισμικού έχουν ενισχυθεί με ορισμούς και κανόνες για τους συνδέσμους ιχνηλασιμότητας που μειώνουν την προσπάθεια και επιτρέπουν την υποστήριξη εργαλείων. Έχει εισαχθεί να πρότυπο για τους συνδέσμους ιχνηλασιμότητας που μπορεί να προσαρμοστεί εάν είναι απαραίτητο. Με βάση τις δραστηριότητες ανάπτυξης και τεχνουργήματα, έχει αναπτυχθεί ένα σύνολο κανόνων για την επαλήθευση των συνδέσμων ιχνηλασιμότητας.

Η ανάπτυξη πρότυπου συνδέσμων ιχνηλασιμότητας ολοκληρώνεται και εκλεπτύνεται προς πλήρη κάλυψη των συστηματικών δραστηριοτήτων, για να διευκολύνει την κατάλληλη υποστήριξη εργαλείων για τη δημιουργία, αναπροσαρμογή και επαλήθευση των συνδέσμων ιχνηλασιμότητας με μια ελάχιστη αλληλεπίδραση με τον υπεύθυνο για την ανάπτυξη.

5. Υιοθέτηση της οδηγημένης με πρότυπα ανάπτυξης (Model-Driven Development -MDD)

Η αυξανόμενη πολυπλοκότητα των συστημάτων λογισμικού έχει οδηγήσει στη διαδεδομένη υιοθέτηση της οδηγημένης με πρότυπα ανάπτυξης (Model-Driven Development - MDD). Η οδηγημένη με πρότυπα ανάπτυξη (MDD) διευκολύνει την ανάπτυξη με την προσθήκη μιας νέας έννοιας στα τεχνουργήματα λογισμικού. Εντούτοις, επιβάλλει μια νέα απαίτηση στη δοκιμή των δραστηριοτήτων: συμμόρφωση της εφαρμογής με τα πρότυπα. Η βασισμένη σε

πρότυπα δοκιμή (Model Based Testing -MBT) χρησιμοποιεί αυτά τα υψηλού επιπέδου τεχνουργήματα ως βάση για την παραγωγή δοκιμής. Είναι συμπληρωματικός έλεγχος στην βασισμένη στον κώδικα δοκιμή (Code Based Testing - CBT), επειδή, ενώ οι προσεγγίσεις CBT εξετάζουν την εφαρμογή, οι MBT προσεγγίσεις εξετάζουν την εφαρμογή ενάντια στις προσδοκίες για τα συστήματα (υπό μορφή προτύπων). Επομένως, η MBT αποκαλύπτει συχνά τα ελαττώματα που η CBT δεν βρίσκει, όπως η απώλεια των λειτουργιών. Επιπλέον, η αυτοματοποιημένη υποστήριξη για τη δοκιμή των συστημάτων βάσισε στις δυνάμεις προτύπων τους την προσπάθεια που τέθηκε στη δημιουργία των προτύπων. Η MBT, εντούτοις, εισάγει τις νέες προκλήσεις για τη δοκιμή των δραστηριοτήτων. Μια κύρια πρόκληση για την MBT είναι πώς να δημιουργήσει και να διατηρήσει, τις σαφής καθορισμένες σχέσεις μεταξύ των τεχνουργημάτων που είναι προς δοκιμή
.

Ο ρόλος που διαδραματίστηκε από αυτές τις σχέσεις για να υποστηρίξει την αυτοματοποίηση της δοκιμής των δραστηριοτήτων, ήταν αναγνωρισμένος από καιρό [Dick and Faivre, 1993]. Καθιερώνεται με διαφορετικούς σκοπούς που περιλαμβάνουν: χαρτογράφηση των εννοιών από τα πρότυπα έως την εφαρμογή, η οποία υποστηρίζει την παραγωγή σεναρίων για δοκιμή ή την εκτέλεση των αφηρημένων σεναρίων δοκιμής που αφορούν τα πρότυπα, τις ακολουθίες δοκιμής και τα αποτελέσματα, τα οποία υποστηρίζουν τη μέτρηση κάλυψης και την βασισμένη σε πρότυπα εκλεκτική δοκιμή οπισθοδρόμησης.

Η δοκιμή οπισθοδρόμησης είναι «ένας απαραίτητος αλλά ακριβός στόχος συντήρησης» [Rothermel and Harrold, 1996]. Η εκλεκτική δοκιμή οπισθοδρόμησης στοχεύει στη μείωση της εξεταστικής προσπάθειας οπισθοδρόμησης, με την επιλογή ενός υποσυνόλου των περιπτώσεων δοκιμής για επανάληψη, λαμβάνοντας παρόμοια έκβαση. Ο καταμερισμός των σχέσεων που χρησιμοποιούνται για να υποστηρίξουν την δοκιμή οπισθοδρόμησης επηρεάζει την ακρίβεια επιλεκτικότητας. Όσο πιο σαφής, τόσο ακριβέστερη μπορεί να είναι η επιλογή [Naslavsky and Richardson, 2007].

Η δοκιμή για τις αρχικές προσδοκίες μπορεί να γίνει με δοκιμή βασισμένη στα πρότυπα, η οποία υιοθετεί τα υψηλού επιπέδου πρότυπα ως βάση για την παραγωγή δοκιμών. Εκτός από την παραγωγή δοκιμών, οι προκλήσεις για τη δοκιμή βασισμένη στα πρότυπα, περιλαμβάνουν τη δημιουργία και τη συντήρηση πληροφοριών ιχνηλασιμότητας μεταξύ των σχετικών τεχνουργημάτων. Η ιχνηλασιμότητα απαιτείται για να υποστηρίξει δραστηριότητες όπως την αξιολόγηση του αποτελέσματος, την δοκιμή οπισθοδρόμησης και την ανάλυση του περιεχομένου. Η οδηγημένη με πρότυπα ανάπτυξη (MDD) και ο μετασχηματισμός προτύπων είναι οι λύσεις που εξετάζουν το πρόβλημα ιχνηλασιμότητας με τη δημιουργία σχέσεων μεταξύ των μετασχηματισμένων τεχνουργημάτων σε όλη τη διάρκεια της διαδικασίας. Οι σχέσεις δημιουργούνται κατά τη διάρκεια της δοκιμής της διαδικασίας παραγωγής. Η προσεκτική μελέτη τους, επιτρέπει την υποστήριξη για την αξιολόγηση αποτελεσμάτων, την ανάλυση του περιεχομένου και τη δοκιμή οπισθοδρόμησης [Naslavsky et al., 2007].

5.1 Η δοκιμή βασισμένη στα πρότυπα συμπληρώνει την βασισμένη σε κώδικα δοκιμή

Στις παραδοσιακές διαδικασίες ανάπτυξης λογισμικού, ο πηγαίος κώδικας ήταν το μόνο τεχνούργημα που χρησιμοποιούταν για δοκιμαστικούς λόγους. Λαμβάνοντας υπόψη το επίπεδο

των αφηρημένων εννοιών και τον πηγαίο κώδικα, πολλές λύσεις είναι διαθέσιμες σήμερα στη βιομηχανία και στον ακαδημαϊκό κόσμο [Xie et al., 2004] , [Csallner et al., 2004] που αυτοματοποιούν μερικές δοκιμαστικές δραστηριότητες και μειώνουν τις προσπάθειες ελέγχου. Η δοκιμή βασισμένη σε κώδικα χρησιμοποιεί την εφαρμογή για να παραγάγει περιπτώσεις δοκιμής. Κατά συνέπεια, μπορεί να χρησιμοποιηθεί για να εξετάσει το σύστημα, αλλά δεν μπορεί να χρησιμοποιηθεί για να εξετάσει τις αρχικές προσδοκίες για το σύστημα. Η υιοθέτηση των υψηλού επιπέδου προτύπων ως βάση για την παραγωγή δοκιμών μπορεί να εξετάσει τα συστήματα ενάντια στις αρχικές προσδοκίες. Επίσης επηρεάζει την προσπάθεια που γίνεται για τη δημιουργία προτύπων που υποστηρίζουν τις δοκιμές.

Η οδηγημένη με πρότυπα ανάπτυξη (MDD) προσθέτει ένα νέο επίπεδο αφηρημένων εννοιών στα τεχνουργήματα ανάπτυξης λογισμικού. Η πρόσφατη διαδεδομένη υιοθέτησή της είναι μια έκβαση από την ανάγκη να αντιμετωπιστεί η αυξανόμενη πολυπλοκότητα των συστημάτων λογισμικού. Η διάδοση της προδιαγραφής UML προτύπων και η αυξανόμενη διαθεσιμότητα των εργαλείων που υποστηρίζουν την παραγωγή κώδικα από τα πρότυπα είναι πρόσθετοι παράγοντες που συμβάλλουν στην υιοθέτηση της MDD. Οι συνέπειες είναι διπλές: (1) ο κώδικας δεν είναι πλέον το σημαντικότερο τεχνούργημα του λογισμικού, (2) ο κώδικας δεν είναι πλέον η μοναδική πηγή για την αυτοματοποιημένη επιλογή περιπτώσεων δοκιμής. Η δοκιμή βασισμένη σε πρότυπα συμπληρώνει τώρα την δοκιμή βασισμένη σε κώδικα.

Οι παραδοσιακές δοκιμαστικές δραστηριότητες περιλαμβάνουν: παραγωγή περιπτώσεων δοκιμής, εκτέλεση των δοκιμών, αξιολόγηση αποτελέσματος, ανάλυση του περιεχομένου, και δοκιμές οπισθοδρόμησης. Οι περισσότερες βασισμένες σε πρότυπα δοκιμές αυτοματοποιημένης προσέγγισης επικεντρώνονται συχνά στις δραστηριότητες παραγωγής και εκτέλεσης δοκιμής, παραμελώντας άλλες. Όπως αναφέρεται στο [Dick and Faivre, 1993], η δοκιμή βασισμένη στα πρότυπα απαιτεί τη δυνατότητα να συσχετίζονται οι «αφηρημένες τιμές της προδιαγραφής με τις συγκεκριμένες τιμές της εφαρμογής». Ο ρόλος που διαδραματίστηκε από τις σχέσεις μεταξύ των τεχνουργημάτων για να υποστηρίξει την αυτοματοποίηση της δοκιμής των δραστηριοτήτων ήταν αναγνωρισμένος από καιρό [Richardson et al., 1992] , [Dick and Faivre, 1993]. Οι σχέσεις μπορούν να καθιερωθούν με διαφορετικούς σκοπούς:

1. Σχέσεις που χαρτογραφούν έννοιες στο πρότυπο οι οποίες βοηθούν στην παραγωγή σεναρίων δοκιμής για την υποστήριξη της εφαρμογής ή την αφηρημένη εκτέλεση σεναρίων δοκιμής.

2. Σχέσεις από τα πρότυπα στις περιπτώσεις δοκιμής και από τις περιπτώσεις δοκιμής στη μέτρηση των αποτελεσμάτων της δοκιμής και την εκλεκτική δοκιμή οπισθοδρόμησης.

Η δημιουργία και η συντήρηση σαφών σχέσεων μεταξύ των δοκιμαστικών τεχνουργημάτων, είναι επομένως, μια κύρια πρόκληση στην αυτοματοποιημένη υποστήριξη τέτοιων δραστηριοτήτων.

Τα τεχνουργήματα στο τέλος τους καθορίζουν το επίπεδο μιας σχέσης ανάπτυξης. Όταν οι σχέσεις χρησιμοποιούνται για να υποστηρίξουν τον προσδιορισμό περιπτώσεων δοκιμής (ή των προτύπων δοκιμής), η ανάπτυξη τους επηρεάζει την ακρίβεια της διαδικασίας της επιλογής τους. Όσο πιο προσεκτικά διαλεγμένα είναι τα σημεία του τέλους των τεχνουργημάτων , τόσο ακριβέστερη μπορεί να είναι η επιλογή (π.χ. κώδικας βασισμένος στην οπισθοδρόμηση). Υπάρχει εντούτοις, μια συσχέτιση μεταξύ της ακρίβειας που λαμβάνεται και του κόστους, για να διατηρηθούν τέτοιες σχέσεις [Balasubramaniam et al., 2001]. Σε αυτή την ενότητα, χρησιμοποιείται μια λύση που επιτρέπει την εξωτερικοποίηση των σχέσεων με διαφορετικά

επίπεδα ανάπτυξης [Didonet et al., 2006]. Αυτή η λύση υιοθετείται επειδή υποστηρίζει την έρευνα τέτοιων συσχετίσεων.

Σε αυτή την ενότητα χρησιμοποιούνται τεχνικές ιχνηλασιμότητας μετασχηματισμού προτύπων [Jouault , 2005] για να δημιουργηθούν σχέσεις μεταξύ των βασισμένων σε πρότυπο δοκιμαστικών τεχνουργημάτων. Οι σχέσεις δημιουργούνται κατά τη διάρκεια της διαδικασίας των δοκιμών. Προσαρμόζεται ένα βασισμένο σε πρότυπο μοντέλο ροής ελέγχου [Rountev et al., 2005] και χρησιμοποιείται για να υποστηριχτεί η παραγωγή στοιχείων περιπτώσεων δοκιμής από τα διαγράμματα ακολουθίας στα πρότυπα UML 2.0. Επιπλέον, προσαρμόζεται ένα πρότυπο ιεραρχίας δοκιμής [Stocks and Carrington, 1996] και χρησιμοποιείται για να περιγραφεί η ιεραρχία των μερών των στοιχείων δοκιμής. Ένα μεταμοντέλο (ιχνηλασιμότητας) χρησιμοποιείται για να υποστηρίξει τη δημιουργία σχέσεων μεταξύ αυτών των προτύπων [Didonet, 2006].

