Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Περιβάλλον Πλαίσιο για την Ανάλυση και Μετατροπή Μοντέλων MOF

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΙΧΑΗΛ Γ. ΦΑΜΕΛΗΣ

Επιβλέπων : Κωνσταντίνος Κοντογιάννης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2008

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Περιβάλλον Πλαίσιο για την Ανάλυση και Μετατροπή Μοντέλων MOF

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΙΧΑΗΛ Γ. ΦΑΜΕΛΗΣ

Επιβλέπων : Κωνσταντίνος Κοντογιάννης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21η Ιουλίου 2008.

.....	.....	.....
Κωνσταντίνος Κοντογιάννης Αναπληρωτής Καθηγητής Ε.Μ.Π.	Ιωάννης Βασιλείου Καθηγητής Ε.Μ.Π.	Τιμολέων Σελλής Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2008

.....

Μιχαήλ Γ. Φαμέλης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Καθώς υιοθετείται όλο και περισσότερο η μοντελοκεντρική προσέγγιση στην τεχνολογία λογισμικού, τα μοντέλα σαν τεχνολογικά υποκείμενα πρώτης τάξης και η διαχείριση τους σαν τέτοια αποκτούν ολοένα και σημαντικότερο ρόλο σε ολόκληρη την διάρκεια ζωής ενός τεχνικού έργου λογισμικού.

Σε πολλές εφαρμογές είναι απαραίτητο να συνθέτουμε, να μετατρέπουμε και να συγκρίνουμε διαφορετικά μοντέλα που περιγράφουν διαφορετικά στοιχεία του συστήματος. Παραδείγματα τέτοιων μοντέλων είναι μοντέλα σχεδίασης, μοντέλα ελέγχου, μοντέλα παρακολούθησης λειτουργίας του συστήματος κτλ. Η σύνθεση νέων μοντέλων από άλλα μοντέλα είναι βασική για την υλοποίηση εργαλείων που υποστηρίζουν την μοντελοκεντρική ανάπτυξη πολύπλοκων εφαρμογών λογισμικού.

Ο σκοπός αυτής της εργασίας είναι η σχεδίαση μιας άλγεβρας μοντέλων με συγκεκριμένους τελεστές που επιτρέπουν λειτουργίες όπως τη σύνθεση μοντέλων (model merging) και τη διαφορά μοντέλων (model differencing). Η άλγεβρα αυτή σχεδιάζεται για μοντέλα που υλοποιούνται σύμφωνα με το πρότυπο Meta Object Facility (MOF) και όσον αφορά τον μαθηματικό φορμαλισμό τους, περιγράφονται με τη μορφή γράφων.

Προς την κατεύθυνση αυτή, οι πράξεις της σύνθεσης και της διαφοράς μελετώνται ως προς τη λειτουργία, την συμπεριφορά και τις αλγεβρικές τους ιδιότητες. Περιγράφονται οι προϋποθέσεις για την εκτέλεση τους και παρουσιάζονται αλγόριθμοι που να τις υλοποιούν. Επιπλέον προτείνεται μία επέκταση της πράξης της σύνθεσης μοντέλων με τη χρήση κανόνων αντιστοίχισης γραμμένων σε τριπλές γραμματικές γράφων, με σκοπό την επίτευξη μεγαλύτερης ευελιξίας στους τρόπους με τους οποίους μπορούν να συσχετίζονται τα μοντέλα που θέλουμε να συγχωνεύσουμε.

Η επεκτεταμένη πράξη της σύνθεσης, χρησιμοποιείται σε ένα πρακτικό παράδειγμα σύνθεσης μοντέλων που ανταποκρίνονται σε αρχεία καταγραφών που υπακούουν στο πρότυπο Common Base Event (CBE), ένα πρότυπο για καταγραφρή γεγονότων σε κατανεμημένα συστήματα λογισμικού.

Λέξεις κλειδιά:

Μοντελοκεντρική μηχανική, τεχνολογία λογισμικού, μοντέλα λογισμικού, άλγεβρα μοντέλων, πράξεις μοντέλων, σύνθεση μοντέλων, διαφορά μοντέλων, τριπλές γραμματικές γράφων, αρχεία καταγραφών.

Abstract

As the model driven approach to software engineering broadens its adoption base, models which are considered as first class technological objects, along with their manipulation as such, become increasingly important during the whole of the technical lifecycle of software projects.

In many applications, it is necessary to merge, transform and compare different models that describe different parts of a system. Examples of such models are design models, control models, system monitoring models etc. Merging new models from existing ones is essential for building tools that support the model driven development of complicated software applications.

The aim of this diploma thesis is to design a model algebra, with specific operators that facilitate operations such as model merging and model differencing. The algebra is designed for models that conform to the Meta Object Facility (MOF) specification and wich are represented as graphs with regard to their mathematical formalism.

To this end, we study the merging and differencing operators concerning their function, their behavior and their algebraic properties. We also look into the essential preconditions for their execution and we present algorithms that implement them. Moreover, we propose an extention to the model merging operator, using matching rules written in triple graph grammars, with the intention of achieving greater flexibility in describing ways to match models that we want to merge.

This, extended merging operator is then used to tackle a practical example of merging models representing log files that conform to the Common Base Event (CBE), a specification for logging events in distributed software systems.

Keywords:

Model driven engineering, software engineering, software models, model algebra, model operators, model merging, model differencing, triple graph grammars, log files.

Ευχαριστίες

Κανένας άνθρωπος δεν μπορεί να εργαστεί και να δημιουργήσει σε κοινωνικό κενό, πόσο μάλλον ένας νέος μηχανικός.

Για την καθοριστική του καθοδήγηση στην συγγραφή της εργασίας, αλλά και για την διεύρυνση των ερευνητικών και ακαδημαϊκών μου οριζόντων, νιώθω την υποχρέωση να ευχαριστήσω εκ βαθέων τον επιβλέποντα καθηγητή μου, κύριο Κώστα Κοντογιάννη, χωρίς την πνευματική και ακαδημαϊκή καθοδήγηση αλλά και την υπομονή και την κατανόηση του οποίου αυτή η εργασία δεν θα είχε υλοποιηθεί ποτέ.

Ευχαριστώ θερμότατα και τους καλούς καθηγητές μου, κυρίους Γιάννη Βασιλείου και Τίμο Σελλή για την συμβολή τους στην ακαδημαϊκή μου διαμόρφωση αλλά και για την τιμή που μου κάνουν συμμετέχοντας στην εξεταστική επιτροπή της διπλωματικής μου εργασίας.

Και βέβαια δεν μπορώ να ευχαριστήσω αρκετά όλους εκείνους τους ανθρώπους χωρίς τους οποίους δεν θα είχα ποτέ τη δυνατότητα να βλέπω με αισιοδοξία το μέλλον μου σαν νέος επιστήμονας μηχανικός αλλά και σαν ολοκληρωμένος πολίτης και άνθρωπος. Δεν μπορώ ποτέ να ευχαριστήσω αρκετά την οικογένεια μου, αλλά και όλους εκείνους τους ανθρώπους που με τιμούν καθημερινά με την αγάπη και τη φιλία τους.

Ιδιαίτερα θέλω επίσης να ευχαριστήσω και τον θείο μου Κωνσταντίνο Σταματάκη για την επιμονή και τις συμβουλές του, τον ανάδοχο μου Μανόλη Γιακουμάκη για τις συμβουλές του, και τις μοναχές της αδελφότητας της μονής *Παναγία η Οδηγήτρια* στην Πορταριά για την τεράστια πνευματική και ηθική τους υποστήριξη.

Η εργασία αυτή αφιερώνεται στη μνήμη των προπατόρων μου Μιχάλη, Μαριέττας, Παναγιώτη, Παναγιώτας και Μίνας.

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Η μοντελοκεντρική προσέγγιση στην Τεχνολογία Λογισμικού

Τα τελευταία χρόνια παρατηρείται μια εκρηκτική ανάπτυξη της βιομηχανίας παραγωγής λογισμικού παγκοσμίως. Η ανάπτυξη αυτής της βιομηχανίας έχει βάλει στο επίκεντρο του ενδιαφέροντος το ζήτημα της διαλειτουργικότητας και της ανεξαρτησίας από συγκεκριμένες υλοποιήσεις. Μια απάντηση που δίδεται σε αυτές τις ανάγκες είναι τόσο η ανάπτυξη και η προδιαγραφή προτύπων και πρωτοκόλλων που να εξασφαλίζουν τη διαλειτουργικότητα, όσο και η μετατόπιση σε τρόπους προγραμματισμού που εναγκαλίζονται όλο και περισσότερο αφαιρετικές μεθοδολογίες.

Σε αυτά τα πλαίσια, σημαντικό ρόλο έχει αρχίσει να παίζει η Μοντελοκεντρική Μηχανική (Model Driven Engineering, MDE), η συστηματική δηλαδή χρήση μοντέλων σαν τεχνολογικά αντικείμενα πρώτης τάξης για ολόκληρο τον κύκλο ζωής ενός τεχνικού έργου. Η Μοντελοκεντρική Μηχανική βασίζεται στις παραδοχές πως κάθε σύστημα μπορεί να αναπαρασταθεί από μοντέλα και πως κάθε μοντέλο υπακούει συντακτικά σε κάποιο μεταμοντέλο [1]. Η Μοντελοκεντρική Αρχιτεκτονική (Model Driven Architecture, MDA) του Object Management Group (OMG) αποτελεί την σημαντικότερη μοντελοκεντρικη προσέγγιση στο πεδίο της Τεχνολογίας Λογισμικού.

Στα πλάισια της MDA προσέγγισης, πρωταρχικό ρόλο παίζει το μοντέλο λογισμικού (software model), το οποίο αναγνωρίζεται σαν πρωτεύον αντικείμενο

της διαδικασίας σχεδίασης, κατασκευής και συντήρησης λογισμικών συστημάτων. Ένα μοντέλο κάποιου συστήματος είναι μια τυπική περιγραφή ή προδιαγραφή της λειτουργίας, δομής και/ή συμπεριφοράς αυτού του συστήματος και του περιβάλλοντός του για την εκπλήρωση κάποιου συγκεκριμένου σκοπού. Επίσης πρωταρχικό ρόλο στην MDA παίζει και η έννοια του μετασχηματισμού μοντέλων (model transformation), δηλαδή η μετατροπή ενός μοντέλου σε ένα άλλο ακολουθώντας συγκεκριμένους κανόνες μετασχηματισμού.

Χαρακτηριστικό της μοντελοκεντρικής προσέγγισης είναι πως η διαδικασία παραγωγής και ανάπτυξης του λογισμικού παραμένει σε υψηλό επίπεδο αφαίρεσης. Σκοπός της MDA προσέγγισης είναι η σταδιακή δημιουργία ενός συγκεκριμένου συστήματος λογισμικού που να ικανοποιεί προδιαγεγραμένες ανάγκες μέσα από μια διαδικασία μετασχηματισμών, ξεκινώντας από ένα μοντέλο ανεξάρτητο από πλατφόρμα (platform independent model, PIM) και καταλήγοντας σε ένα μοντέλο για κάποια συγκέκριμένη πλατφόρμα (platform specific model, PSM) [2].

Για τον σκοπό αυτό, όπως αναφέρεται και στο [3], στα πλαίσια της MDA προσέγγισης, σημαντικό ρόλο παίζουν υπάρχουσες τεχνολογίες, όπως για παράδειγμα η Unified Modeling Language (UML), το Meta Object Facility (MOF), το Common Warehouse Metamodel (CWM), το προφίλ Enterprise Distributed Object Computing (EDOC), το CORBA Component Model (CCM) και άλλα.

Ενα μοντέλο είναι η τυπική αναπαράσταση μιας όψης (viewpoint) ενός συστήματος. Είναι εμφανές πως ένα οποιοδήποτε σύστημα μπορεί να μοντελοποιηθεί από πολλές και διάφορες οπτικές γωνίες. Για παράδειγμα, για ένα λογισμικό σύστημα που μοντελοποιείται χρησιμοποιώντας την UML, μπορεί να ενδιαφέρουν μοντέλα που να υπακούουν σε οποιαδήποτε απο τα 13 διαφορετικά είδη διαγραμμάτων που προβλέπει η UML ώστε να αναπαρασταθεί η δομή, η συμπεριφορά ή η αλληλεπίδραση του συστήματος ή μερών του συστήματος μεταξύ τους.

Επίσης, μπορεί να ενδιαφέρει η αναπαράσταση της εξέλιξης του συστήματος μέσα στο χρόνο, η διασύνδεση του συστήματος με άλλα συστήματα, η μοντελοποίηση των καταγραφών γεγονότων που παράγει ένα ή περισσότερα συστήματα ή τελοσπάντων η αναπαράσταση διαφόρων πλευρών που αφορούν το σύστημα σε οποιαδήποτε επίπεδο, βαθμό λεπτομέρειας, χρονικό σημείο κτλ.

1.2 Στοιχεία διαχείρισης μοντέλων λογισμικού

Γίνεται λοιπόν εμφανές πως στη διάρκεια ζωής οποιουδήποτε τεχνικού έργου λογισμικού (αλλά βέβαια και ευρύτερα), ο μηχανικός ενδιαφέρεται να μπορεί να μεταχειρίζεται μια πληθώρα από μοντέλα. Η ανάγκη αυτή γίνεται ακόμα εμφανέστερη αν κανείς αναλογιστεί πως ένα τεχνικό έργο συνήθως εκτελείται από συνεργαζόμενες ομάδες ανθρώπων οι οποίες μπορεί να αναλαμβάνουν διαφορετικά κομμάτια του έργου. Η δυνατότητα αποτελεσματικής διαχείρισης των διαφόρων μοντέλων και των διαφόρων εκδόσεων αυτών των μοντέλων που παράγονται σε ένα κατανεμημένο και συνεργατικό περιβάλλον εργασίας αποκτά συνεπώς κομβική σημασία για κάθε προσπάθεια εφαρμογής της μοντελοκεντρικής προσέγγισης.

Καθώς λοιπόν σε ένα περιβάλλον μοντελοκεντρικής ανάπτυξης υπάρχουν κατανεμημένες ομάδες οι οποίες κατασκευάζουν και μεταχειρίζονται τα μοντέλα, και κάθε ομάδα δουλεύει σε μια μερική όψη του συνολικού συστήματος, είναι σημαντική η διαχείριση των μοντέλων. Όπως αναφερεται στο [4], η διαχείριση των μοντέλων συνίσταται στο διατηρείται μια καταγραφή των σχέσεων μεταξύ των διαφόρων μοντέλων και σε διάφορα χρονικά σημεία να συνδιάζονται πληροφορίες από πολλά μοντέλα σε ένα ενιαίο μοντέλο με μια διαδικασία που ονομάζεται *Σύνθεση Μοντέλων* (model merging).

Ο κύκλος ζωής ενός λογισμικού συστήματος περιλαμβάνει μεγάλες περιόδους συντήρησης και αναβάθμισης. Καθώς τα λογισμικά συστήματα εξελίσσονται με βάση τις διάφορες μεθοδολογίες ανάπτυξης (καταρράκτη, σπιράλ κτλ), γίνεται σημαντικό το να μπορούμε να μπορούμε να κατανοήσουμε ποια είναι η ακολουθία αλλαγών που υφίσταται το σύστημα [5]. Επίσης, ενδιαφέρει να μπορούμε να συγκρίνουμε την υλοποίηση του συστήματος σε κάποια δεδομένη στιγμή με τον αρχικό σχεδιασμό. Συνεπώς είναι απαραίτητη μια διαδικασία που ονομάζεται *Διαφορά Μοντέλων* (model differencing) με την οποία να είναι δυνατή η σύγκριση δύο μοντέλων και η απαρίθμηση των μεταξύ τους διαφορών.

Εκτός από την Σύνθεση και την Διαφορά μοντέλων, αρκετές ακόμα διαδικασίες μπορούν να επιστρατευτούν για ένα ολοκληρωμένο πλαίσιο μοντελοκεντρικής διαχείρισης ενός λογισμικού συστήματος. Αν μάλιστα θεωρηθεί ο αλγεβρικός χώρος με στοιχεία τα μοντέλα και πράξεις τις μοντελοκεντρικές διαδικασίες, είναι δυνατόν να υπάρξει μια πιο συστηματική μελέτη αυτών των διαδικασιών διαχείρισης.

Παραδείγματα τέτοιων προσεγγίσεων υπάρχουν στην βιβλιογραφία. Στο [4]

αναφέρονται οι πράξεις *Σύνθεση* (merge), *Διαφορά* (diff), *Αντιστοίχιση* (match), *Κατάτμηση* (split), *Προβολή* (slice), στο [6] περιγράφονται πράξεις βασισμένες σε ένοιες της συνολοθεωρίας (ένωση, τομή και διαφορά), ενώ στο [7] περιγράφεται ένας, διαφορετικής οπτικής, εναλλακτικός αλγεβρικός φορμαλισμός.

1.3 Σκοπός της διπλωματικής εργασίας

Με βάση όσα περιγράψαμε νωρίτερα, γίνεται εμφανής η ανάγκη να μελετηθούν μερικά από αυτά τα ζητήματα σε μεγαλύτερο βάθος. Σε αυτήν την κατεύθυνση φιλοδοξεί να συμβάλει αυτή η διπλωματική εργασία.

- Καθώς πολλά από τα ζητήματα που άπτονται της μοντελοκεντρικής προσέγγισης είναι από τη φύση τους σε ένα υψηλό επίπεδο αφαίρεσης, κρίνουμε πως είναι απαραίτητη μια σφαιρικότερη περιγραφή του γενικότερου πλαισίου μέσα στο οποίο αυτή νοείται. Μια τέτοια περιγραφή μοιραία θα πρέπει να αναζητήσει εκείνα τα πρότυπα και τις προδιαγραφές που καθορίζουν τα πλαίσια και τα όρια μιας μοντελοκεντρικής προσέγγισης στην τεχνολογία λογισμικου. Με άλλα λόγια, με αυτήν την διπλωματική θα επιχειρηθεί μια εποπτική παρουσίαση του τεχνολογικού οικοσυστήματος που άπτεται της μοντελοκεντρικής προσέγγισης.
- Όπως περιγράψαμε και νωρίτερα, η ολοένα και εντεινόμενη χρήση αφαιρετικών μορφών πολύ υψηλου επιπέδου για την αναπαράσταση και την προδιαγραφή των τεχνικών συστημάτων, δημιουργεί μεγάλες ανάγκες για την διαχείριση τους. Με την διπλωματική αυτή εργασία, ελπίζουμε να παρουσιάσουμε έναν συστηματικό τρόπο για την διαχείριση των μοντέλων, με τη μορφή μιας άλγεβρας μοντέλων.
- Πέραν όμως της περιγραφής της άλγεβρας μοντέλων, είναι απαραίτητο αυτή η άλγεβρα να τεθεί σε υλοποιήσιμες τεχνολογικά βάσεις, τέτοιες που να είναι δυνατόν να χρησιμοποιηθεί σαν αφετηρία ώστε να δημιουργηθούν πρακτικές εφαρμογές που να την υλοποιούν. Συνεπώς, εκτός από την εξωτερική περιγραφή μιας τέτοιας συστηματοποίησης της διαχείρισης μοντέλων, με την διπλωματική αυτή εργασία, φιλοδοξούμε να δώσουμε και μια λειτουργική διάσταση η οποία να μπορεί εύκολα να χρησιμοποιηθεί για την κατασκευή λογισμικών συστημάτων που να την υλοποιούν.
- Μια απλή προδιαγραφή μιας άλγεβρας μοντέλων δεν είναι βέβαια αρκετή. Σκοπεύουμε να παρουσιάσουμε και μια πρακτική εφαρμογή της

άλγεβρας μας που να δείχνει πως μπορεί να χρησιμοποιηθεί για την επίλυση υπαρκτών προβλημάτων. Συγκεκριμένα, θα επιχειρηθεί μια μοντελοκεντρική προσέγγιση στο πρόβλημα της διαχείρισης των καταγραφών που παράγονται από κατανεμημένες εφαρμογές, επικεντρώνοντας στο ζήτημα της σύνθεσης διαφορετικών καταγραφών σε μία ενιαία καταγραφή.

Στα πλαίσια αυτής της διπλωματικής εργασίας, λοιόν, θα επεκταθούμε στα ζητήματα που αφορούν στις προαναφερθείσες διαδικασίες διαχείρησης των μοντέλων, δίνοντας έμφαση στις πράξεις της *Σύνθεσης* και της *Διαφοράς* με σκοπό την περιγραφή ενός περιβάλλοντος πλαισίου το οποίο θα συνιστά μια άλγεβρα μετεξέλιξης μοντέλων. Απώτερος στόχος είναι η περιγραφή ενός συνολικού συστήματος μοντελοκεντρικής διαχείρισης λογισμικών συστημάτων με βάση σαφώς ορισμένους τελεστές.

Για τις ανάγκες της εργασίας, θα χρησιμοποιηθούν μοντέλα που υπακούουν στο πρότυπο MOF. Επιπλέον, τα μοντέλα θα θεωρηθούν σαν αντικείμενα που περιγράφονται από τον φορμαλισμό της Θεωρίας Γράφων και συγκεκριμένα σαν κατευθυνόμενοι γράφοι με τύπους και ιδιότητες (typed attributed directed graphs). Η χρήση γράφων για την αναπαράσταση των μοντέλων δίνει τη δυνατότητα να χρησιμοποιηθούν γραφοθεωρητικές και συνολοθεωρητικές τεχνικές, ενώ ταυτόχρονα διατηρείται η συμβατότητα με αναπαραστάσεις των μοντέλων που βασίζονται στο πρότυπο Meta Object Facility του OMG.

1.4 Η δομή της διπλωματικής εργασίας

Το υπόλοιπο της εργασίας αρθρώνεται σε τέσσερα κεφάλαια. Στο δεύτερο κεφάλαιο παρουσιάζεται συνοπτικά η υπάρχουσα σχετική επιστημονική γνώση στην οποία θα στηριχθεί η περιγραφή του περιβάλλοντος πλαισίου. Στο τρίτο κεφάλαιο περιγράφονται διεξοδικά οι τελεστές που συνιστούν την άλγεβρα μοντέλων. Στο τέταρτο κεφάλαιο παρουσιάζεται μια εφαρμογή της άλγεβρας σε μια συγκεκριμένη περίπτωση χρήσης, ενώ στο πέμπτο και τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα της περιγραφής του περιβάλλοντος πλαίσιο και της εφαρμογής του, καθώς και κάποιες προτάσεις για μελλοντική εξερεύνηση του πεδίου.

Συγκεκριμένα, στο Κεφάλαιο 2, αρχικά θα γίνει μια αναφορά στους τρόπους με τους οποίους αναπαριστούμε τα πρωταρχικά αντικείμενα που εξετάζουμε, τα μοντέλα. Θα αναφερθούμε στο Meta Object Facility, ένα πρότυπο για

την προδιαγραφή μοντέλων και στη συνέχεια θα παρουσιάσουμε πώς τα μοντέλα μπορούν να αναπαρασταθούν από μια ειδική κατηγορία γράφων. Ακολούθως θα αναφερθούμε στην έννοια του μετασχηματισμού μοντέλων και θα περιγράψουμε συνοπτικά το θεωρητικό πλαίσιο μέσα στο οποίο θεωρούνται οι μετασχηματισμοί. Θα παρουσιαστεί η έννοια της Άλγεβρας Μοντέλων, με συνοπτική παρουσίαση κάποιων χαρακτηριστικών παραδειγμάτων και θα αναφερθούν τα κυριότερα εργαλεία λογισμικού που είναι διαθέσιμα στους μηχανικούς.

Έχοντας εξετάσει περιληπτικά το υπάρχον γνωσιακό οικοσύστημα, στο Κεφάλαιο 3, θα προχωρήσουμε στην διατύπωση μιας στοιχειώδους άλγεβρας μοντέλων που αποτελείται από τις θεμελιώδεις πράξεις της Σύνθεσης και της Διαφοράς. Αρχικά θα γίνει μια σκιαγράφηση των πράξεων από την οπτική γωνία του μαύρου κουτιού και θα περιγραφεί η μακροσκοπική λειτουργία τους. Στη συνέχεια θα περιγραφεί ο τρόπος υλοποίησης τους, με την παρουσίαση των αλγορίθμων που χρησιμοποιούνται για την κάθε μία. Ακολούθως, θα γίνει παρουσίαση μιας επέκτασης της πράξης της Σύνθεσης με σκοπό την δυνατότητα χρησιμοποιήσης τους σε πιο γενικές συνθήκες, με πιο ασθενή προαπαιτούμενα. Το κεφάλαιο καταλήγει με μια σύντομη παρουσίαση των πράξεων από τη σκοπιά των αλγεβρικών τους ιδιοτήτων.