5.2 Τεχνουργήματα που συσχετίζονται με τις δοκιμές

5.2.1 Διαγράμματα Ακολουθίας σε UML 2.0 μεταμοντέλα

Στην τεχνολογία λογισμικού, τα σενάρια χρησιμοποιούνται για να εκφράσουν τις απαιτήσεις συστημάτων, τα συστατικά και τις αλληλεπιδράσεις αντικειμένων, και τα έγγραφα δοκιμής [Egyed, 2003, Jeremiah and Frank, 2001, Uchitel et al., 2004]. Διαδραματίζουν έναν σημαντικό ρόλο στο λογισμικό που τα εξετάζει επειδή τα συγκεκριμένα έγγραφα δοκιμής είναι σενάρια από κάποια επίπεδα του κώδικα που σχολιάζονται με επιπλέον πληροφορίες για να υποστηρίξουν τη σύγκριση μεταξύ των επιτευχθέντων και των αναμενόμενων αποτελεσμάτων. Τα σενάρια χρησιμοποιούνται για να περιγράψουν τις προσδοκίες για το σύστημα. Ανάλογα με το επίπεδο της αφαίρεσης στην οποία περιγράφονται, μπορούν να χρησιμοποιηθούν και να γίνουν κατανοητά από την πλειοψηφία των συμμέτοχων (χρήστες, πελάτες, υπεύθυνοι για την ανάπτυξη, ελεγκτές, κ.λπ.).

Τα διαγράμματα ακολουθίας UML είναι σενάρια στο σχεδιαστικό επίπεδο αφαίρεσης. Χρησιμοποιούνται για να διαμορφώσουν τη ροή των μηνυμάτων μεταξύ των αντικειμένων μέσα σε ένα σύστημα κατά τρόπο οπτικό, εστιάζουν στη χρονική σειρά στην οποία στέλνονται τα μηνύματα. Συνήθως χρησιμοποιούνται για την περιγραφή των κύριας χρήσης σεναρίων, για τη λογική σειρά μεθόδων ή υπηρεσιών. Στην πραγματικότητα, οι βασισμένες σε πρότυπο δοκιμαστικές προσεγγίσεις θεωρούν τα διαγράμματα ακολουθίας ως πραγματοποιήσεις των περιπτώσεων χρήσης [Briand and Labiche, 2001, Nebut et al., 2006]. Επομένως, οι δοκιμαστικές προσεγγίσεις βασισμένες στα σενάρια εξετάζουν τις ανησυχίες της χρησιμοποίησης των προσδοκιών για να εξετάσουν το σύστημα. Για αυτόν τον λόγο, υιοθετούμε τα διαγράμματα ακολουθίας ως βάση για την παραγωγή δοκιμής.

5.2.2 Βασισμένο σε πρότυπο μοντέλο γραφικής παράστασης ελέγχου ροής

Τα διαγράμματα ακολουθίας διαμορφώνουν την αναμενόμενη ροή των μηνυμάτων μεταξύ των αντικειμένων ενός συστήματος, σύμφωνα με μια ιδιαίτερη εισαγωγή. Κατά συνέπεια, είναι δυνατό να περιγραφούν διαφορετικές κλήσεις ακολουθίας μηνυμάτων σε ένα ενιαίο διάγραμμα ακολουθίας. Μια γνωστή στρατηγική για παραγωγή περίπτωσης δοκιμής από τα διαγράμματα ακολουθίας [Binder, 2000] αποτελείται από την άντληση ενός προτύπου ελέγχου ροής από το διάγραμμα ακολουθίας, που αναπτύσσει ένα σύνολο πορειών που επιτυγχάνει τα ιδιαίτερα κριτήρια κάλυψης βασισμένα στο πρότυπο ροής ελέγχου (π.χ. κάλυψη κλάδων, ή κάλυψη πορειών), και που προσδιορίζει τις εισαγωγές που προκαλούν τις πορείες που έχουν τεθεί. Κατά συνέπεια, κάθε πορεία που προήλθε από τα χωρίσματα διαγραμμάτων ακολουθίας, θέτει μια έγκυρη εισαγωγή που τέθηκε στις εισαγωγές στοιχείων περιπτώσεων δοκιμής.

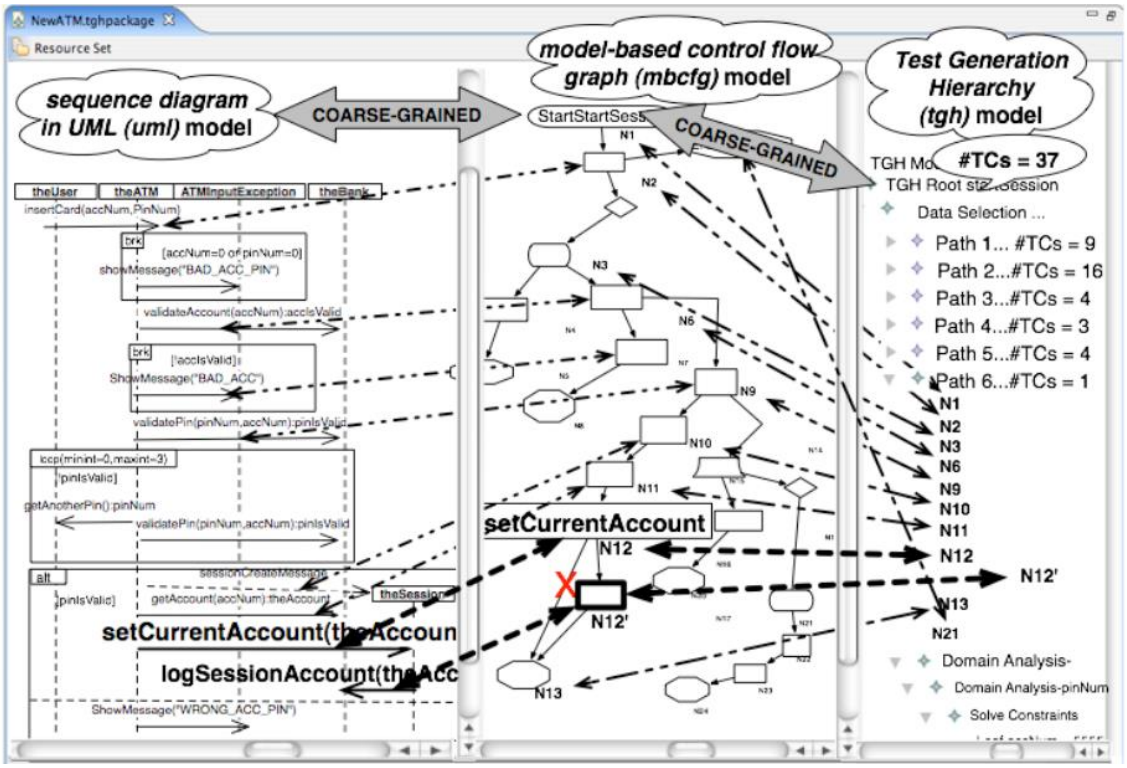
Οι γραφικές παραστάσεις ροής ελέγχου που προέρχονται από τα πρότυπα ονομάζονται μοντέλα ροής ελέγχου βασισμένα σε πρότυπα. Στη βιβλιογραφία, υπάρχουν μερικά προτεινόμενα πρότυπα [Garousi et al., 2005], [Rountev et al., 2005]. Στο [Garousi et al., 2005] συζητά ένα περιεκτικό σύνολο τέτοιων προτύπων. Προτείνουν μια επέκταση από το διάγραμμα δραστηριότητας UML 2.0 μεταμοντέλων, ονομασμένο Concurrent Control Flow Graph (CCFG), για να υποστηρίξουν την ανάλυση ροής ελέγχου των διαγραμμάτων ακολουθίας. Ένα σημαντικό σημείο αυτής της εργασίας είναι ότι υποστηρίζει τον συναγωνισμό.

5.3 Κίνητρο

Σε αυτή την υποενότητα περιγράφονται διάφορα σχετικά πρότυπα, μεταμοντέλα, μετασχηματισμοί προτύπων, και μια διαδικασία παραγωγής δοκιμών που κρύβεται κάτω από την προσέγγισή. Περιγράφεται επίσης η δημιουργία σαφών σχέσεων μεταξύ των προτύπων για να υποστηριχτούν οι δραστηριότητες δοκιμών. Αυτό το τμήμα καταδεικνύει και παρακινεί την προσέγγισή με μια εφαρμογή παραδείγματος της υποδομής και της ροής των εργασιών που καθορίζονται από τα πρότυπά και τις σχέσεις τους. Το σχήμα 5.1 απεικονίζει τις απόψεις των προτύπων σχετικών από την προσέγγισή μας: ένα διάγραμμα ακολουθίας από πρότυπο UML 2.0 (uml), μια βασισμένη σε πρότυπο γραφική παράσταση ελέγχου ροής (mbcfg), και μια ιεραρχία παραγωγής δοκιμής (tgh). Αργότερα, οι παράγραφοι 5.4 και 5.5 περιγράφουν τα πρότυπα και τη δημιουργία τους λεπτομερέστερα. Τα σχήματα 5.5, και 5.7 απεικονίζουν λεπτομερώς το mbcfg, και το πρότυπο tgh του σχήματος 5.1, ενώ το σχήμα 5.4 απεικονίζει μια άλλη έκδοση του προτύπου uml, πριν από την προσθήκη του μηνύματος logSessionAccount.

Η προτεινόμενη υποδομή που κρύβεται στην προσέγγισή αυτής της ενότητας επιτρέπει την αυτοματοποιημένη υποστήριξη σε παρόμοιες εργασίες, όπως την παραγωγή δοκιμών (π.χ. δοκιμή οπισθοδρόμησης). Η διαθεσιμότητα των σαφώς καθορισμένων σχέσεων μπορεί να χρησιμοποιηθεί για να καθορίσει τον αντίκτυπο που έχει μια αλλαγή ενός πρότυπου, σε μια ακολουθία δοκιμής. Ανάλογα με τον διαμερισμό των σχέσεων, είναι δυνατό να βγάλουμε το συμπέρασμα ότι μια αλλαγή ενός μηνύματος σε ένα διάγραμμα ακολουθίας προσκρούει σε ένα

υποσύνολο περιπτώσεων δοκιμής, ενώ άλλες περιπτώσεις δοκιμής δεν επηρεάζονται. Επομένως, η επανάληψη μερικών ακόμη περιπτώσεων δοκιμής απαιτείται για να εξεταστεί εάν η αλλαγή παρενέβαλε νέα ελαττώματα στο σύστημα. Η διαφορετική ακρίβεια επιλεκτικότητας επιτυγχάνεται ανάλογα με τον καταμερισμό και τη σημασιολογία των διαθέσιμων σχέσεων. Υπάρχει μια συσχέτιση μεταξύ της ακρίβειας που λαμβάνεται και του κόστους για να εξωτερικεύσουν και να διατηρήσουν τέτοιες σχέσεις. Η υποδομή που δημιουργείται με την προσέγγισή μας θα χρησιμοποιηθεί για να ερευνήσει αυτήν την συσχέτιση.



Σχήμα 5.1: Απόψεις των σχετικών προτύπων.

Για παράδειγμα, το σχήμα 5.1 παρουσιάζει τρία πρότυπα με σαφώς καθορισμένες και μη σχέσεις (διπλής κατεύθυνσης βέλη) μεταξύ τους. Οι σχέσεις αντιπροσωπεύουν ένα ή περισσότερα ελλοχεύοντα (υπονοούμενα) πρότυπα ιχνηλασιμότητας. Τα κατευθυνόμενα βέλη δείχνουν ότι το μήνυμα setCurrentAccount (στο πρότυπο uml) συσχετίζεται με τον κόμβο setCurrentAccount (στο mbcfg), ο οποίος συσχετίζεται στον κόμβο N12 (στο tgh). Υποθέστε το πρότυπο uml μέσα στο σχήμα 5.1 αρχικά δεν είχε το μήνυμα logSessionAccount (δείτε το σχήμα 5.4), έτσι ο κόμβος N12′ στο mbcfg και tgh πρότυπα δεν θα υπήρχε αρχικά, ούτε άκρες N12-N12′ και N12'- N13 στο πρότυπο mbcfg. Η προσθήκη του μηνύματος logSessionAccount οδηγεί στην προσθήκη του κόμβου N12′ στο πρότυπο mbcfg, στην προσθήκη των ακρών N12-N12′ και N12'-N13, και στην αφαίρεση της άκρης από N12-N13, που απεικονίζονται στο σχήμα 5.1. Αυτή η αλλαγή, στη συνέχεια, προσκρούει σε μερικά μονοπάτια στην ιεραρχία παραγωγής δοκιμών (Path5 και Path6). Επομένως, επηρεάζει τις περιπτώσεις δοκιμής κάτω από εκείνα τα μέρη (χωρίσματα). Οι άλλες 32 περιπτώσεις δοκιμής που παράγονται από το ίδιο διάγραμμα ακολουθίας (TGH Root startSession στο πρότυπο tgh), δεν επηρεάζονται. Η απόκτηση αυτών των πληροφοριών είναι δυνατή επειδή οι απεικονισμένες σχέσεις συνδέουν τις σαφώς καθορισμένες

οντότητες (μηνύματα) με τα μέρη (χωρίσματα) περιπτώσεων δοκιμής, η δημιουργία των οποίων επηρεάστηκε από αυτές τις οντότητες.