Στο Κεφάλαιο 4 θα προχωρήσουμε στην παρουσίαση μιας εφαρμογής της επεκτεταμένης πράξης της σύνθεσης. Το πεδίο στο οποίο θα εφαρμοστεί η μέθοδος είναι εκείνο των μοντέλων που υπακούουν στο πρότυπο Common Base Event (CBE), ένα πρότυπο για την προδιαγραφή ενός ενοποιημένου τρόπου ανταλλαγής δεδομένων που αφορούν σε γεγονότα που συμβαίνουν σε κατανεμημένα συστήματα. Αρχικά θα κάνουμε μια σύντομη παρουσίαση μιας απλοποιημένης εκδοχής του CBE, παρουσιάζοντας τις βασικές σχεδιαστικές του αρχές και τον τρόπο με τον οποίο αυτό μπορεί να χρησιμοποιηθεί για την δημιουργία αρχείων καταγραφών. Στη συνέχεια θα προσεγγίσουμε το πρόβλημα της σύνθεσης δύο αρχείων καταγραφών δομημένων κατά CBE σε ένα ενιαίο αρχείο καταγραφών, χρησιμοποιώντας την επεκτεταμένη πράξη της σύνθεσης που περιγράψαμε στο προηγούμενο κεφάλαιο.

Τέλος στο Κεφάλαιο 5 θα γίνει μια συνολική αποτίμηση των όσων θα έχουμε δει μέχρι τότε, καταλήγοντας την διπλωματική εργασία με μια ανακεφαλαιωτική ματιά. Θα αναφερθούμε στα συμπεράσματα στα οποία καταλήξαμε κατά τη διάρκεια της περιγραφής της άλγεβρας μοντέλων και του γενικότερου οικοσυστήματος μέσα στο οποίο αυτή νοείται και η διπλωματική εργασία θα καταλήξει με αναφορά σε εκείνα τα ζητήματα που θεωρήσαμε πως χρήζουν περαιτέρω διερεύνησης, με τη μορφή προτάσεων για μελλοντική έρευνα.

Κεφάλαιο 2

Σχετική επιστημονική γνώση

Έχοντας παρουσιάσει τις γενικές κατευθύνσεις και τον σκοπό αυτής της εργασίας, προχωρούμε τώρα σε μια συνοπτική παρουσίαση του σχετικού επιστημονικού περιβάλλοντος.

Για τον σκοπό αυτό αρχικά θα γίνει μια αναφορά στους τρόπους με τους οποίους αναπαριστούμε τα πρωταρχικά αντικείμενα που εξετάζουμε, τα μοντέλα. Θα αναφερθούμε στο Meta Object Facility, ένα πρότυπο για την προδιαγραφή μοντέλων και στη συνέχεια θα παρουσιάσουμε πώς τα μοντέλα μπορούν να αναπαρασταθούν από μια ειδική κατηγορία γράφων.

Στη συνέχεια θα αναφερθούμε στην έννοια του μετασχηματισμού μοντέλων και θα περιγράψουμε συνοπτικά το θεωρητικό πλαίσιο μέσα στο οποίο θεωρούνται οι μετασχηματισμοί. Ακολούθως, θα παρουσιαστεί η έννοια της Άλγεβρας Μοντέλων, με συνοπτική παρουσίαση κάποιων χαρακτηριστικών παραδειγμάτων.

Το κεφάλαιο αυτό της εργασίας καταλήγει με την περιληπτική παρουσίαση των κυριότερων εργαλείων που είναι διαθέσιμα για την χρήση από τους μηχανικούς της συνολικής μοντελοκεντρικής προσέγγισης στην πράξη.

2.1 Αναπαράσταση μοντέλων

Σύμφωνα με το OMG, ένα μοντέλο κάποιου συστήματος είναι μια τυπική περιγραφή ή προδιαγραφή της λειτουργίας, δομής και/ή συμπεριφοράς αυτού του

συστήματος και του περιβάλλοντός του για την εκπλήρωση κάποιου συγκεκριμένου σκοπού και συχνά αναπαρίσταται σαν ένας συνδυασμός σχημάτων και κειμένου [2].

Καθώς τα μοντέλα παίζουν κεντρικό ρόλο στην μοντελοκεντρική μηχανική, είναι επιτακτική η ανάγκη να αναπαρίστανται με ενιαίο, εκφραστικό και πλήρη τρόπο. Για τον σκοπό αυτό, έχουν προταθεί κατά καιρούς διάφορες *γλώσσες μοντελισμού* (modeling languages). Οι γλώσσες αυτές μπορούν να είναι γραφικές ή γραπτές.

Από τις πρώτες προσπάθειες στο χώρο της μοντελοκεντρικής τεχνολογίας λογισμικού ήταν η *Μέθοδος Booch*, η *Object-modeling technique* (OMT), το *Object-oriented software engineering* (OOSE), οι οποίες άλλωστε αποτέλεσαν προπομπούς της *Unified Modeling Language* (UML). Επίσης, στον χώρο των γλωσσών μοντελισμού, μπορούν να αναφερθούν και η οικογένεια γλωσσών *IDEF*, η *SysML*, τα *Petri Nets* και άλλες γλώσσες.

Με τους όρους, της πιο διαδεδομένης γλώσσας μοντελισμού, της UML, όπως αναφέρεται στο [6], ένα μοντέλο είναι ένα στιγμιότυπο του UML μεταμοντέλου και ένα διάγραμμα αναπαριστά μια γραφική αναπαράσταση ενός μοντέλου. Επιπλέον, ένα στοιχείο ενός μοντέλου είναι ένα στιγμιότυπο κάποιας UML μέτα-κλάσης και ένα πεδίο ενός στοιχείου ενός μοντέλου είναι ένα στιγμιότυπο ενός μέτα-πεδίου που ανήκει στο στοιχείο του μοντέλου. Η κατάσταση ενός μοντέλου ορίζεται από τις τιμές των πεδίων.

2.1.1 Το πρότυπο Meta Object Facility (MOF)

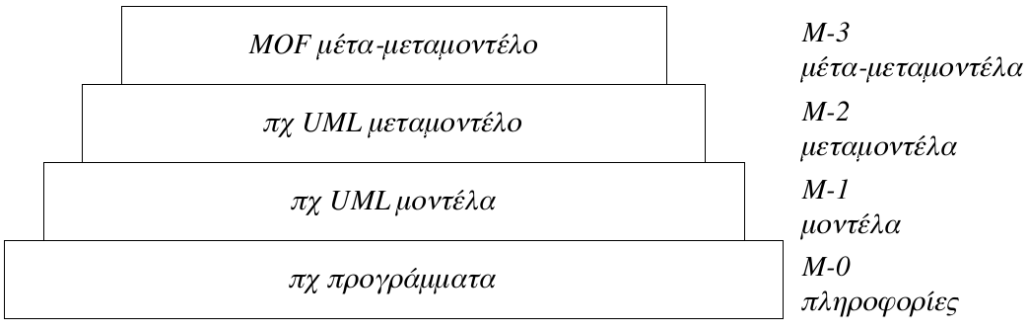
Η Model Driven Architecture (MDA), που αποτελεί την σημαντικότερη σύγχρονη μεντελοκεντρική προσέγγιση στο χώρο της τεχνολογίας λογισμικού, είναι μια ενοποιητική προσέγγιση, η οποία έχει σαν τεχνολογική βάση μια σειρά από ήδη υπάρχουσες σχετικές τεχνολογίες, στις οποίες περιλαμβάνονται η Unified Modeling Language (UML), το Meta Object Facility (MOF), το XML Metadata Interchange (XMI) κτλ.

Όπως αναφέρεται στο [8], το Meta Object Facility περιγράφει μια αφηρημένη γλώσσα και ένα περιβάλλον πλαίσιο για την προδιαγραφή, κατασκευή και διαχείριση τεχνολογικά ουδέτερων μεταμοντέλων, όπου ένα μεταμοντέλο είναι κατ' ουσίαν μια αφηρημένη γλώσσα για κάποιου είδους μεταδεδομένα. Σαν παραδείγματα μεταμοντέλων αναφέρονται η UML, το Common Warehouse Metamodel (CWM), αλλα και το ίδιο το MOF.

Επιπλέον, το MOF ορίζει ένα περιβάλλον πλάισιο για να ορίζονται μοντέλα που περιγράφονται από τα προαναφερθέντα μεταμοντέλα. Στα πλαίσια του MOF, τα μοντέλα ορίζονται και αναπαρίστανται με τρόπο τέτοιο ώστε να διατηρείται η συνέπεια και η συμβατότητα μεταξύ προιόντων διαφορετικών κατασκευαστών και διαφορετικών τεχνολογικών υλοποιήσεων. Αυτό επιτυγχάνεται με την χρήση προτύπων αντιστοιχίσεων μεταξύ MOF μεταμοντέλων και των διαφόρων υπαρχόντων προγραμματιστικών διεπαφών (APIs) για μοντέλα.

Μετα-επίπεδα

Ο παραδοσιακός τρόπος με τον οποίο περιγράφουμε το περιβάλλον στο οποίο κινούμαστε όταν δουλεύουμε με μοντέλα είναι να περιγράφουμε μια ιεραρχία συνόλων από μοντέλα που οργανώνεται σε μια αρχιτεκτονική τεσσάρων επιπέδων [8], όπου τα στοιχεία σε κάθε επίπεδο αποτελούν στιγμιότυπα στοιχείων του προηγούμενου επιπέδου.



Σχήμα 2.1: Η αρχιτεκτονική των τεσσάρων επιπέδων.

Στην κορυφή της ιεραρχίας των τεσσάρων επιπέδων βρίσκεται το επίπεδο M-3. Στο επίπεδο αυτό βρίσκονται τα μέτα-μεταμοντέλα, των οποίων ο ρόλος είναι το να ορίζουν γλώσσες με τις οποίες να είναι δυνατή η προδιαγραφή μεταμοντέλων. Το MOF είναι μια χαρακτηριστική περίπτωση ενός μέτα-μεταμοντέλου. Σύμφωνα με το [9], είναι εν γένει επιθυμητό τα μεταμοντέλα και το μέτα-μεταμοντέλο που τα ορίζει να μοιράζονται κοινές δομές και φιλοσοφία σχεδιασμού, αν και κάθε επίπεδο της ιεραρχίας πρέπει να νοείται ανεξάρτητα από τα υπόλοιπα. Τα μέτα-μεταμοντέλα ορίζονται χρησιμοποιώντας μέτα-μεταμοντέλα, δηλαδή τα στοιχεία του επιπέδου M-3 ορίζονται αυτοαναφορικά: το MOF είναι ορισμένο με χρήση του ίδιου του MOF.

Τα στοιχεία του επιπέδο M-2 είναι τα μεταμοντέλα, δηλαδή οι γλώσσες μοντελισμού, τις οποίες χρησιμοποιούμε για να ορίσουμε μοντέλα. Χαρακτηριστικά παραδείγματα μεταμοντέλων είναι η UML και το CWM. Τα μεταμοντέλα υπακούουν σε κάποιο μέτα-μεταμοντέλο: για παράδειγμα, η UML είναι στιγμιότυπο του MOF (ιστορικά, το MOF ορίστηκε με σκοπό να δημιουργηθεί ένας φορμαλισμός για την περιγραφή της UML). Τα μεταμοντέλα είναι συνήθως πιο περίπλοκα από τα αντίστοιχα μέτα-μεταμοντέλα, ειδικά όταν υπάρχει η ανάγκη να περιγραφούν δυναμικές σημασιολογίες.

Τα ίδια τα μοντέλα κατοικούν στο επίπεδο M-1 και είναι στιγμιότυπα κάποιου μεταμοντέλου από το επίπεδο M-2. Όπως έχει αναφερθεί, σκοπός των μοντέλων είναι η περιγραφή κάποιας όψης ενός συστήματος ή, με την ευρεία ένοια, η αναπαράσταση μιας όψης του πραγματικού κόσμου, και με αυτή την έννοια τα στοιχεία του επιπέδου M-1 αναφέρονται σε κάποιο συγκεκριμένο πεδίο εφαρμογής. Στο πεδίο των συστημάτων λογισμικού, χαρακτηριστική περίπτωση στοιχείων του επιπέδου M-1 είναι τα μοντέλα των λογισμικών συστημάτων τα οποία περιγράφονται με κάποια UML μοντέλα (πχ διαγράμματα κλάσεων, ακολουθιακά διαγράμματα κτλ).

Στη βάση της ιεραρχίας των τεσσάρων επιπέδων βρίσκεται το επίπεδο M-0. Στπ επίπεδο αυτό βρίσκονται τα ίδια τα δεδομένα τα οποία θέλουμε να μοντελοποιήσουμε και τα οποία ανταποκρίνονται σε στοιχεία που περιγράφονται στο επίπεδο M-1.

Παρόλο που η αρχιτεκτονική των τεσσάρων επιπέδων είναι πολύ περιγραφική του ρόλου τον οποίο καλείται να εκπληρώσει το MOF, είναι σημαντικό να τονιστεί πως η ιεραρχία αυτή δεν είναι απόλυτο χαρακτήρα. Για την ακρίβεια, το μόνο απαράλλακτο χαρακτηριστικό των συστημάτων μοντελισμού είναι η ύπαρξη του ζεύγους κλάση-στιγμιότυπο μαζί με ένα μηχανισμό για να πηγαίνουμε από το στιγμιότυπο στην κλάση του. Από εκεί και πέρα, συστήματα μοντελισμού μπορούν να αποτελούνται από οποιονδήποτε αριθμό μετα-επιπέδων μεγαλύτερο του δύο. Στο [10] αναφέρεται ως παράδειγμα, η ιεραρχία τριών μετα-επιπέδων που μπορεί να αναγωνριστεί στα σχεσιακά συστήματα βάσεων δεδομένων: Πίνακας Συστήματος/Πίνακας/Γραμμή (SysTable/Table/Row). Το MOF έχει σχεδιαστεί με σκοπό να μπορεί να χρησιμοποιηθεί σε αρχιτεκτονικές από δύο και περισσότερα μετα-επίπεδα, ανάλογα με τις μοντελιστικές ανάγκες του εκάστοτε προβλήματος.

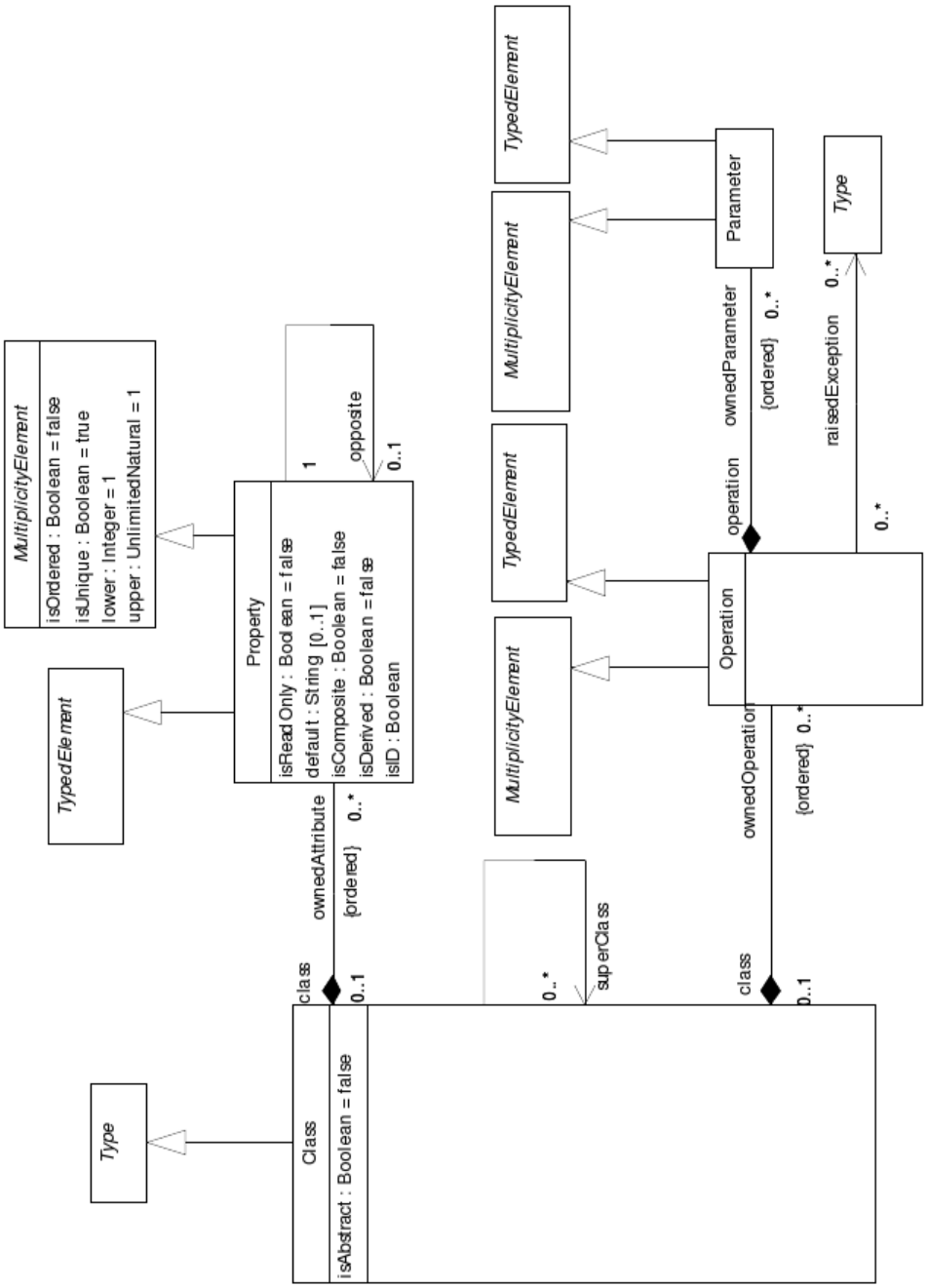
Δομή και περιγραφή του MOF

Η πιο πρόσφατη έκδοση του MOF (2.0) έχει σχεδιαστεί και προδιαγραφεί με τέτοιο τρόπο ώστε να είναι στενά συνδεδεμένο με την UML 2.0. Επιπλέον, για την προδιαγραφή του MOF χρησιμοποιείται σε μεγάλο βαθμό το συντακτικό και η σημασιολογία της UML. Για την ακρίβεια, το MOF προδιαγράφεται με χρήση ενός υποσυνόλου της UML (κατά βάση τα διαγράμματα κλάσεων), μιας γλώσσας προδιαγραφής περιορισμών (object contstaint language, OCL) και ακριβή φυσική γλώσσα.

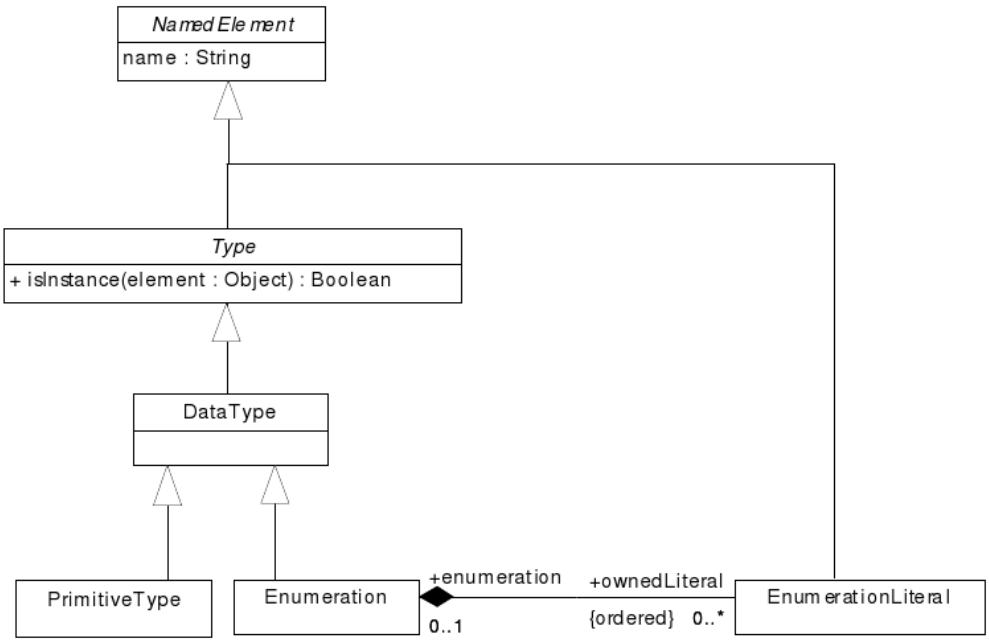
Η προδιαγραφή προβλέπει δύο παραλλαγές του MOF: την βασική (Essential MOF, EMOF) και την πλήρη (Complete MOF, CMOF). Και οι δύο μπορούν να περιγραφούν χρησιμοποιώντας τον φορμαλισμό του CMOF (με χρήση του οποίου έχει προδιαγραφεί και η UML 2.0), ενώ το EMOF μπορεί να περιγραφεί πλήρως και από τον εαυτό του. Όπως αναφέρεται χαρακτηριστικά και στο [10]: *το EMOF και το CMOF περιγράφονται χρησιμοποιώντας τον εαυτό τους και το καθένα προέρχεται ή επαναχρησιμοποιεί μέρος της προδιαγραφής της UML 2.0*.

Ενώ ο σκοπός του πλήρους MOF (CMOF) είναι να παρέχει ένα γενικό περιβάλλον πλαίσιο για μεταμοντελισμό, το βασικό MOF (EMOF), το οποίο αποτελεί υποσύνολο του CMOF, είναι προσανατολισμένο στο να παρέχει ένα τρόπο να περιγράφει τα χαρακτηριστικά και τις λειτουργίες που συναντώνται εν γένει στις αντικειμενοστρεφείς γλώσσες προγραμματισμού και στα αρχεία αναπαράστασης δεδομένων που υπακούουν στο XML πρότυπο.

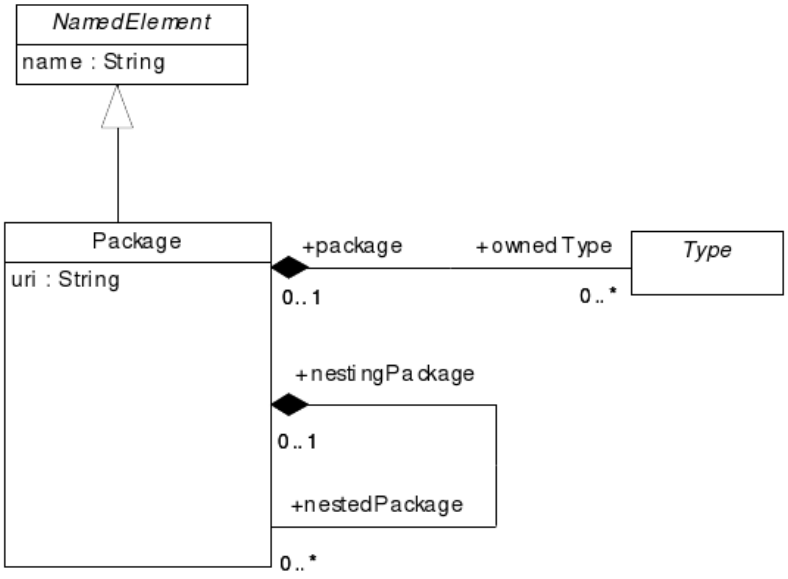
Καθώς η λειτουργικότητα του EMOF είναι αρκετή για τις ανάγκες αυτής της εργασίας, παρατίθενται παρακάτω τα MOF διαγράμματα που προδιαγράφουν το EMOF, όπως παρουσιάζονται στο έγγραφο προδιαγραφής του MOF 2.0 [10]. Να σημειωθεί πως ο πλήρης ορισμός του EMOF περιλαμβάνει και μια σειρά από περιορισμούς (constraints) και κάποιες επιπλέον οδηγίες και ορισμούς, τα οποία μπορούν να αναζητηθούν στο έγγραφο προδιαγραφής.



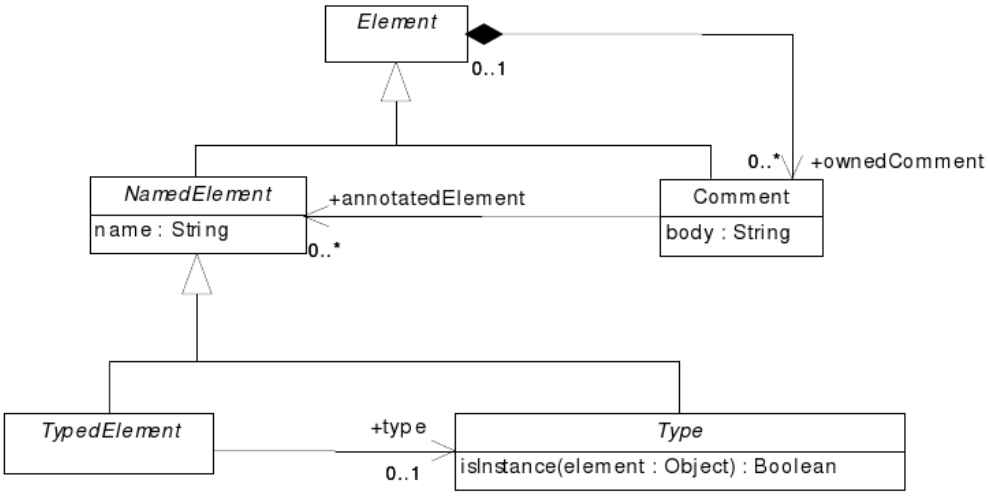
Σχήμα 2.2: EMOF προδιαγραφή Κλάσεων.



Σχήμα 2.3: EMOF προδιαγραφή Τύπων Δεδομένων.