Εάν οι σαφώς καθορισμένες σχέσεις δεν ήταν ακριβής, δεν θα ενσωματώνονταν στον αλγόριθμο παραγωγής δοκιμής. Κατά συνέπεια, ο ακριβής προσδιορισμός ενός υποσυνόλου περιπτώσεων δοκιμής για μια δοκιμασία οπισθοδρόμησης θα ήθελε εντατικότερη εργασία. Επιπλέον, λαμβάνοντας υπόψη την ίδια αλλαγή, οι μη σαφώς καθορισμένες σχέσεις θα προσδιορίσουν 37 περιπτώσεις δοκιμής για να γίνει δοκιμασμένη οπισθοδρόμηση (όλες τις περιπτώσεις δοκιμής κάτω από το TGH Root startSession επηρεάστηκαν).

Μια άλλη ελπιδοφόρος εφαρμογή για την υποδομή που καθιερώνεται με αυτή την προσέγγισή είναι αξιολόγηση συμπεριφοράς του αποτελέσματος των περιπτώσεων δοκιμής. Στην πραγματικότητα, ένα πλεονέκτημα της βασισμένης σε πρότυπο δοκιμής είναι ότι χρησιμοποιεί τα πρότυπα σαν δοκιμαστικές προβλέψεις (test oracles). Οι δοκιμαστικές προβλέψεις αποτελούνται από τη διαδικασία προβλέψεων και τις διατιθέμενες πληροφορίες. Η διαδικασία χρησιμοποιεί τις πληροφορίες για να ελέγξει ότι η συμπεριφορά εκτέλεσης είναι η αναμενόμενη [Richardson, 1994] . Μια πρόβλεψη μπορεί επομένως να ελέγξει ότι ένα τελικό αποτέλεσμα ήταν το αναμενόμενο. Μπορεί επίσης να συγκρίνει ότι τα μέτρα που λαμβάνονται για να φθάσουμε στο αποτέλεσμα είναι τα ίδια βήματα που περιγράφονται στο πρότυπο. Αυτό είναι δυνατό επειδή οι περιπτώσεις δοκιμής ανήκουν σε μια ιεραρχία που καθορίζει μια πορεία ως γονέα της, έτσι οι πορείες χρησιμοποιούνται ως προβλέψεις συμπεριφοράς.

Εάν υποθέσουμε ότι χαρτογραφούμε τις σχέσεις από τις έννοιες στο πρότυπο και από τις έννοιες στην εφαρμογή, που είναι διαθέσιμες με το πρότυπο της uml, οι περιπτώσεις δοκιμής στην ιεραρχία παραγωγής δοκιμής μπορούν να εκτελεσθούν ενάντια στην εφαρμογή. Κατά συνέπεια, εάν τα αποκτηθείσα αποτελέσματα αντιστοιχούν με τα αναμενόμενα, αλλά η αποκτηθείσα ακολουθία μηνυμάτων δεν ταιριάζει με την πορεία, η εφαρμογή δεν «συμμορφώνεται» με το πρότυπο.

Για παράδειγμα, υποθέτουμε ότι το αναμενόμενο αποτέλεσμα για την εκτέλεση των υποθέσεων του κάτω μέρους του Path6 είναι: δημιουργία τμημάτων και υπολογισμός με τον αριθμό accNum (σχήμα 5.1). Υποθέτουμε ότι η εκτέλεση μιας περίπτωσης δοκιμής ενάντια σε μερικά αποτελέσματα εφαρμογής στην ακολουθία μηνύματος καλεί: insertCard, sessionCreateMessage, getAccount, setCurrentAccount. Κατόπιν, το αποτέλεσμα ταιριάζει με το αναμενόμενο (το τμήμα δημιουργείται με το sessionCreateMessage, ο υπολογισμός τίθεται setCurrentAccount). Εντούτοις, αξιολογώντας Path6, και χρησιμοποιώντας τις σαφής σχέσεις από τους κόμβους στα σχετικά μηνύματά τους (στο πρότυπο uml), προσδιορίζουμε ότι μια κλήση στο validatePin (N9 στο πρότυπο tgh) λείπει.

Τέλος, δεδομένου ότι τα πρότυπα χρησιμοποιούνται ως βάση για την παραγωγή δοκιμών, η βασισμένη σε πρότυπο μέτρηση κάλυψης είναι μια άλλη δραστηριότητα που πρέπει να υποστηρίζεται από τις βασισμένες σε πρότυπο δοκιμαστικές προσεγγίσεις. Είναι επίσης μια άλλη ελπιδοφόρος εφαρμογή για την υποδομή που προτείνεται σε αυτό το έγγραφο. Στο σχήμα 5.1, οι περιπτώσεις δοκιμής κάτω από το startSession Root TGH δημιουργήθηκαν για να προκαλέσουν τις διαφορετικές πορείες από το διάγραμμα ακολουθίας startSession (uml πρότυπο, σχήμα 5.1). Όταν αυτές οι περιπτώσεις δοκιμής εκτελούνται, η κάλυψη που λαμβάνεται σε σχέση με τις σαφώς προσδιορισμένες οντότητες (π.χ. μηνύματα) στο αρχικό πρότυπο (uml πρότυπο) οριοθετείται με την υποστήριξη των σαφώς καθορισμένων σχέσεων.

5.4 Διαδικασία παραγωγής δοκιμής

Η δοκιμή οπισθοδρόμησης είναι μέρος των παραδοσιακών εξεταστικών δραστηριοτήτων, οι οποίες περιλαμβάνουν επίσης: παραγωγή περιπτώσεων δοκιμής, εκτέλεση δοκιμής, αξιολόγηση αποτελέσματος, και ανάλυση κάλυψης. Ενώ η αυτοματοποιημένη βασισμένη σε κώδικα εξέταση οπισθοδρόμησης είχε μελετηθεί κατά ένα μεγάλο μέρος και υποστηρίζεται από πολλά εργαλεία, υπάρχει μια έλλειψη αυτοματοποιημένων μελετών για τη δυνατότητα πραγματοποίησης της βασισμένης σε πρότυπα δοκιμής οπισθοδρόμησης. Στην πραγματικότητα, οι περισσότερες βασισμένες σε πρότυπα δοκιμές (MBT) αυτοματοποιημένες προσεγγίσεις επικεντρώνονται συχνά στις δραστηριότητες παραγωγής και εκτέλεσης περιπτώσεων δοκιμής, ενώ είναι διαθέσιμη περιορισμένη υποστήριξη για άλλες δραστηριότητες [Naslavsky, 2006]. Με την οδηγημένη με πρότυπα ανάπτυξη (MDD), οι αλλαγές εκτελούνται στα πρότυπα, έτσι το αποτέλεσμα της σύγκρισης των διαφορετικών προτύπων, σε αντιδιαστολή με τις διαφορετικές εφαρμογές, πρέπει να χρησιμοποιηθεί ως βάση για την επιλογή δοκιμής οπισθοδρόμησης περιπτώσεων δοκιμής. Συνεπώς, αυτή η ενότητα ερευνά την βασισμένη σε πρότυπα εκλεκτική δοκιμή οπισθοδρόμησης.

Σε αυτή την υποενότητα, περιγράφεται μια προσέγγιση για την βασισμένη σε πρότυπα δοκιμή που χρησιμοποιεί τεχνικές μετασχηματισμού μοντέλων [Jouault, 2005] για να δημιουργήσει μια υποδομή που περιλαμβάνει τα βασισμένα σε πρότυπα τεχνουργήματα προς δοκιμή και τις σαφής σχέσεις μεταξύ αυτών των προτύπων. Εκτός από την παραγωγή δοκιμής, η προτεινόμενη προσέγγιση υποστηρίζει την βασισμένη σε πρότυπα δοκιμή οπισθοδρόμησης με σαφής σχέσεις που καθιερώνονται κατά τη διάρκεια της διαδικασίας παραγωγής δοκιμής.

5.4.1. Υπόθεση

Η διαθεσιμότητα των αυτοματοποιημένων βασισμένων σε πρότυπα δοκιμών μπορεί να υποκινήσει τη χρήση των προτύπων και να επηρεάσει τη προσπάθεια που τίθεται και κατά τη διάρκεια της δημιουργίας τους. Όταν οι αλλαγές συμβαίνουν, η εκλεκτική δοκιμή οπισθοδρόμησης διαδραματίζει έναν σημαντικό ρόλο στη μείωση των πιθανοτήτων ότι νέα ελαττώματα θα παρεμβληθούν στο σύστημα. Λαμβάνοντας υπόψη τη σημασία τέτοιας δραστηριότητας, οι αυτοματοποιημένες προσεγγίσεις δοκιμών πρέπει να βελτιώσουν την υποστήριξή τους για την εκλεκτική δοκιμή οπισθοδρόμησης.

Οι σχέσεις που υποστηρίζουν την βασισμένη σε πρότυπα δοκιμή οπισθοδρόμησης, αφορούν μοντέλα περιπτώσεων δοκιμής που προέρχονται από αυτά τα πρότυπα, και προσδιορίζουν τις περιπτώσεις δοκιμής που είναι για επανάληψη. Το επίπεδο καταμερισμού των τελικών σημείων τους, επηρεάζει την ακρίβεια επιλεκτικότητας (οι σαφής σχέσεις αυξάνουν την ακρίβεια). Αυτή η μελέτη [Naslavsky, 2006] επιβάλλει την παρατήρηση [Dick and Faivre, 1993] εκείνων των σχέσεων που παίζουν σημαντικό ρόλο στην υποστήριξη των εξεταστικών δραστηριοτήτων. Μια βασική παρατήρηση σε αυτή την προσέγγισή είναι για τις πληροφορίες σχετικά με τις σχέσεις που χρησιμοποιήθηκαν για να υποστηρίξουν τη δοκιμή οπισθοδρόμησης οι οποίες ενσωματώνονται στη στρατηγική παραγωγής περιπτώσεων δοκιμής. Κατά συνέπεια, εάν προσδιορίστηκαν με σαφήνεια κατά τη διάρκεια της διαδικασίας παραγωγής περιπτώσεων δοκιμής, οι πληροφορίες ενσωματώνονται στη στρατηγική παραγωγής περιπτώσεων δοκιμής

ώστε να υποστηρίξουν και άλλες εξεταστικές δραστηριότητες. Επομένως, προτείνεται μια προσέγγιση βασισμένη στην υπόθεση ότι όταν οι σχέσεις μεταξύ των περιπτώσεων δοκιμής και των αρχικών προτύπων τους είναι σαφώς προσδιορισμένες, μπορούν να χρησιμοποιηθούν για να υποστηρίξουν αποτελεσματικά την βασισμένη σε πρότυπα εκλεκτική δοκιμή οπισθοδρόμησης.

Μια άλλη ματιά της προσέγγισής είναι ότι η παραγωγή περιπτώσεων δοκιμής μπορεί να υποστηριχθεί με τον μετασχηματισμό μοντέλων. Επιπλέον, οι πρότυπες λύσεις μετασχηματισμού εξετάζουν το πρόβλημα της ιχνηλασιμότητας με τη δημιουργία σχέσεων μεταξύ των μετασχηματισμένων τεχνουργημάτων ως τμήμα της διαδικασίας μετασχηματισμού [Jouault, 2005]. Αυτές οι λύσεις υποστηρίζουν επίσης την ιχνηλασιμότητα σε διαφορετικά επίπεδα.

Όσον αφορά τα πρότυπα συμπεριφοράς, και οι βασισμένες στο σενάριο και οι βασισμένες σε κατάστασεις γλώσσες προδιαγραφών, έχουν προκύψει ως σημαντικές προοπτικές. Ένα μεγάλο ποσό της ακαδημαϊκής έρευνας εξερευνά τη δοκιμή προδιαγραφών: οι γλώσσες τους έχουν συχνά έναν υψηλό βαθμό φορμαλισμού που μειώνει την πιθανότητα της υιοθέτησης τους από τους πρακτικούς όταν συγκρίνουν τα σενάρια. Στην πραγματικότητα, οι πρακτικοί είχαν υιοθετήσει τα σενάρια ως μέσα να εκφράσουν οι απαιτήσεις και τις προδιαγραφές συστημάτων [Weidenhaupt et al, 1998]. Διαγράμματα ακολουθίας UML είναι σενάρια στο επίπεδο σχεδίου αφαίρεσης, και για αυτόν τον λόγο, υιοθετούνται ως βάση για την εξεταστική αυτοματοποιημένη προσέγγιση.