Σχήμα 2.4: EMOF προδιαγραφή Πακέτων.



Σχήμα 2.5: EMOF προδιαγραφή τύπων.

Πρότυπες αντιστοιχίσεις και XML αναπαράσταση

Το MOF περιγράφει ένα περιβάλλον πλαίσιο στο οποίο μπορούν να εκτελεστούν εργασίες που περιλαμβάνουν μοντέλα και μεταμοντέλα. Όμως το MOF είναι προδιαγραφή, όχι υλοποίηση. Το OMG αφήνει στην ευχέρεια των διαφόρων παραγωγών λογισμικού την δημιουργία υλοποιήσεων που να ανταποκρίνονται στην προδιαγραφλη του MOF. Το MOF μπορεί μάλιστα να αποδειχτεί ιδιαίτερα χρήσιμο σε ένα προγραμματιστικό περιβάλλον όπου συνυπάρχουν διάφορες τεχνολογίες και άρα είναι επιτακτικό το ζήτημα της διασύνδεσιμότητας με πρότυπους τρόπους.

Το ζήτημα αυτό αντιμετωπίζεται από την προτυποποίηση του MOF με την προδιαγραφή πρότυπων αντιστοιχήσεων σε άλλες πρότυπες τεχνολογικές πλατφόρμες. Για τον σκοπό αυτόν, παρέχονται προδιαγραφές για αντιστοιχίσεις από MOF σε διάφορα άλλα πρότυπα και τεχνολογίες, όπως για παράδειγμα αντιστοιχίσεις από MOF σε Interface Definition Language (IDL) [11], από MOF σε Java με το Java Metadata Interface (JMI) [12], από MOF σε XML με το XML Metadata Interchange (XMI) [13], και άλλες.

Το XMI είναι ένα πρότυπο του OMG με το οποίο καθορίζονται κανόνες με τους οποίους μπορεί να παράγεται ένα XML σχήμα που να περιγράφει αρχεία XML που περιέχουν μεταδεδομένα τα οποία υπακούουν σε κάποιο MOF μοντέλο. Με αυτόν τον τρόπο επιτυγχάνεται συνέπεια και συμβατότητα μεταξύ εφαρμογών που χρησιμοποιούν κάποιο πρότυπο που περιγράδεται με MOF. Χαρακτηριστικότερο παράδειγμα είναι η χρήση του XMI για την αναπαράσταση μοντέλων προδιαγεγραμμένων σε UML.

2.1.2 Τα μοντέλα σαν γράφοι

Έχουμε λοιπόν ένα πλήρες περιβάλλον πλαίσιο για την αναπαράσταση των μοντέλων. Έχουμε τόσο γραφικούς τρόπους να τα αναπαριστούμε (UML, MOF), όσο και γραπτούς (XMI). Χαρακτηριστικό των γραφικών τρόπων αναπαράστασης είναι πως είναι δυνατόν οι αναπαραστάσεις να αναγνωριστούν σαν μια ειδική περίπτωση γράφων.

Στην γενική περίπτωση στα πλαίσια της εργασίας, τα μοντέλα θεωρούνται επιπλέον και σαν κατευθυνόμενοι γράφοι με τύπους και ιδιότητες (typed attributed directed graphs). Καθώς είναι δυνατόν μια οποιαδήποτε ιδιότητα ενός στοιχείου ενός μοντέλου να αναπαρασταθεί με έναν επιπλέον κόμβο ο οποίος συνδέεται με το στοιχείο-πατέρα, μπορούμε να θεωρούμε τα μοντέλα απλώς σαν

κατευθυνόμενους γράφους με τύπους.

Παρακάτω περιγράφεται το γενικό θεωρητικό πλαίσιο το οποίο θα χρησιμοποιηθεί για τον χειρισμό των μοντέλων με τη μορφή γράφων. Κάποιες από τις έννοιες είναι δανεισμένες από την Θεωρία Κατηγοριών (Category Theory), χωρίς όμως να αναφερόμαστε με μεγαλύτερη λεπτομέρεια στις κατηγοροθεωρητικές πλευρές του πλαισίου.

Κατευθυνόμενοι γράφοι

Όπως αναφέρεται στο [14], ένας γράφος, ορίζεται σαν το διατεταγμένο ζεύγος $G = (N, E)$ όπου N είναι ένα σύνολο κόμβων, E είναι ένα σύνολο ακμών και ισχύει ότι $N \neq \emptyset$ και το E είναι ένα υποσύνο του συνόλου των μη διατεταγμένων ζευγών του N.

Όπως περιγράφεται στο [15], ένας *κατευθυνόμενος* γράφος ορίζεται σαν την πλειάδα $G = (N, E, src, tgt)$ όπου $src, tgt : E \rightarrow N$ είναι συναρτήσεις που δίνουν αντίστοιχα τον αρχικό και τελικό κόμβο μιας κατευθυνόμενης ακμής.

Ομομορφισμοί

Αν $G = (N, E, src, tgt)$ και $G' = (N', E', src', tgt')$ είναι δυο γράφοι, τότε ένας ομομορφισμός (homomorphism) $h : G \rightarrow G'$ μεταξύ γράφων ορίζεται σαν ένα ζευγάρι συναρτήσεων

$$\langle h_{node} : N \rightarrow N', h_{edge} : E \rightarrow E' \rangle$$

τέτοιο ώστε για κάθε ακμή $e \in E$, αν η h_{edge} αντιστοιχίζει την e στην e', τότε η h_{node} αντιστοιχίζει τους κόμβους αρχής και τέλους της e με εκείνους της e'. Δηλαδή:

$$src'(h_{edge}(e)) = h_{node}(src(e))$$
$$tgt'(h_{edge}(e)) = h_{node}(tgt(e))$$

Μοντέλα

Τα μοντέλα αναπαρίστανται σαν κατευθυνόμενοι γράφοι με τύπους, όπως περιγράφεται στο [15]. Άν λοιπόν το μεταμοντέλο της υπό μελέτη γλώσσας μοντέλων αναπαρίσταται από τον γράφο τύπων $\mathcal{M}$, κάθε μοντέλο περιγράφεται

από ένα ζεύγος $\langle G, t\rangle$ όπου G είναι ένας γράφος και t ένας ομομορφισμός $t : G \to \mathcal{M}$ ο οποίος αναθέτει έναν τύπο σε κάθε στοιχείο του G και καλείται *αντιστοίχιση τύπου* (typing map).

Στα πλαίσια της διπλωματικής εργασίας, σαν γράφο τύπων $\mathcal{M}$ θα θεωρούμε το μεταμοντέλο EMOF. Δηλαδή, για παράδειγμα, για μια κλάση c ενός μεταμοντέλου $L = \langle G, t\rangle$, θα είναι $t(c) =$ EMOF::Class. Όμοια, για ένα στιγμιότυπο o της κλάσης c σε ένα μοντέλο $\langle G_i, t_i\rangle$, θα είναι $t_i(o) = L{::}c$.

Συνεπώς, καθώς σαν γράφο τύπων χρησιμοποιούμε το πρότυπο EMOF, τα στοιχεία των μοντέλων έχουν σαν τύπο κάποια από τις μη αφηρημένες κλάσεις του EMOF. Δηλαδή τα στοιχεία μπορεί να είναι:

- EMOF::Class
- EMOF::Property
- EMOF::Operation
- EMOF::Parameter
- EMOF::Package
- EMOF::Enumeration
- EMOF::PrimitiveType
- EMOF::DataType
- EMOF::EnumerationLiteral
- EMOF::Comment

Ένας ομομορφισμός με τύπους $\underline{h} : \langle G, t\rangle \to \langle G', t'\rangle$ είναι απλά ένας ομομορφισμός $h : G \to G'$ που διατηρεί τους τύπους, δηλαδή $t'(h(x)) = t(x)$ για κάθε στοιχείο x του γράφου G.

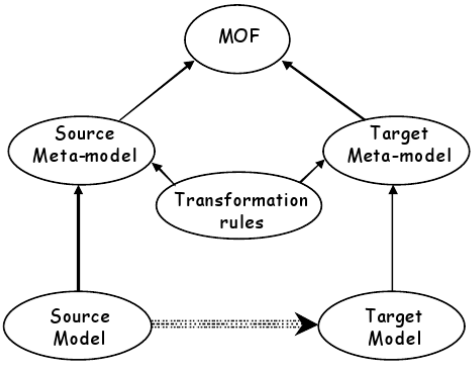
2.2 Μετασχηματισμός μοντέλων

Ένας μετασχηματισμός μοντέλων (model transformation) είναι μια διαδικασία η οποία παίρνει σαν είσοδο ένα μοντέλο και ακολουθώντας συγκεκριμένους κανόνες βγάζει σαν έξοδο κάποιο άλλο μοντέλο.

Στην MDA, η διαδικασία του μετασχηματισμού μοντέλων παίζει σημαντικό ρόλο. Όπως περιγράφεται στο [2], από την προδιαγραφή ενός μοντέλου ανεξάρτητου από συγκεκριμένη τεχνολογική πλατφόρμα (Platform Independent Model, PIM), με τη χρήση κάποιου μετασχηματισμού μοντέλων κατασκευάζεται ένα μοντέλο για κάποια συγκεκριμένη τεχνολογική πλατφόρμα (Platform Specific Model, PSM). Ο μετασχηματισμός αυτός προβλέπεται να γίνεται είτε εντελώς αυτόματα, είτε με τη βοήθεια του ηλεκτρονικού υπολογιστή, είτε χειρωνακτικά. Η διαδρομή από PIM σε PSM δεν είναι βέβαια η μόνη διαδρομή μετασχηματισμών που ενδιαφέρει, αφού με την ευρύτερη έννοια, κάθε επέμβαση σε κάποιο μοντέλο αποτελεί έναν μετασχηματισμό.

Στην πιο γενική περίπτωση, το είδος του μοντέλου εισόδου μπορεί να διαφέρει από το είδος του μοντέλου εξόδου. Με τις διατυπώσεις της μοντελοκεντρικής αρχιτεκτονικής, αυτό σημαίνει πως το μεταμοντέλο του μοντέλου εξόδου μπορεί να είναι διαφορετικό από εκείνο του μοντέλου εισόδου. Όπως αναφέρεται στο [16], οι μετασχηματισμοί όπου τα μοντέλα εισόδου και εξόδου υπακούουν στο ίδιο μεταμοντέλο ονομάζονται ενδογενείς (endogenous), ενώ αν υπακούουν σε διαφορετικά μεταμοντέλα ονομάζονται εξωγενείς (exogenous).

Σε ένα περιβάλλον εργασίας όπου χρησιμοποιείται σαν μετα-μεταμοντέλο το MOF, η διαδικασία του μετασχηματισμού μπορεί να περιγραφεί από το παρακάτω διάγραμμα, που προέρχεται από το [17].

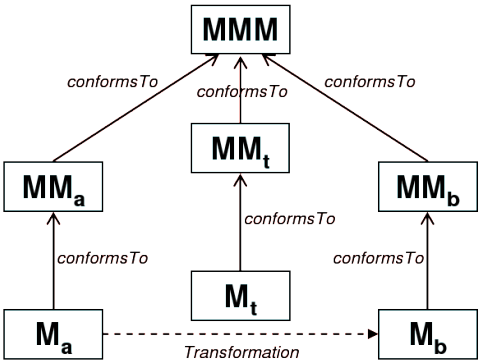


Σχήμα 2.6: Μετασχηματισμός μοντέλων.

Για τον μετασχηματισμό μοντέλων έχουν αναπτυχθεί διάφορες σχτικές γλώσσες, προδιαγραφές και τεχνολογίες. Χαρακτηριστική περίπτώση είναι το MOF QVT (Query/View/Transformation) [18]. Το MOF QVT αποτελεί προδιαγραφή του OMG, για το οποίο έχουν αναπτυχθεί διάφορες υλοποιήσεις, η σημαντικότερη των οποίων είναι το Eclipse M2M, σημαντικό ρόλο στο οποίο παίζει η γλώσσα ATL που αναπτύσσεται από την INRIA.

Να σημειωθεί πως, καθώς το MOF είναι στενά συνδεδεμένο με το πρότυπο XMI, είναι δυνατό να υλοποιηθεί η διαδικασία του μετασχηματισμού μοντέλων και με την χρήση τεχνολογιών μετασχηματισμού αρχείων XML. Χαρακτηριστικότερη περίπτωση είναι η χρήση της τεχνολογίας XSLT, όπως για παράδειγμα στο [19].

Στα πλαίσια της μοντελοκεντρικής προσέγγισης, ακόμα και ο ίδιος ο μετασχηματισμός είναι ένα μοντέλο. Ακόμα περισσότερο, ο μετασχηματισμός είναι ένα μοντέλο το οποίο έχει προδιαγραφεί με κάποια γλώσσα μοντελισμού η οποία με τη σειρά της δεν είναι παρά ένα μεταμοντέλο το οποίο υπακούει στο κοινό μετα-μεταμοντέλο στο οποίο υπακούουν και τα μεταμοντέλα των μοντέλων εισόδου και εξόδου. Η προσέγγιση αυτή αναπαρίσταται στο παρακάτω διάγραμμα, που προέρχεται από το [20].



Σχήμα 2.7: Ο μετασχηματισμός μοντέλων σαν μοντέλο.

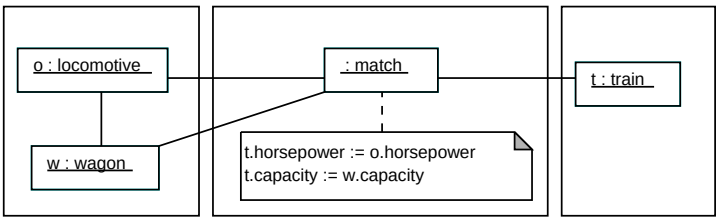
Δουλεύοντας σε ένα τέτοιο περιβάλλον, όπου οι μετασχηματισμοί μοντέλων είναι επίσης μοντέλα, και έχοντας κατά νου πως με τη χρήση γραφικού φορμαλισμού, όπως πχ του EMOF, τα μοντέλα αναπαρίστανται σαν γράφοι, είναι δυνατή η περιγραφή της διαδικασίας του μετασχηματισμού μοντέλων με Γραμματικές Γράφων.

Χαρακτηριστικό τέτοιο παράδειγμα είναι ο μετασχηματισμός μοντέλων με την χρήση *Τριπλών Γραμματικών Γράφων* (Triple Graph Grammars, TGG). Οι τριπλές γραμματικές γράφων, μια αναλυτική παρουσίαση των οποίων περιλαμβάνεται στο [21] είναι μια τεχνική με την οποία είναι δυνατόν να περιγράφονται συσχετίσεις μεταξύ μοντέλων, με γραφικό τρόπο. Ταυτόχρονα οι συσχετίσεις είναι δυνατόν να μετατραπούν σε λειτουργικούς κανόνες, υποστηρίζοντας έτσι την δυνατότητα του μετασχηματισμού μοντέλων. Το [22] αναφέρεται ακριβώς στον μετασχηματισμό μοντέλων με τη χρήση τριπλών γραμματικών γράφων.

Ένας κανόνας γραμμένος σε TGG αποτελείται από τρία μέρη. Στην αριστερή πλευρά ενός κανόνα περιγράφεται το μοτίβο (pattern) το οποίο βρίσκεται σε ένα μοντέλο εισόδου και το οποίο θα μετασχηματιστεί μέσω του κανόνα. Με την εφαρμογή του κανόνα, το μοτίβο που βρίσκεται στην αριστερή πλευρά του κανόνα μετασχηματίζεται στο μοτίβο το οποίο γράφεται στην δεξιά πλευρά του κανόνα. Οι αντιστοιχίσεις μεταξύ των δύο μοτίβων βρίσκονται στο κεντρικό μέρος του κανόνα.

Όπως περιγράφεται στο [21], οι κανόνες μπορούν να περιλαμβάνουν περιορισμούς, ώστε να περιγράφονται σχέσεις μεταξύ των τιμών των πεδίων των μοτίβων εισόδου και εξόδου. Επιπλέον, παρέχεται η δυνατότητα να περιγράφονται πληροφορίες που αφορούν τον περίγυρο των μοτίβων, αλλά και η δυνατότητα να περιγράφονται απαγορευμένες καταστάσεις, συνδέσεις και μοτίβα. Για λεπτομερή περιγραφή όλων των παραπάνω, ο αναγνώστης παρακαλείται να ανατρέξει στην βιβλιογραφία.

Παρακάτω φαίνεται ένα παράδειγμα ενός κανόνα TGG. Ο κανόνας περιγράφει πώς το μοτίβο ατμομηχανή-συνδεδεμένη-με-βαγόνι μετατρέπεται σε ένα ενιαίο τρένο. Ο περιορισμός (αναπαρίσταται με σχόλιο, όμοια με τον τρόπο συμβολισμού του [22]) καθορίζει πώς τα χαρακτηριστικά της ατμομηχανής και του βαγονιού θα μεταφερθούν στο τρένο.



Σχήμα 2.8: Κανόνας TGG.

Οι τριπλές γραμματικές γράφων δεν είναι βέβαια ο μόνος τρόπος να οριστούν γραμματικές γράφων. Άλλα παραδείγματα περιλαμβάνουν Γραμματικές Γράφων με τύπους και ιδιότητες (typed attributed Graph Grammars) όπως αυτές που περιγράφονται στο [23] και το [24], οι Διαδικασίες Γράφων που περιγράφονται στο [15], ενώ το [25] είναι μια διεξοδική μελέτη των γραμματικών γράφων και των λειτουργιών πάνω σε γράφους.

2.3 Άλγεβρες μοντέλων

Αν αναγνωριστεί ο ρόλος των μετασχηματισμών μοντέλων στην μοντελοκεντρική προσέγγιση στην τεχνολογία λογισμικού, αρχίζει να γίνεται σαφέστερη η ανάγκη συστηματοποίησης αυτών των μετασχηματισμών. Η κεντρική υπόθεση είναι πως είναι δυνατόν να αναγνωριστούν κάποιοι στοιχειώδεις μετασχηματισμοί ή έστω κάποιοι συχνά χρησιμοποιούμενοι μετασχηματισμοί και να συστηματοποιηθούν σαν ένα σύνολο από βασικές λειτουργίες.

Είναι λοιπόν δυνατό, χωρίς αξιώσεις απόλυτης μαθηματικής αυστηρότητας, να θεωρηθούν αφηρημένες άλγεβρες με θεμελιώδεις πράξεις που επενεργούν πάνω σε μοντέλα. Αν επιπλέον θεωρηθεί η αναπαράσταση των μοντέλων με γράφους, είναι δυνατόν να αξιωθεί περισσότερη μαθηματική αυστηρότητα, ορίζοντας πλέον πράξεις πάνω σε γράφους, και να διερευνηθούν οι αλγεβρικές ιδιότητες αυτών των πράξεων. Έχουν προταθεί διάφορες προσεγγίσεις στο θέμα με διάφορους βαθμούς μαθηματικής αυστηρότητας. Η μορφή και η δομή της εκάστοτε θεωρούμενης άλγεβρας μπορεί φυσικά να διαφέρει.

Στο [6], περιγράφεται μια άλγεβρα που αποτελείται από από πράξεις συνόλων πάνω σε UML μοντέλα προδιαγεγραμμένα με χρήση διαγραμμάτων κλάσεων, συνθετηματικών διαγραμμάτων (component diagrams), και διαγραμμάτων παράταξης (deployment diagrams). Καθώς το πρότυπο MOF επαναχρησιμοποιεί τη γλώσσα και τους φορμαλισμούς της UML, η άλγεβρα αυτή μπορεί να χρησιμοποιηθεί και για μοντέλα MOF.

Η άλγεβρα των πράξεων συνόλων, απαρτίζεται από πράξεις που παίρνουν δύο μοντέλα σαν είσοδο και παράγουν ένα μοντέλο σαν έξοδο. Οι πράξεις επιπλέον θεωρείται πως είναι δυνατόν να συνδιαστούν μεταξύ τους για την κατασκευή πολυπλοκότερων πράξεων.

Στα πλαίσια αυτής της άλγεβρας ορίζονται τρεις δυαδικές πράξεις συνόλων. Όλες οι πράξεις νοούνται σαν λειτουργίες τριών φάσεων: ανακάλυψη των ένα προς ένα αντιστοίχισεων μεταξύ των στοιχείων των μοντέλων εισόδου, συμβιβασμός των κοινών στοιχείων με παράλληλη επίλυση αντιφάσεων και εκτέλεση της πράξης.

Οι πράξεις που ορίζονται είναι η Ένωση, η Τομή και η Διαφορά. Η Ένωση δύο μοντέλων παράγει ένα μοντέλο που περιέχει όλα τα στοιχεία και των δύο μοντέλων, όπου τα στοιχεία των δύο μοντέλων που έχουν αντιστοιχιστεί μεταξύ τους συντίθενται σε ενιαία στοιχεία του νέου μοντέλου. Η Τομή, παράγει ένα μοντέλο το οποίο περιέχει μόνο τα στοιχεία εκείνα των δύο μοντέλων εισόδου

που έχουν αντιστοιχιστεί μεταξύ τους. Και για τις δύο πράξεις υιοθετείται μια στρατηγική επίλυσης αντιφάσεων και διαφορών όπου προτιμάται η κατάσταση του ένός από τα δύο μοντέλα ανάλογα με το βαθμό αφαίρεσης, πληρότητας και παλαιότητας του κάθε μοντέλου.

Η Διαφορά ενός μοντέλου από ένα άλλο παράγει ένα μοντέλο που περιέχει μόνο εκείνα τα στοιχεία του πρώτου μοντέλου τα οποία δεν έχουν αντιστοιχιστεί με τα στοιχεία του δεύτερου. Επιπλέον, ορίζονται δύο υποπεριπτώσεις της Διαφοράς, ανάλογα με το πώς αυτή χειρίζεται στοιχεία που περιέχουν μη αντιστοιχισμένα στοιχεία. Ορίζεται η Διατηρητική Διαφορά, όπου στοιχεία που περιέχουν μη αντιστοιχισμένα στοιχεία παραμένουν στο μοντέλο ανεξάρτητα από το αν τα ίδια έχουν αντιστοιχιστεί και η Καταστρεπτική Διαφορά, όπου αυτά τα στοιχεία αφαιρούνται από το τελικό μοντέλο.

Μια παρόμοια προσέγγιση περιγράφεται στο [4]. Σε αυτήν την περίπτωση, η άλγεβρα μοντέλων επικεντρώνεται στην πράξη της σύνθεσης μοντέλων και με κέντρο αυτήν την πράξη ορίζονται και οι υπόλοιπες πράξεις.

Οι πράξεις που περιγράφονται στα πλαίσια αυτής της προσέγγισης είναι η Σύνθεση μοντέλων, η Αντιστοίχιση, η Διαφορά, η Διαμέριση και η Προβολή. Επιπλέον αυτών των πράξεων ορίζονται και σαν βοηθητικές πράξεις ο Έλεγχος Ιδιότητας, ο Έλεγχος Συνέπειας, η Εφαρμογή και η Προώθηση, αν και στην συνολική εικόνα, όλες οι πράξεις θεωρούνται από τη σκοπιά της Σύνθεσης και άρα λίγο πολύ θεωρούνται σαν βοηθητικές προς αυτήν.

Οι πράξεις που περιγράφονται σε αυτήν την προσέγγιση δεν ακολουθούν το αυστηρά δυαδικό μοτίβο των πράξεων συνόλων της άλγεβρας που περιγράφηκε νωρίτερα, τουλάχιστον όχι σε τυπικό επίπεδο. Επίσης, οι πράξεις αυτές δεν ορίζονται μόνο πάνω στο σύνολο των μοντέλων. Στην είσοδο και την έξοδο τους εμφανίζονται και άλλου είδους οντότητες, όπως σχέσεις, μετασχηματισμοί, κριτήρια, αληθοτιμές και ιδιότητες. Επίσης, δεν ακολουθούν κάποιο αυστηρό τρόπο εκτέλεσης, όπως η εκτέλεση τριών σταδίων της άλγεβρας των πράξεων συνόλων και γενικά σαν προσέγγιση επιδιώκει περισσότερο να περιγράψει τις πράξεις σαν ένα περιβάλλον εργασίας παρά σαν μια αυστηρά ορισμένη άλγεβρα.

Η πράξη της Σύνθεσης δημιουργεί ένα ενιαίο μοντέλο από δύο μοντέλα και την μεταξύ τους σχέση. Αντιστοιχεί στην πράξη της Ένωσης που περιγράφηκε νωρίτερα. Η πράξη της Αντιστοίχισης παράγει μια σχέση μεταξύ δύο δεδομένων μοντέλων και αντιστοιχεί στο πρώτο στάδιο εκτέλεσης των πράξεων συνόλων. Η πράξη της Διαφοράς είναι παρόμοια σε σημασιολογία με την πράξη της Τομής, αλλά αντί για μοντέλο παράγει στην έξοδο της έναν μετασχηματισμό, δη-

λαδή μια διαδικασία για να μετατραπέι το ένα μοντέλο εισόδου στο άλλο. Η πράξη της Διαμέρισης ορίζεται σαν την ακριβώς αντίστροφη πράξη της Σύνθεσης, λαμβάνοντας ένα μοντέλο και διασπώντας το σε δύο, δίνοντας παράλληλα και την μεταξύ τους σχέση, ενώ η πράξη της Προβολής δημιουργεί μια μερική όψη του μοντέλου εισόδου αφαιρώντας όσα στοιχεία δεν εκπληρούν κάποιο κριτήριο.