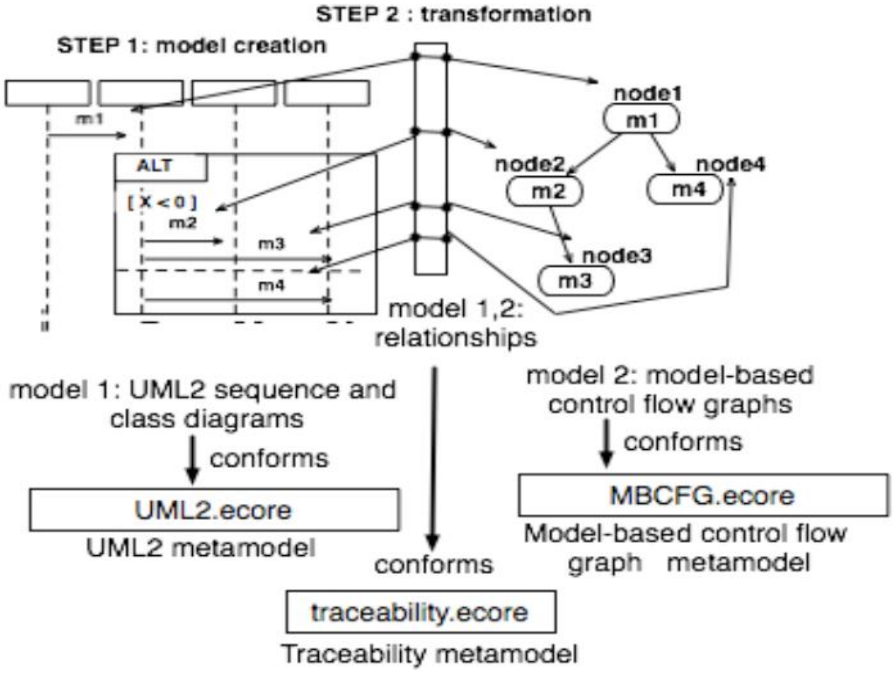
5.4.2 Επισκόπηση προσέγγισης

Αυτή η προσέγγιση παρέχει στους ελεγκτές μια εναλλακτική (βασισμένη σε πρότυπο) μέθοδο που υποστηρίζει την εκλεκτική δοκιμή οπισθοδρόμησης. Υποστηρίζει τεχνικές ιχνηλασιμότητας μετασχηματισμού μοντέλων για να δημιουργηθούν οι ακριβής σχέσεις μεταξύ των βασισμένων σε πρότυπο δοκιμαστικών τεχνουργημάτων. Χρησιμοποιεί αυτήν την υποδομή για να υποστηρίξει την βασισμένη σε πρότυπα δοκιμή οπισθοδρόμησης. Αυτή η προσέγγιση συγκρίνει τις διαφορετικές εκδόσεις των προτύπων σε αντιδιαστολή με τις διαφορετικές εκδόσεις της προκύπτουσας σύγκρισης εφαρμογών και χρήσης ως βάση για να επιλέξει τις περιπτώσεις δοκιμής για να είναι η οπισθοδρόμηση δοκιμασμένη. Αναμένεται ότι πρέπει να είναι τουλάχιστον τόσο οικονομικώς αποδοτικό όσο και οι διαθέσιμες (που εξετάζουν την οπισθοδρόμηση) βασισμένες σε κώδικα προσεγγίσεις [Orso, 2004] όταν γίνονται οι αλλαγές στον κώδικα για να απεικονίσουν τις αλλαγές στα πρότυπα. Αναμένεται επίσης ότι πρέπει να είναι οικονομικώς πιο αποδοτικό από τις βασισμένες σε πρότυπο προσεγγίσεις που εξετάζουν μόνο τις επιφανειακές σχέσεις [Briand et al., 2002].

Αυτή η υποενότητα είναι επάνω στις τεχνικές μετασχηματισμού μοντέλων για να επιτύχει η υποστήριξη για την βασισμένη σε πρότυπα παραγωγή δοκιμής και την ιχνηλασιμότητα στα τεχνουργήματα. Τα πρότυπα ιχνηλασιμότητας αποτελούνται από σαφής σχέσεις, και δημιουργούνται κατά τη διάρκεια της δοκιμής διαδικασίας παραγωγής.

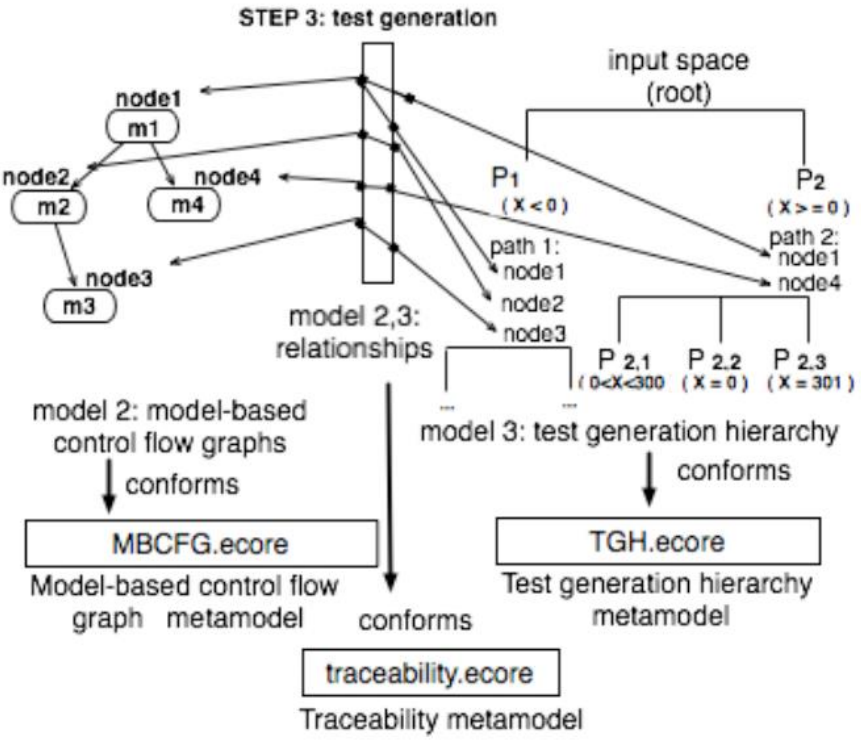
Τα σχήματα 5.2 και 5.3 παρουσιάζουν μια επισκόπηση της προσέγγισης. Παράγει τελικά μια ιεραρχία περιπτώσεων δοκιμής [Stocks and Carrington, 1996] από τα διαγράμματα ακολουθίας και κλάσης που ανήκουν σε ένα πρότυπο UML 2.0. Κατ' αρχάς, το βήμα 1 στο σχήμα 5.2 δημιουργεί ένα πρότυπο UML 2.0. Κατόπιν, το βήμα 2 μετασχηματίζει κάθε διάγραμμα ακολουθίας σε ένα βασισμένο σε πρότυπο μοντέλο γραφικών παραστάσεων ελέγχου ροής (mbcfgV1) και το UML πρότυπο (umlV1). Κατά τη διάρκεια αυτού του μετασχηματισμού, οι

πληροφορίες ιχνηλασιμότητας γίνονται σαφής σε ένα χωριστό πρότυπο ιχνηλασιμότητας με σαφής σχέσεις (Traceability metamodel) (βήμα 2).



Σχήμα 5.2 : Επισκόπηση προσέγγισης: βήματα 1 και 2.

Κατόπιν, πραγματοποιείται η παραγωγή δοκιμής (βήμα 3 στο σχήμα 5.3). Αποτελείται από τη χρήση και του UML 2.0 πρότυπου και της βασισμένης σε πρότυπο γραφικής παράστασης ελέγχου ροής για να παραγάγει μια ιεραρχία περιπτώσεων δοκιμής. Πάλι, κατά τη διάρκεια της παραγωγής δοκιμής στοιχείων, οι πληροφορίες ιχνηλασιμότητας γίνονται σαφής και ακριβής (βήμα 3). Το πρώτο επίπεδο της ιεραρχίας tghV1 περιγράφει τις πιθανές πορείες για κάθε διάγραμμα ακολουθίας στο πρότυπο UML. Κατόπιν, οι συγκεκριμένες τιμές περιπτώσεων δοκιμής παράγονται, αυτές είναι τιμές που ορίζονται στις παραμέτρους ενός διαγράμματος ακολουθίας, και δημιουργούν το κάθε μονοπάτι. Αυτός ο στόχος εξετάζει κάθε πιθανή πορεία στην ιεραρχία, και προσδιορίζει τους περιορισμούς στις παραμέτρους του διαγράμματος ακολουθίας, που είναι υπεύθυνες για να δημιουργήσουν την κάθε πορεία. Κατόπιν, εφαρμόζει τις στρατηγικές επιλογής στοιχείων για να χωρίσει τις περιοχές εισαγωγής των παραμέτρων. Μια λύση που ικανοποιεί τον σύνδεσμο αυτών των περιορισμών είναι ένα δεδομένο εισόδου περίπτωσης δοκιμής.



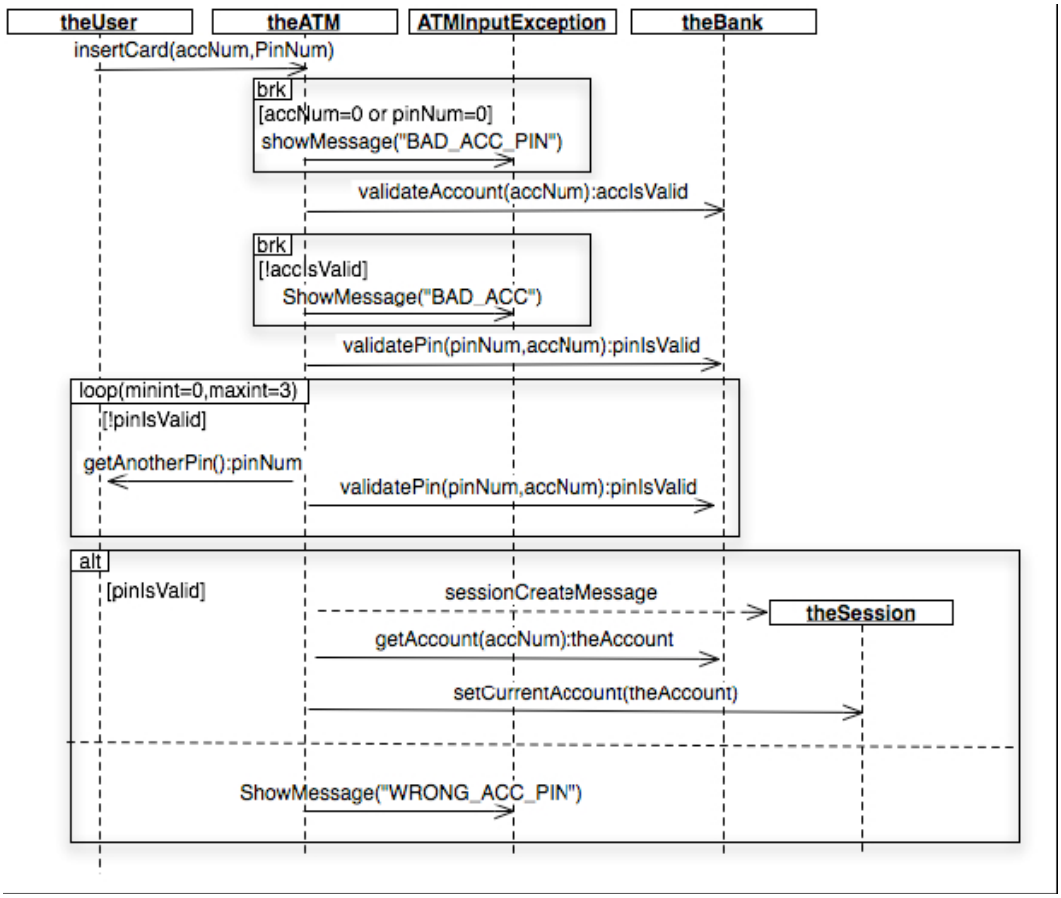
Σχήμα 5.3 : Επισκόπηση προσέγγισης: βήμα 3.

Οι ακόλουθες υποενότητες περιγράφουν, λεπτομερέστερα, τα βήματα και τα κύρια συστατικά της προσέγγισής μας.

5.5 Παράδειγμα ATM

Για να πειραματιστούμε με αυτήν την ιδέα, παρατηρούμε τα διαγράμματά κλάσης και ακολουθίας ενός πρότυπου UML Παραδείγματος ATM. Σε αυτήν την υποενότητα, επιδεικνύεται πώς η προσέγγιση λειτουργεί βαθμιαία.

Το σχήμα 5.4 απεικονίζει το διάγραμμα ακολουθίας «φάση έναρξης- start session» . Ο πίνακας 5.1 παρουσιάζει πρόσθετους περιορισμούς σχετικούς με το διάγραμμα ακολουθίας. Ο C1 και C2 είναι περιορισμοί στις παραμέτρους του διαγράμματος ακολουθίας, το accNum, που είναι ο αριθμός λογαριασμού, και το pinNum, που είναι ο μυστικός αριθμός της κάρτας. Ανάλογα με τις τιμές αυτών των μεταβλητών, προκαλούνται διαφορετικές πορείες στο διάγραμμα ακολουθίας. Για παράδειγμα, εάν είτε το accNum είτε το pinNum είναι ίσο με μηδέν, η πρώτη έξοδος (πρώτο brk στο σχήμα 5.4) γίνεται και ένα μήνυμα λάθους παρουσιάζεται. Διαφορετικά, εάν και οι δύο είναι διαφορετικοί από μηδέν, και ισχύουν, ο πρώτος τελεστής εναλλαγής (ALT στο σχήμα 5.4) ενεργοποιείται. Κατόπιν, μια περίοδος δημιουργείται, το αντικείμενο του λογαριασμού ανακτάται από την τράπεζα, και τίθεται ο τρέχων λογαριασμός για τη δημιουργημένη περίοδο.



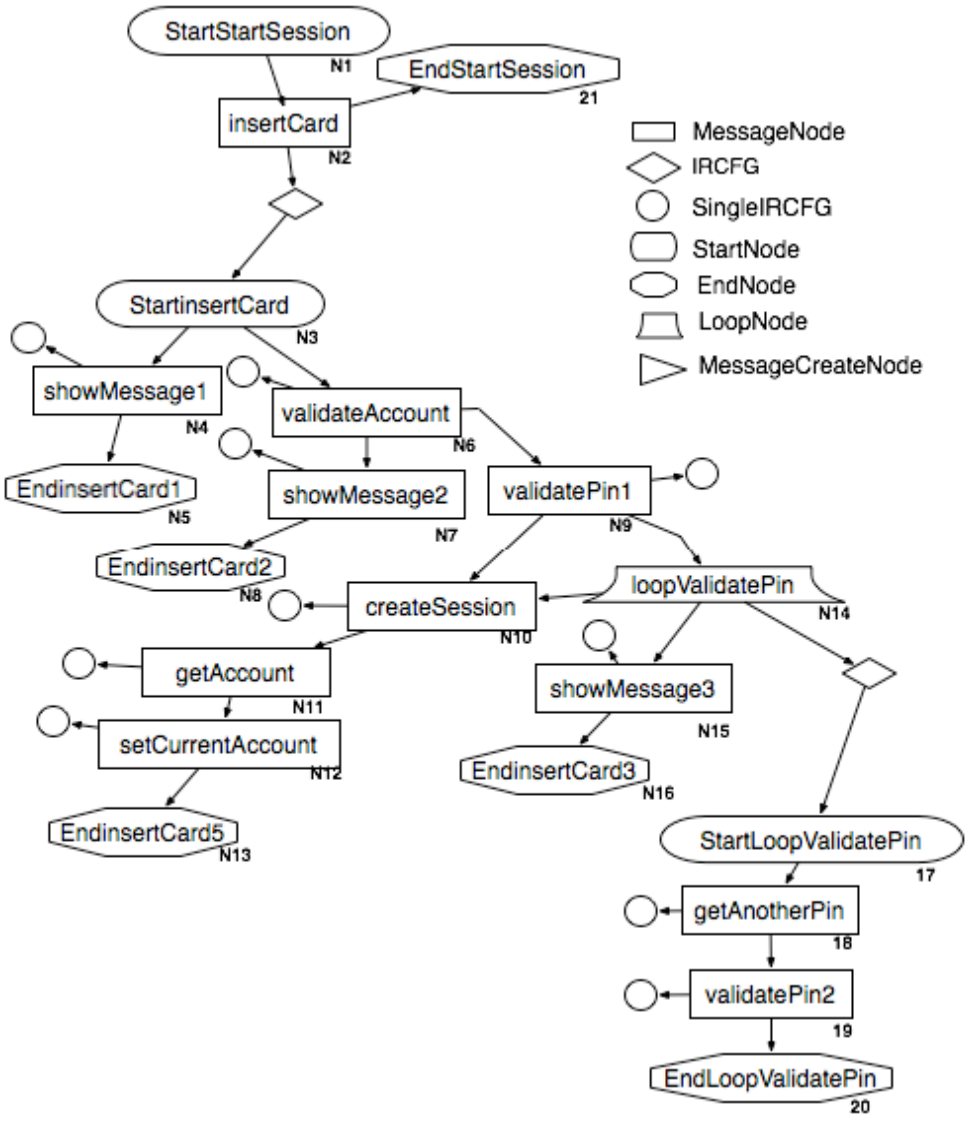
Σχήμα 5.4 : Διάγραμμα ακολουθίας <φάση έναρξης>.

Η δημιουργία προτύπου περιγράφηκε ως βήμα 1 της προσέγγισης (σχήμα 5.2). Το ακόλουθο βήμα μετασχηματίζει τα διαγράμματα ακολουθίας από το πρότυπο UML στις βασισμένες σε πρότυπο γραφικές παραστάσεις ροής ελέγχου, και τη δημιουργία των πληροφοριών ιχνηλασιμότητας (βήμα 2 στο σχήμα 5.2). Η γραφική αντιπροσώπευση του μετασχηματισμένου προτύπου απεικονίζεται στο σχήμα 5.5, ενώ ένα μέρος των πληροφοριών ιχνηλασιμότητας απεικονίζεται στο σχήμα 5.6.

C1: on accNum domain	0<accNum<9999
C2: on pinNum domain	0 < pinNum<999
C3: on validateAccount (accNum)	context Bank post: if self.accounts-> select(accountNumber = accNum)-> notEmpty() then result= true else result= false endif
C4: on validatePin (pinNum,accNum)	context Bank post: if self.accounts ->select (accountNumber = accNum and pinNumber=pinNum) -> notEmpty() then result= true else result= false endif

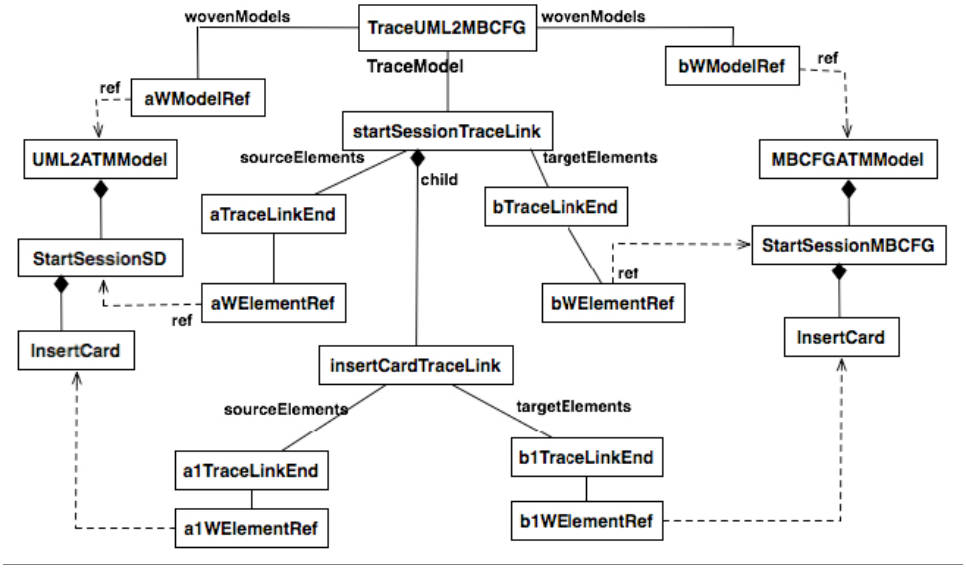
Πίνακας 5.1: Πρόσθετοι περιορισμοί που αφορούν τα στοιχεία της <φάση έναρξης>.

Το επόμενο και τελευταίο βήμα στην προσέγγιση (βήμα 3 στο σχήμα 5.3) είναι δημιουργία της ιεραρχίας παραγωγής δοκιμής, και οι πληροφορίες ιχνηλασιμότητας. Χρησιμοποιούμε τις πληροφορίες από το διάγραμμα ακολουθίας «φάση έναρξης» (σχήμα 5.4, αποκαλούμενο από τώρα και στο εξής ssSD), από τους πρόσθετους περιορισμούς, από την γραφική παράσταση ροής ελέγχου βασισμένη σε πρότυπα «φάση έναρξης» (σχήμα 5.5, αποκαλούμενο από τώρα και στο εξής ssMBCFG), και από το πρότυπο ιχνηλασιμότητας.



Σχήμα 5.5 : Βασισμένη σε πρότυπα γραφική παράσταση ροής ελέγχου για τη <φάση έναρξης>.

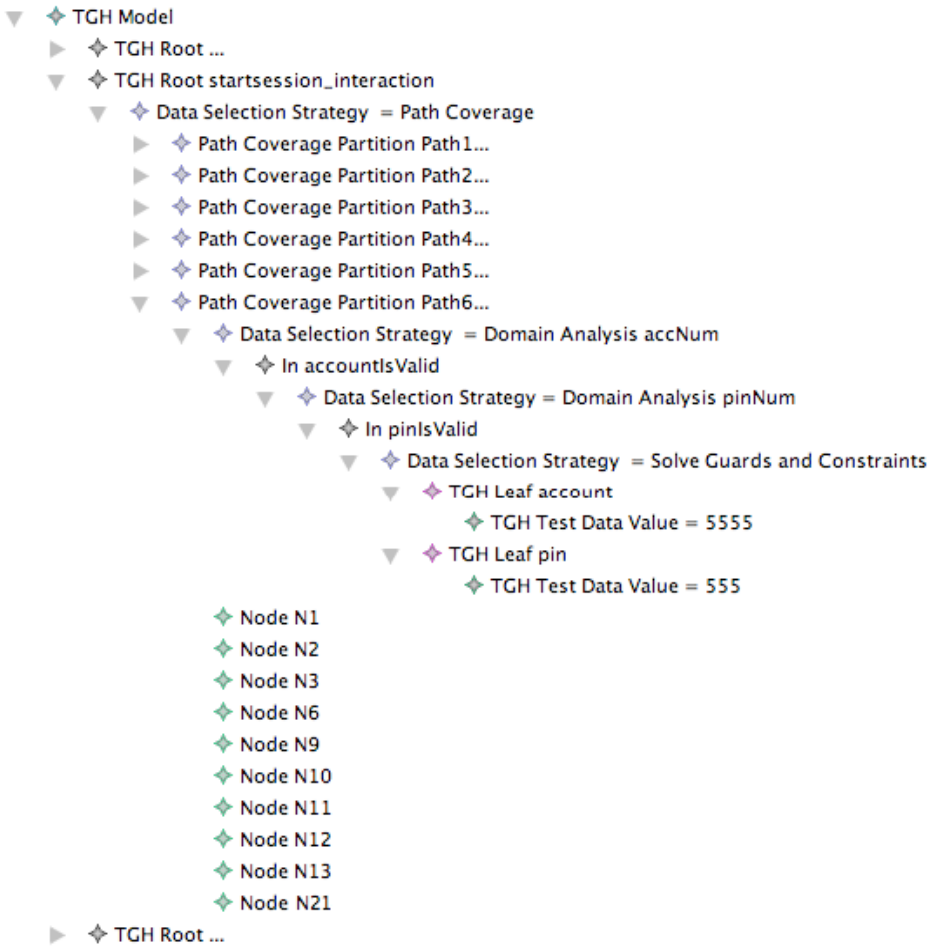
Στο σχήμα 5.6 , το πρότυπο ιχνηλασιμότητας ονομάζεται TraceUML2MBCFG. Αφορά τα πρότυπα UML2ATMModel και MBCFGATMModel. Ονομάζει μια σύνδεση ιχνών startSessionTraceLink που αφορά το StartSessionSD στο UML2ATMModel και το StartSessionMBCFG στο MBCDFATMModel. Το startSessionTraceLink έχει πολλά παιδιά. Το σχήμα 5.6 παρουσιάζει insertCardTraceLink, το οποίο συσχετίζει με το μήνυμα InsertCard στο StartSessionSD με τον InsertCard κόμβο του StartSessionMBCFG.



Σχήμα 5.6: Ένα μικρό μέρος του προτύπου ιχνηλασιμότητας.

Εφαρμόζοντας τη στρατηγική επιλογής πορειών στο ssMBCFG, παράγονται πιθανές πορείες (π.χ. N1-N2-N3-N9-N10-N11-N12 -N21). Χρησιμοποιώντας το πρότυπο ιχνηλασιμότητας και το ssSD, είναι γνωστό ότι αυτή η πορεία επιτρέπεται υπό τον όρο ότι μερικοί φρουροί και περιορισμοί την ικανοποιούν. Κατ' αρχάς, το accNum και το pinNum πρέπει να είναι διαφορετικά από μηδέν (φρουρός για πρώτο brk στο ssSD). Δεύτερον, το accNum πρέπει να είναι ο αριθμός ενός έγκυρου λογαριασμού (φρουρός για το δεύτερο brk στο ssDD). Τρίτον, το pinNum πρέπει να είναι έγκυρο για τον λογαριασμό (φρουρός για το βρόχο και για το ALT στο ssSD). Έπειτα αναλύονται οι περιοχές του accNum και του pinNum. Το C3 στον πίνακα 5.1 λέει ότι ο validateAccount επιστρέφει αλήθεια όταν ανήκει το accNum σε ένα σύνολο έγκυρων λογαριασμών. Το C4 στον πίνακα 5.1 λέει ότι το validatePin (accNum, pinNum) επιστρέφει αλήθεια εάν υπάρχει λογαριασμός και μυστικός αριθμός στο σύνολο προκαθορισμένων έγκυρων λογαριασμών, όπου ο λογαριασμός και ο μυστικός αριθμός ταιριάζουν με το accNum και το pinNum που παραλαμβάνονται ως εισαγωγή από τον χρήστη. Επομένως, αυτό που απαιτείται ως εισαγωγή για να εξεταστεί αυτή η πορεία είναι να ταιριάζει ο συνδυασμός έγκυρου μυστικού αριθμού και λογαριασμού. Αυτός ο συνδυασμός των περιορισμών είναι αρκετά απλός διότι μια λύση μπορεί να ανακτηθεί τυχαία άμεσα από το σύνολο έγκυρων λογαριασμών, η οποία είτε καθορίζεται στο πρότυπο, είτε σε μια χωριστή βάση δεδομένων. Σε περίπτωση πιο σύνθετου συνδυασμού, μπορεί να μετασχηματιστεί μια έγκυρη εισαγωγή λύσεων περιορισμού που θα δημιουργούσε μία ή περισσότερες λύσεις. Αυτό το χαρακτηριστικό γνώρισμα δεν εφαρμόζεται

ακόμα. Το σχήμα 5.7 απεικονίζει την ιεραρχία παραγωγής δοκιμής που δημιουργείται με αυτήν την διαδικασία.