Όσον αφορά τις βοηθητικές πράξεις, ο Έλεγχος Ιδιότητας, ο Έλεγχος Συνέπειας είναι πράξεις που επιστρέφουν, αληθοτιμές (true και false). Ο Έλεγχος Ιδιότητας ελέγχει αν κάποιο μοντέλο εισόδου ικανοποιεί κάποια ιδιότητα, ενώ ο Έλεγχος συνέπειας ελέγχει δύο μοντέλα σε σχέση με μια επιθυμητή αντιστοίχηση μεταξύ τους. Τέλος, η πράξη της Εφαρμογής μετατρέπει το μοντέλο εισόδου της σε ένα μοντέλο εξόδου ακολουθώντας τις οδηγίες που περιγράφονται σε έναν μετασχηματισμό εισόδου, ενώ η πράξη της Προώθησης συγχρονίζει δύο μοντέλα που σχετίζονται με μια δεδομένη σχέση σύμφωνα με έναν μετασχηματισμό.

Στην παραπάνω άλγεβρα, με κάποιες μεταβολές βασίζεται και η παρούσα εργασία.

Τέλος, ενδιαφέρον παρουσιάζει και μια αρκετά διαφορετική προσέγγιση στο θέμα, η οποία ακολουθείται στο [7], όπου περιγράφεται η λεγόμενη *Άλγεβρα Συστημάτων* (Algebra of Systems, AoS). Η προσέγγιση αυτή δεν ξεκινά από τις αναπαραστάσεις μοντέλων που συναντήσαμε έως τώρα (όπως πχ αυτή που προδιαγράφεται απο το MOF), αλλά χτίζει ένα δικό της τρόπο να περιγράφονται τα μοντέλα, με σκοπό της να αποτελέσει ένα πλαίσιο στο οποίο να είναι δυνατή η αποδοτική περιγραφή τόσο της λειτουργιας όσο και της δομής των συστημάτων.

Στα πλαίσια της Άλγεβρας Συστημάτων ορίζονται τα μοντέλα χρησιμοποιώντας μια τριάδα από πεδία ορισμού (πεδίο ιδιοτήτων, πεδίο αληθοτιμών και πεδίο κατασκευής) για την αναπαράσταση διαφόρων απόψεων των μοντέλων. Για κάθε ένα από αυτά τα πεδία ορισμού ορίζονται κλειστές πράξεις πάνω στα στοιχεία του. Τα τρία αυτά πεδία ορισμού σαν τριάδα αναπαριστούν τα μοντέλα και αποτελούν επίσης ένα πεδίο ορισμού πάνω στο οποίο ορίζονται κλειστές πράξεις. Για αναλυτικότερη περιγραφή, ο αναγνώστης παραπέμπεται στην βιβλιογραφία.

2.4 Εργαλεία

Σε μια διαδικασία όπως αυτή του μοντελισμού, είναι δεδομένο ότι η διαθεσιμότητα επαρκών εργαλείων είναι τεράστιας σημασίας. Στα παρακάτω θα επιχειρηθεί μια σύντομη παρουσίαση κάποιων από τα χαρακτηριστικότερα, με έμφαση σε προγράμματα που εμπίπτουν στην κατηγορία του ελεύθερου λογισμικού.

Το *Eclipse Modeling Framework* (EMF) [26] ίσως είναι η σημαντικότερη τεχνολογία στον χώρο. Το EMF ξεκίνησε από το Eclipse Foundation, σαν μια προσπάθεια να δημιουργηθεί μια υλοποιηση της προδιαγραφής MOF, αλλά στην πορεία διαφοροποιήθηκε σε διάφορα σημεία με βάση την εμπειρία χρήσης του σε διάφορα πρακτικά ζητήματα από τους μηχανικούς που το υλοποιήσαν.

Παρά τις διαφοροποιήσεις, οι δημιουργοί του EMF συνεχίζουν να το περιγράφουν σαν *μια πολύ αποδοτική υλοποίηση σε Java ενός βασικού υποσυνόλου του MOF*. Επιπλέον, στον πυρήνα του EMF έχει προβλεφθεί να βρίσκεται ένα μέτα-μεταμοντέλο το οποίο έχει ελάχιστες διαφορές (κυρίως λεξικογραφικού χαρακτήρα) με το Essential MOF (EMOF), το οποίο ονομάζουν Ecore.

Στην γενική του περιγραφή, το EMF παρουσιάζεται σαν ένα περιβάλλον πλαίσιο (framework) στο οποίο ο μηχανικός ή ο προγραμματιστής μπορεί να μοντελοποιήσει μια εφαρμογή και στη συνέχεια να χρησιμοποιήσει τις παρεχόμενες δυνατότητες του EMF για την αυτόματη δημιουργία πηγαίου κώδικα. Η μοντελοποίηση μπορεί να γίνει είτε με απευθείας γραφή του αρχείου XMI που προδιαγράφει το μοντέλο, είτε με χρήση κάποιου γραφικού εργαλείου υποβοήθησης της σχεδίασης (Computer Aided Software Engineering, CASE), είτε τέλος με την χρήση κατάλληλα υπομνηματισμένου κώδικα Java.

Με βάση τις παρεχόμενες πληροφορίες και με την προαιρετική βοήθεια κάποιων επιπλέον καθοδηγητικών εντολών, το EMF αναλαμβάνει να δημιουργήσει αυτόματα πηγαίο κώδικα ο οποίος μπορεί στη συνέχεια να χρησιμοποιηθεί για την δημιουργία στιγμιοτύπων των στοιχείων του αρχικού μοντέλου. Ταυτόχρονα, παρέχονται και δυνατότητες να ακολουθηθεί η αντίστροφη διαδρομή, ενώ το γενικότερο περιβάλλον πλαίσιο μπορεί να χρησιμοποιηθεί σαν βάση για την ανάπτυξη πιο εξειδικευμένων εργαλείων για την μοντελοκεντρική ανάπτυξη εφαρμογών λογισμικού.

Παράδειγμα τέτοιου εργαλείου είναι το *Acceleo* [27]. Το Acceleo χτίζει πάνω στην τεχνολογία του EMF και χρησιμέυει σαν αυτόματη γεννήτρια πηγαίου

κώδικα. Ουσιαστικά, το Acceleo αναλαμβάνει να παράξει ένα Platform Specific Model για κάποιο Platform Independent Model το οποίο έχει προδιαγραφεί στο περιβάλλον του EMF. Μάλιστα, το Acceleo δίνει την δυνατότητα στον μηχανικό να χρησιμοποιήσει διαφορετικές γεννήτριες κώδικα, ώστε να μπορεί να παράξει υλοποιήσεις του αρχικού μοντέλου σε διάφορες τεχνολογικές πλατφόρμες, όπως Java, C♯, PHP κτλ.

Η προσπάθεια για την υλοποίηση του EMF ξεκίνησε όταν ακόμα το MOF πρότυπο βρισκόταν στα πρώτα του βήματα. Η διαδικασία ανάπτυξης του MOF έφερε στον χώρο και νέες προτυποποιήσεις οι οποίες προς το παρόν δεν υλοποιούνται στο EMF. Χαρακτηριστικό παράδειγμα τέτοιας περίπτωσης είναι το JMI, η διεπαφή του MOF για την πλατφόρμα Java. Παρόλο που το EMF είναι μια τεχνολογία η οποία υλοποιεί το MOF στην Java πλατφόρμα, δεν έχει σχεδιαστεί με βάση το JMI. Αντίθετα, το *Metadata Repository* (MDR) που υλοποιείται στα πλαίσια του ανταγωνιστικού προς το Eclipse περιβάλλοντος εργασίας, Netbeans, έχει σχεδιαστεί ακριβώς με σκοπό να υλοποιεί το MOF πρότυπο, παραμένοντας πιστό στην προδιαγραφή JMI [28].

Τα περιβάλλοντα πλαίσια που επιχειρούν να κατασκευάσουν μια συνολική τεχνολογική προσέγγιση της ιδέας της Μοντελοκεντρικής Ανάπτυξης, είναι απαραίτητο και ένα εκτεταμένο οπλοστάσιο βοηθητικών εργαλείων. Χαρακτηριστική κατηγορία τέτοιων εργαλείων είναι τα εργαλεία σχεδίασης μοντέλων, τα οποία παίζουν έναν προφανώς σημαντικό ρόλο στην όλη διαδικασία καθώς τα μοντέλα προδιαγράφονται με τη χρήση γραφικών γλωσσών, πρώτα και κύρια με τη χρήση της Unified Modeling Language (UML).

Ίσως το σημαντικότερο εργαλείο στον χώρο των γραφικών επεξεργαστών διαγραμμάτων UML, είναι το *ArgoUML*. Το ArgoUML είναι μια πλήρης εφαρμογή ελεύθερου λογισμικού με την οποία ο μοντελιστής μπορεί να σχεδιάσει διάφορα μοντέλα σε UML και, με τη χρήση κατάλληλων προσθέτων, όπως το Argo2Ecore, να εξάγει τα μοντέλα σε κάποιο κοινό μορφότυπο αρχείου, όπως το XMI.

Άξιο αναφοράς είναι επίσης το *Poseidon For UML*, το οποίο αποτελεί εμπορική μετεξέλιξη του ArgoUML και παρέχει την ίδια λειτουργικότητα με περισσότερη σταθερότητα και ευκολία, ενώ επίσης δίνεται η δυνατότητα ενσωμάτωσης του Poseidon στο περιβάλλον εργασίας του Eclipse, καθώς και η δυνατότητα roundtrip engineering, δηλαδή του συγχρονισμού του προδιαγεγραμμένου σε UML μοντέλου με τον πηγαίο κώδικα που το υλοποιεί.

Τέλος, αρκετά διαδεδομένο είναι και η σχετικά περιορισμένων δυνατοτήτων εφαρμογή *Umbrello*, μέρος του περιβάλλοντος εργασίας KDE, ενώ σημαντικές

δυνατότητες παρέχουν και διάφορες επεκτάσεις του περιβάλλοντος εργασίας Eclipse, όπως ο *Eclipse Visual Editor* και το *Papyrus for UML.*

Κεφάλαιο 3

Άλγεβρα Μοντέλων

Έχοντας εξετάσει περιληπτικά το υπάρχον γνωσιακό οικοσύστημα, προχωρούμε τώρα στην διατύπωση μιας στοιχειώδους άλγεβρας μοντέλων που αποτελείται από τις θεμελιώδεις πράξεις της Σύνθεσης και της Διαφοράς.

Αρχικά θα γίνει μια σκιαγράφηση των πράξεων από την οπτική γωνία του μαύρου κουτιού και θα περιγραφεί η μακροσκοπική λειτουργία τους. Στη συνέχεια θα περιγραφεί ο τρόπος υλοποίησης τους, με την παρουσίαση των αλγορίθμων που χρησιμοποιούνται για την κάθε μία.

Ακολούθως, θα γίνει παρουσίαση μιας επέκτασης της πράξης της Σύνθεσης με σκοπό την δυνατότητα χρησιμοποιήσης τους σε πιο γενικές συνθήκες, με πιο ασθενή προαπαιτούμενα. Καταλήγοντας, θα γίνει μια σύντομη παρουσίαση των πράξεων από τη σκοπιά των αλγεβρικών τους ιδιοτήτων.

3.1 Θεμελιώδεις πράξεις

3.1.1 Σύνθεση μοντέλων

Η πράξη της σύνθεσης μοντέλων (model merging) έχει σαν σκοπό την δημιουργία ενός μοντέλου εξόδου από την συγχώνευση δύο μοντέλων εισόδου. Τα δύο μοντέλα εισόδου θεωρείται πως συνδέονται μέσω ενός δοσμένου συνόλου από αντιστοιχίσεις μεταξύ των στοιχείων τους. Το σύνολο αυτό από αντιστοιχίσεις ονομάζεται *σχέση* (relationship).

Χρησιμοποιώντας τους συμβολισμούς του [4], η σύνθεση μοντέλων γράφεται ως:

$$\text{σύνθεση} : \text{μοντέλο} \times \text{μοντέλο} \times \text{σχέση} \rightarrow \text{μοντέλο}$$

Να σημειωθεί πως αυτή είναι μια πολύ αφηρημένη (και εν πολλοίς εξιδανικευμένη) μορφή της πράξης της σύνθεσης. Στην πραγματικότητα, για να μπορεί να περιγραφεί η σύνθεση με τόσο απλό τρόπο, πρέπει να έχουν γίνει από πριν μια σειρά από υποθέσεις που αφορούν τόσο τα μοντέλα εισόδου όσο και τη σχέση μεταξύ τους. Ανάλογα με τις δεδομένες υποθέσεις μπορεί να έχουμε αρκετά διαφορετική συμπεριφορά της πράξης της σύνθεσης.