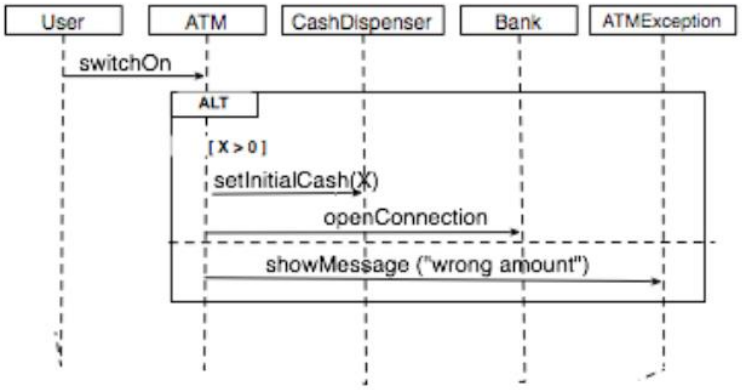


Σχήμα 5.7: Παράδειγμα ιεραρχίας παραγωγής δοκιμής.

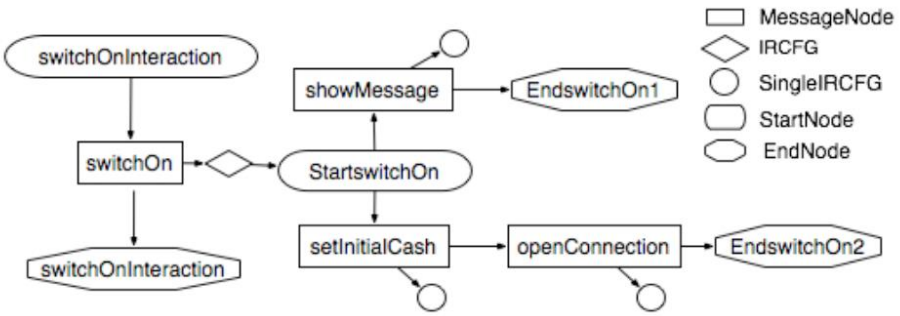
Ο κύριος στόχος της προσέγγισής είναι να ερευνηθούν οι σχέσεις μεταξύ των τεχνουργημάτων δοκιμής. Η βασισμένη σε πρότυπο γραφική παράσταση ροής ελέγχου χρησιμοποιείται για να υποστηρίξει την παραγωγή πορειών από τα διαγράμματα ακολουθίας, και για να καθιερώσει τις σχέσεις από τα διαγράμματα ακολουθίας στην ιεραρχία παραγωγής δοκιμής. Δεν εξετάζεται η παραγωγή δοκιμής από ταυτόχρονα πρότυπα. Επομένως, χρησιμοποιείται μια απλούστερη βασισμένη σε πρότυπο ροή ελέγχου μοντέλου, η όποια είναι μια διαδικαστικά περιορισμένη γραφική παράσταση ροής ελέγχου (IRCFG - inter-procedural restricted control flow graph). «Αντιπροσωπεύει κατά τρόπο συμπαγή το σύνολο ακολουθιών μηνυμάτων σε ένα διάγραμμα ακολουθίας» [Rountev et al., 2005]. Αυτή η ενότητα, εντούτοις, δεν καθορίζει ένα μοντέλο για τη γραφική παράσταση ροής ελέγχου και δεν υποστηρίζει τη χρήση βρόχων.

Το σχήμα 5.9 απεικονίζει μια περίπτωση μιας ροής ελέγχου βασισμένης σε πρότυπο. Το πρότυπο προήλθε από το διάγραμμα ακολουθίας στο σχήμα 5.8. Ο πρώτος κόμβος στη γραφική παράσταση (switchOnInteraction StartNode) αντιπροσωπεύει την έναρξη ολόκληρου του διαγράμματος ακολουθίας στο σχήμα 5.8. Ακολουθείται από τον switchOn MessageNode, που αναπαριστά την κλήση του μηνύματος switchOn του διαγράμματος ακολουθίας. Σύμφωνα με το

μεταμοντέλο, ο MessageNode μπορεί να συσχετιστεί και με άλλο IRCFG, έτσι λοιπόν, ο switchOn MessageNode έχει ένα συσχετιζόμενο IRCFG, το οποίο αρχίζει με το StartswitchOn StartNode. Αυτός ο κόμβος ακολουθείται είτε από (1) το showMessage MessageNode, το οποίο ακολουθείται από το EndswitchOn1 EndNode, και έπειτα από switchOnInteraction EndNode που ολοκληρώνει το διάγραμμα ακολουθίας, ή από (2) το setInitialCash MessageNode, έπειτα το openConnection MessageNode, έπειτα το EndswitchOn2, επίσης ακολουθούμενο από το switchOnInteraction EndNode.



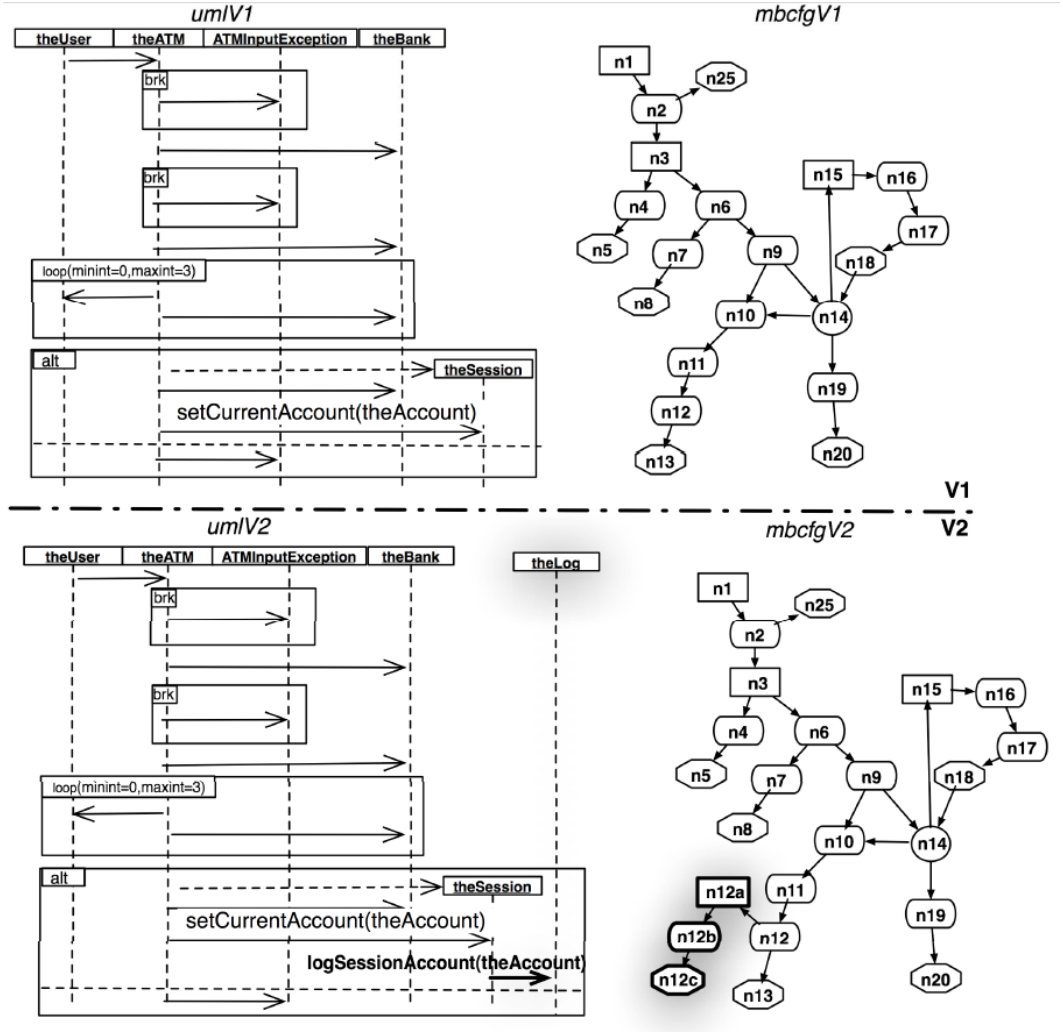
Σχήμα 5.8 : Διάγραμμα ακολουθίας.



Σχήμα 5.9 : Βασισμένη σε πρότυπο γραφική παράσταση ροής ελέγχου.

5.5.1 Προσθήκη μηνύματος στο παράδειγμα ATM

Σε αυτή την υποενότητα, παρουσιάζονται τα αποτελέσματα μιας μετατροπής της βασισμένης σε πρότυπο δοκιμής οπισθοδρόμησης του παραδείγματος του ATM.



Σχήμα 5.10: Διάγραμμα «σύνοδος ακολουθίας έναρξης».

Το σχήμα 5.10 απεικονίζει δύο εκδόσεις ενός διαγράμματος ακολουθίας (umlV1 και umlV2) με τις αντίστοιχες βασισμένες σε πρότυπο γραφικές παραστάσεις ελέγχου ροής (mbcfgV1 και mbcfgV2). Από umlV1 και mbcfgV1, παρήχθησαν 6 πορείες και 37 περιπτώσεις δοκιμής. Στο umlV2, προστέθηκε ένα νέο μήνυμα (logSessionAccount). Συγκρίνοντας τα mbcfgV1 και mbcfgV2, και χρησιμοποιώντας το πρότυπο που προέκυψε για να εντοπιστούν οι διαφορές (κόμβοι n12a, n12b, n12c οι οποίοι είναι τονισμένοι στο mbcfgV2).

Αυτή η αλλαγή τροποποιεί τις πορείες που περιέχουν τον κόμβο n12, και επομένως, θα προσκρούσει στις περιπτώσεις δοκιμής, στο πρότυπο ιεραρχίας παραγωγής δοκιμής, με το συγκεκριμένο σύνολο πορειών. Χρησιμοποιήσαμε το πρότυπο ιχνηλασιμότητας (βήμα 3 του σχήματος 5.3) για να εντοπίσουμε τον κόμβο n12 στο πρότυπο ιεραρχίας παραγωγής δοκιμής. Αυτός ο κόμβος είναι κάτω από τις πορείες #1 (n1, n2, n3, n6, n9, n11, n12, n14, n25) και #5 (n1, n2, n3, n6, n9, n11, n12, n14, n15-n18, n15-n18, n15-n18, n25). Επομένως, οι υποθέσεις κάτω από εκείνες τις δύο πορείες στην ιεραρχία πρέπει να έχουν δοκιμασμένη οπισθοδρόμηση.

Αυτές οι πληροφορίες θα μπορούσαν να αποκτηθούν επειδή χρησιμοποιήθηκαν οι σαφής σχέσεις μεταξύ των μερών των περιπτώσεων δοκιμής και των κόμβων που επηρέασαν τη δημιουργία αυτών των μερών. Εάν οι σαφής σχέσεις δεν ήταν ακριβής όπως είναι στην προσέγγισή, θα ενσωματώνονταν στον αλγόριθμο παραγωγής δοκιμής. Κατά συνέπεια, ο

προσδιορισμός ενός ακριβούς υποσυνόλου των περιπτώσεων δοκιμής για να είναι η οπισθοδρόμηση δοκιμασμένη, θα ήταν ένας στόχος με εντατικότερη εργασία. Επιπλέον, λαμβάνοντας υπόψη την ίδια αλλαγή και τις μη σαφής σχέσεις μεταξύ διαγραμμάτων ακολουθίας και των παραγόμενων περιπτώσεων δοκιμής, θα έπρεπε να προσδιοριστούν και οι 37 περιπτώσεις δοκιμής για να υπάρχει δοκιμασμένη οπισθοδρόμηση.

5.5.2 Βασισμένη σε πρότυπο δοκιμή οπισθοδρόμησης

Η προσέγγισή δημιουργεί μια υποδομή που αποτελείται από τα δοκιμαστικά πρότυπα (πρότυπα umlV1, mbcfgV1, και tghV1), και τις σαφώς προσδιορισμένες σχέσεις-ίχνη (βήμα 2 του σχήματος 5.2 και βήμα 3 του σχήματος 5.3). Αυτή η υποδομή υποστηρίζει τη δοκιμή για την βασισμένη σε πρότυπο οπισθοδρόμηση .

Λειτουργεί ως εξής:

(1) Σύγκριση δύο εκδόσεων ενός προτύπου UML (umlV1 και umlV2), και παραγωγή του προτύπου που περιγράφει τις αλλαγές (diffUmls)

(2) Προσδιορισμός του υποσυνόλου των στοιχείων του βασισμένου σε πρότυπο μοντέλο γραφικών παραστάσεων ροής ελέγχου (mbcfgV1) ανεπηρέαστο από τις αλλαγές

(3) Δημιουργία της νέας έκδοσης της βασισμένης σε πρότυπο γραφικής παράστασης ροής ελέγχου (mbcfgV2)

(4) Σύγκριση των δύο εκδόσεων των βασισμένων σε πρότυπο, μοντέλων γραφικών παραστάσεων ροής ελέγχου (mbcfgV1 και mbcfgV2), και παραγωγή του προτύπου που περιγράφει τις αλλαγές (diffMbcfgs)

(5) Επιλογή περιπτώσεων δοκιμής.

Η λειτουργία (2) μειώνει την προσπάθεια του μετασχηματισμού UML σε πρότυπα γραφικών παραστάσεων ελέγχου ροής επειδή αποφεύγει τα μη-προσκρουμένα στοιχεία. Η λειτουργία (3) παράγεται από: (α) στοιχεία από τα μη προσκρουόμενα υποσύνολα (mbcfgV1), και (β) νέα στοιχεία που απαιτούνται για να εξετάσουν τις αλλαγές στο πρότυπο UML. Η λειτουργία (5) χρησιμοποιεί το πρότυπο ιχνηλασιμότητας (βήμα 3 του σχήματος 5.3), για να προσδιορίσει το πρότυπο παραγωγής δοκιμής πορειών ιεραρχικά (tghV1). Το πρότυπο επηρεάζεται από τροποποιήσεις των βασισμένων σε πρότυπο γραφικών παραστάσεων ελέγχου ροής. Οι τιμές περιπτώσεων δοκιμής κάτω από μια πορεία στην ιεραρχία παραγωγής δοκιμής προκύπτουν από το χωρισμό του διαστήματος εισαγωγής των παραμέτρων που προκαλούν κάθε πορεία. Επομένως, οι περιπτώσεις δοκιμής για τις προσκρουόμενες πορείες πρέπει να έχουν δοκιμασμένη οπισθοδρόμηση.

Θα θεωρήσουμε την προσέγγισή μας επιτυχή εάν (1) μειώσει την ακολουθία δοκιμών, (2) διατηρήσει ή μειώσει το κόστος του υποσύνολου ακολουθιών δοκιμής, και συγχρόνως (3) διατηρήσει (ή αυξήσει) την αποτελεσματικότητα ανίχνευσης ελαττωμάτων.

6. Σχετικές εργασίες

Οι Ramesh και Jarke [Ramesh and Jarke, 2001] έχουν ένα ευρύ όραμα για τις πληροφορίες που απαιτούνται στην ιχνηλασιμότητα απαιτήσεων. Η μελέτη τους είναι βασισμένη στην ανάλυση βιομηχανικού προγράμματος ανάπτυξης λογισμικού. Χωρίζουν σε δύο τμήματα τους χρήστες ιχνηλασιμότητας και προτείνουν δύο αντίστοιχα μεταμοντέλα ιχνηλασιμότητας (το ένα είναι μια απλοποίηση του άλλου). Το πιο ολοκληρωμένο μεταμοντέλο έχει 31 τύπους οντοτήτων (μετακλάσεις στο μεταμοντέλο) και περίπου 50 τύπους συνδέσμων. Επιπλέον στην πολυπλοκότητα συνδέεται η ποικιλομορφία των τύπων οντοτήτων και συνδέσμων, ένας ακριβής καθορισμός για τα στοιχεία δεν παρέχεται, και καθιστά την εφαρμογή του μεταμοντέλου δύσκολο στόχο. Περιέργως, η ανάλυση και οι προδιαγραφές σχεδίου αντιπροσωπεύονται μόνο από έναν τύπο οντότητας που ονομάζεται «system_subsystem_component», δηλαδή δεν υπάρχει περισσότερη λεπτομέρεια ή σύνδεση με οποιοδήποτε πρότυπο. Στην εργασία τους ο

μόνος προτεινόμενος μηχανισμός για να διαμορφώσει το μεταμοντέλο σύμφωνα με τις ανάγκες προγράμματος είναι να κοπούν ή να προστεθούν μέρη του μεταμοντέλου. Με βάση την ανάλυση της βιομηχανικής ανάπτυξης λογισμικού των εργασιών των Ramesh και Jarke [Ramesh and Jarke, 2001] καθορίζονται δύο μοντέλα για την ιχνηλασιμότητα. Οι συντάκτες διαφοροποιούν τους χρήστες της ιχνηλασιμότητας σε low- end και high-end χρήστες . Αντίστοιχα παρέχουν μια απλουστευμένη και πλήρη έκδοση του μοντέλου τους. Επιπλέον, καθιερώνουν ένα προκαθορισμένα τυποποιημένο σύνολο τύπων συνδέσμων. Οι συντάκτες στρέφονται ειδικά στη διαχείριση του προγράμματος και στις οργανωτικές ανάγκες της ιχνηλασιμότητας. Δεν λύνουν το πρόβλημα πώς ιχνηλασιμότητα μπορεί να καθιερωθεί στην ανάλυση και το σχέδιο.

Ο Letelier [Letelier, 2002] προσφέρει ένα μοντέλο για τις απαιτήσεις ιχνηλασιμότητας στα προγράμματα βασισμένα στη UML . Δίνει ένα παράδειγμα από τη χρήση σε ένα UP πρόγραμμα. Ο συντάκτης στρέφεται σε ένα γενικό πρότυπο ιχνηλασιμότητας και δίνει συμβουλές στο πώς να το προσαρμόσει χρησιμοποιώντας μηχανισμούς UML. Κρατώντας το πρότυπο γενικά χρησιμοποιήσιμο, δεν είναι δυνατό να καθορίσει τους κανόνες και τις δραστηριότητες για τη δημιουργία, την επαλήθευση και την αναπροσαρμογή των συνδέσμων, οι οποίοι θα μπορούσαν να εκτελεσθούν (ημι)αυτόματα από ένα εργαλείο.

Οι Toranzo και Castro [Toranzo and Castro, 1999] παρουσιάζουν ένα μεταμοντέλο ιχνηλασιμότητας που καθορίζεται από πολλές απόψεις, κάθε μια από αυτές συνδέεται με ένα συγκεκριμένο είδος χρήστη για τις πληροφορίες ιχνηλασιμότητας (διευθυντής προγράμματος, μηχανικός απαιτήσεων ή μηχανικός λογισμικού). Στην εργασία τους δεν υπάρχει μηχανισμός διαμόρφωσης για να προσαρμόσει την ιχνηλασιμότητα στις ανάγκες της εφαρμογής. Επιπλέον, το επίπεδο λεπτομέρειας των τεχνουργημάτων είναι υπερβολικό, ασχολείται με έγγραφα και διαγράμματα.

Οι Spence και Probasco [Spence and Probasco , 2000] συζητούν διάφορες εναλλακτικές λύσεις για την ιχνηλασιμότητα μεταξύ των απαιτήσεων. Το έγγραφο τους αφορά τη UP. Οι συντάκτες δεν δίνουν την απάντηση στο πώς πρέπει να επισημανθεί η μετάβαση στην ανάλυση, στο σχέδιο και στα ακόλουθα βήματα ανάπτυξης. Οι Spence και Probasco [Spence and Probasco, 1998] παρουσιάζουν διάφορες στρατηγικές για την ιχνηλασιμότητα όταν χρησιμοποιείται μια προσανατολισμένη προς την περίπτωση χρήσης διαδικασία (όπως η περίπτωση RUP). Κάθε στρατηγική περιγράφεται με ένα απλό μεταμοντέλο ιχνηλασιμότητας, με τύπους τεχνουργημάτων και συνδέσμων. Όλες οι προτεινόμενες στρατηγικές εξετάζουν μόνο συνδέσμους μεταξύ των τεχνουργημάτων που είναι κοντά σε απαιτήσεις (ανάγκες των χρηστών, χαρακτηριστικά γνωρίσματα λογισμικού, περιπτώσεις χρήσης, κ.λπ.). Η σύνδεση με τα τεχνουργήματα για μοντελοποίηση και δοκιμή αφήνεται να υπονοηθεί σύμφωνα με αυτά που καθιερώνει μια διαδικασία προσανατολισμένη προς την περίπτωση χρήσης (περιπτωσιολογική ανάλυση χρήσης ή πραγματοποίηση σχεδίου, λειτουργική δοκιμή για κάθε περίπτωση χρήσης, κ.λπ.). Επιπλέον, μοναδικός τύπος συνδέσμου που χρησιμοποιούν είναι το ισοδύναμο traceTo. Ομοίως, ο Leite και άλλοι [Leite and Oliveira, 1995, Leite et al., 1997] παρέχουν ένα πλαίσιο για την απόκτηση και την οργάνωση των απαιτήσεων που εκφράζεται σε φυσική γλώσσα. Εγκαθιστούν τους συνδέσμους ιχνηλασιμότητας μεταξύ των απαιτήσεων αλλά δεν περιλαμβάνουν την ιχνηλασιμότητα σε άλλα επόμενα τεχνουργήματα.

7. Συμπεράσματα

Η ιχνηλασιμότητα απαιτήσεων είναι το κλειδί για την επιτυχία της διαδικασίας διαχείρισης απαιτήσεων. Εντούτοις, δεν υπάρχει κανένα συμπέρασμα για την καταλληλότερη στρατηγική ώστε να εκτελεστεί πιο αποτελεσματικά η ιχνηλασιμότητα απαιτήσεων. Κατά συνέπεια, στην πράξη, η ιχνηλασιμότητα απαιτήσεων παρουσιάζει διαφορετικά επίπεδα ικανοποίησης και αποδοχής στις μελέτες ανάπτυξης λογισμικού. Συνεπώς, δεν υπάρχει καταλληλότερη υποστήριξη εργαλείων για αυτή τη δραστηριότητα [Letelier , 2002]. Ιστορικά, η έρευνα για τη διαχείριση απαιτήσεων, εκτελεί παράλληλα έρευνα για τη διαμόρφωση και ανάπτυξη λογισμικού, που αποτελείται από δύο διαφορετικές ενότητες: μια ενότητα που στρέφεται στη βελτίωση της επεξεργασίας των απαιτήσεων των προδιαγραφών χωρίς να αναλύει περαιτέρω τις σημειώσεις διαμόρφωσης λογισμικού ή τη μεθοδολογία, και μια άλλη που επικεντρώνεται στη βελτίωση των τεχνικών ανάλυσης και σχεδίου αλλά χωρίς να δίνει πολλή προσοχή στις προδιαγραφές απαιτήσεων. Σήμερα, η UML, η δημοφιλέστερη αντικειμενοστραφής γλώσσα ανάλυσης λογισμικού, αντιπροσωπεύει μια άριστη ευκαιρία για να εκμεταλλευτούν τα αποτελέσματα και από τις δύο ενότητες. Η UML μπορεί να χρησιμοποιηθεί για να καθορίσει ένα κοινό πλαίσιο στις απαιτήσεις υποστήριξης και τις προδιαγραφές διαμόρφωσης.

Η κύρια συμβολή αυτής της εργασίας είναι μια υποδομή και μια ροή δουλειάς για την διερεύνηση των τρόπων εξέτασης της ιχνηλασιμότητας στα τεχνουργήματα, από την σκοπιά των τεχνικών ελέγχου λογισμικού.

Σε αυτό το έγγραφο, παρουσιάστηκε μια προσέγγιση που επεξεργάζεται τις διαθέσιμες τεχνικές ιχνηλασιμότητας μετασχηματισμού προτύπων για να δημιουργηθούν οι σαφώς καθορισμένες σχέσεις μεταξύ των βασισμένων σε πρότυπο τεχνουργημάτων δοκιμής. Οι σχέσεις δημιουργούνται με τη βοήθεια διαγραμμάτων ακολουθίας σε UML πρότυπα, με τη βασισμένη σε πρότυπα γραφική παράσταση ελέγχου ροής τους, και μια ιεραρχία παραγωγής δοκιμών που περιγράφει τις σχέσεις γονέων/παιδιών μέσω των διαγραμμάτων ακολουθίας [Naslavsky et al., 2007]. Οι σχέσεις μεταξύ του προτύπου UML και των βασισμένων σε πρότυπα γραφικών παραστάσεων ελέγχου ροής γίνονται ακριβής και διασαφηνίζονται κατά τη διάρκεια του μετασχηματισμού από το ένα πρότυπο σε άλλο, ενώ οι σχέσεις μεταξύ της βασισμένης σε πρότυπα γραφικής παράστασης ελέγχου ροής και της ιεραρχίας παραγωγής δοκιμών διασαφηνίζονται κατά τη διάρκεια παραγωγής της εισαγωγής δοκιμαστικών δεδομένων (δημιουργία της ιεραρχίας παραγωγής δοκιμών από την βασισμένη σε πρότυπα γραφική παράσταση ελέγχου ροής). Ο σαφής προσδιορισμός των σχέσεων βοηθάει την υποστήριξη για την βασισμένη σε πρότυπα αξιολόγηση αποτελέσματος των δοκιμών, την ανάλυση κάλυψης και τη δοκιμή οπισθοδρόμησης.

Η συγκεκριμένη προσέγγισή αφορά την παραγωγή δοκιμών και υποθέτει την ύπαρξη διαγραμμάτων ακολουθίας στο πρότυπο UML. Δημιουργεί μια βασισμένη σε πρότυπα γραφική παράσταση ελέγχου ροής από κάθε διάγραμμα ακολουθίας, και δημιουργεί στρατηγική κάλυψης πορειών που ακολουθείται από την ανάλυση περιοχών που βασίζεται σε παραμέτρους του διαγράμματος ακολουθίας. Για να λυθούν οι σύνδεσμοι περιορισμών που καθορίζουν διαφορετικές πορείες στην βασισμένη σε πρότυπα γραφική παράσταση ελέγχου ροής, οι σύνδεσμοι μετατρέπονται σε εισαγωγές λύσεων περιορισμού που δημιουργούν ένα ή πολλά στοιχεία περιπτώσεων δοκιμής που ικανοποιούν τους περιορισμούς. Στο μέλλον ερευνούνται εργασίες που θα περιλαμβάνουν την αυτοματοποίηση αυτής της διαδικασίας.

Αναφορές - Βιβλιογραφία

Χατζηγεωργίου[1] Αλέξανδρος, Αντικειμενοστραφής Ανάλυση, UML, Αρχές, Πρότυπα και Ευρετικοί κανόνες, Εκδόσεις Κλειδάριθμος, 2005, σελ 224, ISBN 960-209-882-1.

Χατζηγεωργίου[2] Αλέξανδρος, Αντικειμενοστραφής Ανάλυση, UML, Αρχές, Πρότυπα και Ευρετικοί κανόνες, Εκδόσεις Κλειδάριθμος, 2005, σελ 38, ISBN 960-209-882-1.

Χατζηγεωργίου[3] Αλέξανδρος, Αντικειμενοστραφής Ανάλυση, UML, Αρχές, Πρότυπα και Ευρετικοί κανόνες, Εκδόσεις Κλειδάριθμος, 2005, σελ 58-64, ISBN 960-209-882-1.

Alexander I., Towards automatic traceability in the industrial practice, 1st international workshop on traceability in emerging forms of software engineering, Edinburgh, UK, 2002.

Arlow J. and I. Neustadt. UML 2 and the Unified Process Second Edition: Practical Object-Oriented Analysis and Design. Addison-Wesley, 2005.

Balasubramaniam R., Matthias,J., Toward Reference Models for Requirements Traceability, IEEE Press, 2001.

Basanieri F., Bertolino, A., Marchetti, E., The Cow_Suite Approach to Planning and Deriving Test Suites in UML Projects, Proceedings of the 5th International Conference on The Unified Modeling Language, Springer-Verlag, 2002, pp. 383-397.

Binder R. V., Round-trip Scenario Test, Testing Object-Oriented Systems - Models, Patterns, and Tools, Addison-Wesley, 2000, pp. 579-591.

Bouquet F., Jaffuel, E., Legeard, B., Peureux, F., Utting, M., Requirements Traceability in Automated Test Generation - Application to Smart Card Software Validation, ICSE Int. Workshop on Advances in Model-Based Software Testing (A-MOST'05), ACM Press, St. Louis, USA, 2005.

Briand L. C., Labiche, Y., A UML-Based Approach to System Testing, 4th International Conference on the Unified Modeling Language (UML), Toronto, Canada, 2001, pp. 194-208.

Corriveau J.-P., "Traceability Process for Large OO Projects". IEEE Computer, pp. 63-68, September 1996.

Csallner, Smaragdakis, Y., JCrasher: An automatic robustness tester for Java, 2004, pp. 1025-1050.

Dick, Faivre, A., Automating the Generation and Sequencing of Test Cases from Model-Based Specifications, Springer-Verlag, 1993.

Didonet Del Fabro, Generating an ATL Execution Trace as a Traceability Weaving Model (ATL to Tracer), http://www.eclipse.org/gmt/amw/examples/#ATL2WTracer.

Didonet Del Fabro, Bézivin, J., Valduriez, P., Weaving Models with the Eclipse AMW plugin, Eclipse Modeling Symposium, Eclipse Summit Europe 2006, Esslingen, Germany, 2006.

Dinh-Trong, Ghosh, S., France, R. B., A Systematic Approach to Generate Inputs to Test UML Design Models, 17th International Symposium on
Software Reliability Engineering (ISSRE'06), 2006, pp. 95-104.

Dömges and K. Pohl. "Adapting Traceability Environments to Project-Specific Needs". Communications of ACM, Vol. 41, No 21, December 1998.

Egyed A., A scenario-driven approach to trace dependency analysis, IEEE Transactions on Software Engineering, 2003.

Fraikin, Leonhardt, T., SeDiTeC — Testing Based on Sequence Diagrams, 17th IEEE International Conference on Automated Software Engineering, 2002, pp. 261 - 266.

Garousi, Briand, L. C., Labiche Y., Control Flow Analysis of UML 2.0 Sequence Diagrams, European Conference on Model Driven Architecture- Foundations and Applications (ECMDA-FA), 2005.

Gotel, O., Finkelstein, A. An Analysis of the Requirements Traceability Problem Proc. of First International Conference on Requirements Engineering, 1994, pages 94-101.

Grieskamp, Nachmanson, L., Tillmann, N., Veanes, M., Test Case Generation from AsmL Specifications - Tool Overview, 10th International Workshop on Abstract State Machines, Taormina, Italy, 2003.

Hartman A., Nagin, K., The AGEDIS tools for model based testing, 2004 ACM SIGSOFT international symposium on Software testing and analysis, ACM Press, Boston, Massachusetts, USA, 2004, pp. 129- 132.

Haumer, M. Jarke, K. Pohl and K. Weidenhaupt. "Improving reviews of conceptual models by extended traceability to captured system usage". Interacting with Computers, Elsevier Science, 13 (1) pp. 77-95, 2000. ftp://sunsite.informatik.rwth-aachen.de/pub/CREWS/CREWS-99-16.ps.gz .

Hull, Elizabeth, Ken Jackson, Jeremy Dick (2005). Requirements Engineering (Second Edition). Springer, pp. 9-13, 131-151. ISBN 1-85233-879-2.

IEEE Standards Software Engineering, IEEE Standard Glossary of Software Engineering Terminology, IEEE Std. 610-1990, The Institute of Electrical and Electronics Engineers, 1999, ISBN 0-7381-1559-2.

Jacobson I.. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison Wesley, Reading, Massachusetts, June 1992.

Jacobson I., G. Booch and J. Rumbaugh. The Unified Software Development Process. Addison-Wesley, 1999.

Jeremiah, Frank, M., Using UML to Partially Automate Generation of Scenario-Based Test Drivers, 7th International Conference on Object Oriented Information Systems, 2001, pp. 303-306.

Jouault, Loosely Coupled Traceability for ATL, European Conference on Model Driven Architecture Workshop on Traceability, Nuremberg, Germany, 2005.

Knethen, A. A Trace Model for System Requirements Changes on Embedded Systems Proc. of 4th International Workshop on Principles of Software Evolution, September 2001.

Knethen, A. Change-Oriented Requirements Traceability. Support for Evolution of Embedded Systems Proc. of International Conference on Software Maintenance, October 2002, pages 482-485.

Lehman, M., Ramil, J. Software Evolution – Background, Theory, Practice Information Processing Letters, Vol. 88, Issues 1-2, October 2003, pages 33- 44.

Leite J.C. and A.Oliveira. "A Client Oriented Requirements Baseline". In Proceedings of the Second IEEE International Symposium on Requirements Engineering (RE'95), pp. 108-115, IEEE Computer Society Press, 1995.

Leite J.C., G. Rossi, F. Balaguer, V. Maiorana, G. Kaplan, G. Hadad, A. Oliveros. "Enhancing a Requirements Baseline with Scenarios". Requirements Engineering Vol. 2, No. 4, pp. 184-198, 1997.

Letelier P., A framework for requirements traceability in UML-based projects. In Proc. of 1st TEFSE, Edinburgh, UK, Sept. 2002.

Lindval M., Sandahl, K., Practical Implications of Traceability, Software Practice and Experience, 1996, pp. 1161-1180.

Maeder Patrick, Ilka Philippow and Matthias Riebisch, Technical University of Ilmenau, Germany, A Traceability Link Model for the Unified Process, Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing 2007.

Naslavsky, L., Ziv, H., Richardson, D. J., "Towards Traceability of Model-based Testing Artifacts", Third Workshop on the Advances of Model-Based Testing (A-MOST 2007), co-located with the International Symposium on Software Testing and Analysis (ISSTA 2007), London, UK, July 2007 .

Naslavsky, L., Richardson, D. J., "Using traceability to support model-based regression testing", 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'07) Doctoral Symposium, Atlanta, GA, November 2007.

Naslavsky, L., Ziv, H., Richardson, D. J., Scenario-based and State Machine-based Testing: An Evaluation of Automated Approaches, ISR, University of California,
Irvine, 2006.

Nebut, C., Fleurey, F., Traon, Y. L., Jézéquel, J., Automatic Test Generation: A Use Case Driven Approach, IEEE Transactions on Software Engineering, 32 (2006), pp. 140-155.

OMG Unified Modeling Language Specification. UML 1.4 with Action Semantics, Final Adopted Specification, January 2002. www.omg.org .

Orso, A., Shi, N., Harrold, M. J., Scaling Regression Testing to Large Software Systems, 12th ACM SIGSOFT Sysmposium on the Foundations of Software Engineering (FSE 2004), 2004.

Palo Mirka, Requirements Traceability, Seminar Report, Department of Computer Science, University of Helsinki, 30th October 2003.

Pfleeger Shari Lawrence, Τεχνολογία Λογισμικού Θεωρία και Πράξη, Εκδόσεις Κλειδάριθμος, 2003, σελ. 56 , 157, ISBN 690-209-620-9.

Pinheiro and J. Goguen. "An Object-Oriented Tool for Tracing Requirements". IEEE Software, pp. 52-64, March 1996.

Ramesh B. and M. Jarke. Toward reference models of requirements traceability. IEEE Trans. Software Eng, 27(1):58–93, 2001.

Ramesh B., "Factors influencing requirements traceability practice". Communication of the ACM, Vol. 41, No. 12, pp. 37-44, December 1998.

Richardson D. J. , TAOS: Testing with Analysis and Oracle Support, ACM Press, Seattle, Washington, United States, 1994.

Richardson D. J., Aha, S. L., O'Malley, T. O., Specification-based test oracles for reactive systems, Proceedings of the 14th international conference on Software engineering, ACM Press, Melbourne, Australia, 1992, pp. 105-118.

Rothermel, G., Harrold, M. J., Analyzing Regression Test Selection Techniques, IEEE Transactions on Software Engineering, 1996, pp. 529-551.

Rountev, Kagan, S., Sawin, J., Coverage Criteria for Testing of Object Interactions in Sequence Diagrams, Fundamental Approaches to Software
Engineering, Springer Berlin / Heidelberg, 2005, pp. 289-304.

Scheer, A. ,Business Process Engineering: Reference Models for Industrial Enterprises
Springer-Verlag, 1998.

SEI, CMMISM for Systems Engineering/Software Engineering, Version 1.02 (CMMISW/
SW, V 1.02); CMMI Staged Representation, CMU/SEI-2000-TR-018, ESC-TR-2000-018; Continuous Representation, CMU/SEI-2000-TR-019, ESC-TR-2000-019;
Product Development Team; Software Engineering Institute; November 2000.

Spanoudakis, Zisman, A., Software Traceability: A Roadmap, Advances in Software Engineering and Knowledge Engineering, World Scientific Publishing, 2005.

Spence I. and L. Probasco. Traceability strategies for managing requirements with use cases. Rational Software White Paper TP166, IBM, 2000.

Spence and Probasco, "Traceability Studies for Managing Requirements with Use Cases". Rational Software White Paper, 1998. www.rational.com/products/whitepapers/022701.jsp.

Stocks, P., Carrington, D., A Framework for Specification- Based Testing, IEEE Transactions on Software Engineering, 22 (1996), pp. 777-793.

Toranzo and Castro. "A Comprehensive Traceability Model to Support the Design of Interactive Systems". ECOOP Workshops 1999, pp. 283-284, Lecture Notes in Computer Science 1743, Springer-Verlag, 1999.

Uchitel, Chatley, R., Kramer, J., Magee, J., System Architecture: the Context for Scenario-based Model Synthesis, 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering, 2004, pp. 33-42.

Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P., Scenarios in System Development: Current Practice, IEEE Software, 1998.

Weilkiens, Systems Engineering with SysML/UML. dpunkt.verlag, 2006.

Wikipedia, ανακτήθηκε από την παρακάτω ιστοσελίδα τον Αύγουστο του 2009: "http://en.wikipedia.org/wiki/Requirements_traceability".

Wittevrongel, Maurer, F., SCENTOR: Scenario- Based Testing of E-Business Applications, Tenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2001, pp. 41 - 46.

Xie T., Marinov, D., Notkin, D., Rostra: A Framework for Detecting Redundant Object-Oriented Unit Tests, 19th IEEE International Conference on Automated Software Engineering (ASE'04), 2004, pp. 196-205.