Μια βασική υπόθεση είναι ότι πρέπει τα μοντέλα εισόδου να είναι του ίδιου τύπου. Για παράδειγμα, μπορούμε να συνθέσουμε δύο μοντέλα που έχουν περιγραφεί με UML διαγράμματα κλάσεων ή δυο μοντέλα που έχουν περιγραφεί με Δίκτυα Petri.

Επίσης, μπορούμε με κάποια πολυπλοκότητα παραπάνω και κάποιες υποθέσεις παραπάνω, να συνθέσουμε μοντέλα τα οποία μπορούν να αναχθούν σε κάποια κοινή γλώσσα. Ένα παράδειγμα αναγωγής σε κοινή γλώσσα παρουσιάζεται στο [29]. Για παράδειγμα, μπορούμε να συνθέσουμε ένα μοντέλο που περιγράφεται με UML διάγραμμα κλάσεων με ένα μοντέλο που περιγράφεται σαν UML ακολουθιακό διάγραμμα, αν μετατρέψουμε πρώτα και τα δύο μοντέλα σε MOF διαγράμματα. Το μοντέλο εξόδου θα είναι βέβαια MOF διάγραμμα και χρειάζοντια κάποιες παραπάνω εξηγήσεις αν θέλουμε να επιστρέψουμε σε κάποιον από τους τύπους εισόδου.

Αντιστοίχιση Μοντέλων

Επιπλέον ζητήματα προκύπτουν όταν κανείς αρχίζει να εξετάζει την σχέση μεταξύ των δύο μοντέλων εισόδου. Στα πλαίσια της εργασίας, το σύνολο των αντιστοιχίσεων που συνιστούν την σχέση, θα θεωρηθεί δοσμένο από πριν. Όμως, σε μια οποιαδήποτε πρακτική εφαρμογή της υπό συζήτηση διαδικασίας, θα πρέπει να υπάρχει ένας τρόπους να ανακαλύπτεται μια τέτοιου είδους σχέση μεταξύ δυο δοσμένων μοντέλων.

Συνεπώς έχει νόημα να οριστεί η επιπλέον, βοηθητική πράξη της αντιστοίχισης μοντέλων (model matching). Η αντιστοίχιση παίρνει σαν είσοδο δύο μοντέλα και βγάζει σαν έξοδο μια σχέση (όπως την ορίσαμε παραπάνω). Με τους συμβολισμούς που χρησιμοποιήσαμε και νωρίτερα, θα γράφαμε πως η

αντιστοίχιση είναι:

$$\text{αντιστοίχιση} : \text{μοντέλο} \times \text{μοντέλο} \rightarrow \text{σχέση}$$

Ένας απλός τρόπος είναι να υποθέσουμε πως ο μηχανικός καθορίζει χειροκίνητα τις αντιστοιχίσεις. Βέβαια, το ζητούμενο είναι η αυτοματοποίηση της διαδικασίας αλλά, όπως αναφέρεται και στο [4], η σύνθεση μοντέλων είναι στην καθημερινή πρακτική, από τη φύση της μια διαδικασία εξερευνητικού χαρακτήρα, όπου δοκιμάζοντας διάφορους τρόπους να συνθέτει τα μοντέλα εισόδου, ο μηχανικός μπορεί να αντλεί χρήσιμες πληροφορίες για την σχέση μεταξύ τους. Ακόμα και με μια τέτοια οπτική όμως, το ζητούμενο είναι να μπορούν να δημιουργηθούν τέτοιες τεχνικές και εργαλεία που να κάνουν αυτό το έργο του μηχανικού πιο αποδοτικό.

Μπορούν να περιγραφούν πολλές και διαφορετικές μέθοδοι αυτοματοποιημένου καθορισμού της σχέσης μεταξύ δύο μοντέλων. Μια σημαντική κατηγορία τέτοιων μεθόδων βασίζεται στην υπόθεση πως τα μοντέλα εκτός από το να είναι του ίδιου τύπου, χρησιμοποιούν και κάποιου είδους λεξικογραφική σύμβαση, ώστε στοιχεία που αναπαριστούν τις ίδιες οντότητες, να παίρνουν όμοια ονόματα και στα δύο μοντέλα εισόδου ή έστω να χρησιμοποιούν ονόματα από ένα προκαθορισμένο πρότυπο αντιστοίχισης ονομάτων με οντότητες [30].

Στο [31] παρουσιάζεται μια μέθοδος ανακάλυψης των κοινών οντοτήτων μέσα από την ευριστική αποτίμηση λεξικογραφικών ομοιοτήτων. Χρήσιμη επίσης μπορεί να αποδειχθεί η θεωρία της Τυπικής Ανάλυσης Οντοτήτων (formal concept anaylsis). Για παράδειγμα, στο [32], χρησιμοποιείται η θεωρία της Τυπικής Ανάλυσης Οντοτήτων για την ανακάλυψη αντιστοιχίσεων μεταξύ μοντέλων, έχοντας πρώτα εφαρμόσει με μια τεχνική που βασίζεται σε Σχεσιακούς Κανόνες (association rules) για την δημιουργία ομάδων (clusters) από στοιχεία των μοντέλων που μπορεί να σχετίζονται.

Σχετικά με το πρόβλημα της εύρεσης των αντιστοιχίσεων είναι και τα ζητήματα που έχουν να κάνουν με την διαχείριση της αντφατικής πληροφορίας που μπορεί να βρίσκεται στα δύο μοντέλα. Κάποιες τεχνικές δεν δέχονται την ύπαρξη αντιφάσεων στα δύο μοντέλα εισόδου και επιτρέπουν την σύνθεση τους μόνο όταν υπάρχει συμαβότητα μεταξύ τους ή τελος πάντων προϋποθέτουν πως τα μοντέλα εισόδου τους είναι συνεπή. Άλλες τεχνικές, όπως στο [33] και στο [34], δίνουν τη δυνατότητα ενσωμάτωσης ή επίλυσης των αντιφάσεων, συχνά προβλέποντας κάποιο βαθμό χειροκίνητης παρέμβασης του μηχανικού.

Επιπλέον, τίθεται το θέμα της διαχείρισης της υπονοούμενης πληροφορίας, δηλαδή της πληροφορίας που δεν αναφέρεται ρητά στα δύο μοντέλα εισόδου. Όπως περιγράφεται στο [4], άλλες τεχνικές απορρίπτουν την υπονοούμενη πληροφορία, υιοθετώντας μια προσέγγιση κλειστού κόσμου, ενώ άλλες προσεγγίζουν το θέμα επιτρέποντας την ύπαρξη πληροφορίας που δεν ορίζεται σαφώς στα μοντέλα εισόδου.

Τέλος, έχει ενδιαφέρον η πράξη της σύνθεσης, όταν οι αντιστοιχίσεις μεταξύ των στοιχείων των δύο μοντέλων εισόδου, δεν ορίζονται μόνο σαν ένα προς ένα. Σε μια τέτοια προσέγγιση, μπορούν να ορίζονται διαφόρων τύπων σχέσεις μεταξύ των στοιχείων των μοντέλων, όπως για παράδειγμα σχέσεις πολλά προς ένα, όπου μια οντότητα αναπαρίσταται με ένα στοιχείο στο ένα μοντέλο και με ένα σύνολο από στοιχεία στο δεύτερο ή σχέσεις όπου η παρουσία ενός στοιχείου στο ένα μοντέλο προϋποθέτει ή αποκλείει την παρουσία κάποιου άλλου στο δεύτερο μοντέλο, κτλ.

Για της ανάγκες αυτής της εργασίας, θα θεωρηθεί πως η σχέση μεταξύ των μοντέλων είναι δεδομένη από πριν. Επιπλέον, θα θεωρηθεί πως τα μοντέλα είναι δοσμένα σε κοινή γλώσσα, πως δεν υπάρχουν αντιφάσεις μεταξύ τους, πως χρησιμοποιείται κοινό λεξιλόγιο και πως οι σχέσεις μεταξύ των μοντέλων είναι ένα προς ένα αντιστοιχίες.

Στην παράγραφο *Επέκταση των θεμελιωδών πράξεων*, θα παρουσιαστέι μια επέκταση της πράξης της σύνθεσης λαμβάνοντας υπ' όψη πιο περίπλοκες σχέσεις μεταξύ των μοντέλων αλλά και κανόνες που ρυθμίζουν αυτές τις σχέσεις.

3.1.2 Διαφορά μοντέλων

Ορίσαμε νωρίτερα την πράξη της αντιστοίχισης μοντέλων, η οποία από δύο μοντέλα εισόδου παράγει σαν έξοδο μια σχέση μεταξύ τους, δηλαδή ένα σύνολο από αντιστοιχίσεις μεταξύ των στοιχείων τους.Όταν χρησιμοποιούμε την πράξη της αντιστοίχισης, το ζητούμενο είναι να δούμε πού τα δύο μοντέλα μοιάζουν, ποια στοιχεία είναι κοινά και πώς τα κοινά στοιχεία ταιριάζουν ανάμεσα στα μοντέλα.

Όμως, αν αντί για τις ομοιότητες προσανατολιστούμε στο να δούμε πού ακριβώς και με ποιον τρόπο διαφέρουν τα δύο μοντέλα, μπορούμε να κατασκευάσουμε μια αλληλουχία από επεμβάσεις (edits), με την οποία το ένα μοντέλο εισόδου μπορεί να μετασχηματιστεί στο άλλο. Αυτή είναι η πράξη της διαφοράς.

Με άλλα λόγια, η πράξη της διαφοράς μοντέλων, παίρνει σαν είσοδο δύο μοντέλα, το ένα από τα οποία θεωρείται σαν σημείο εκκίνησης και το άλλο σαν σημείο κατάληξης. Με την πράξη της διαφοράς, αναδεικνύουμε εκείνα τα διαδοχικά βήματα με τα οποία το μοντέλο εκκίνησης μετασχηματίζεται στο μοντέλο κατάληξης. Αυτή η αλληλουχία από επεμβάσεις, ονομάζεται *μετασχηματισμός*.

Για την πράξη της διαφοράς, όπως και για την πράξη της σύνθεσης, απαιτείται ο προσδιορισμός της σχέσης μεταξύ των στοιχείων των δύο μοντέλων εισόδου. Η σχέση αυτή μπορεί να παρέχεται από την βοηθητική πράξη της αντιστοίχισης, όπως και στην περίπτωση της σύνθεσης. Σε πρακτικά πλαίσια, η πράξη της διαφοράς συνήθως χρησιμοποιείται για να μελετηθεί η εξέλιξη ενός συγκεκριμένου μοντέλου στο χρόνο. Συνεπώς ανοίγεται το πεδίο για χρησιμοποίηση τεχνικών από το πεδίο της Ανιχνευσιμότητας Μοντέλων (model traceability), όπου το ζητούμενο είναι ακριβώς η ανίχνευση της σχέσης μεταξύ στοιχείων ενός συστήματος μεταξύ των οποίων υπάρχει σχέση προγόνου-απογόνου [35]. Ταυτόχρονα, μπορούν να χρησιμοποιηθούν και οι τεχνικές αντιστοίχησης που περιγράφηκαν στην προηγούμενη παράγραφο.

Στα πλαίσια της εργασίας, θα θεωρήσουμε πως μαζί με τα μοντέλα εισόδου, παρέχεται στην πράξη της διαφοράς και ένα σύνολολο από αντιστοιχίσεις μεταξύ των στοιχείων τους, δηλαδή με την ορολογία που χρησιμοποιήθηκε προηγουμένως, μια σχέση. Επίσης, όπως και με την περίπτωση της σύνθεσης, θα θεωρηθεί πως τα μοντέλα είναι δοσμένα σε κοινή γλώσσα, πως δεν υπάρχουν αντιφάσεις μεταξύ τους και πως χρησιμοποιείται κοινό λεξιλόγιο.

Συνεπώς, χρησιμοποιώντας τους ίδιους συμβολισμούς με νωρίτερα, γράφουμε:

$$\text{διαφορά} : \text{μοντέλο} \times \text{μοντέλο} \times \text{σχέση} \rightarrow \text{μετασχηματισμός}$$

Αξίζει να παρατηρηθεί πως στην πράξη της διαφοράς, τα δύο μοντέλα εισόδου έχουν καθορισμένους ρόλους: η διαφορά βγάζει σαν έξοδο τον μετασχηματισμό από το μοντέλο εκκίνησης στο μοντέλο κατάληξης. Συνεπώς, ενώ η σειρά με την οποία τα μοντέλα εισόδου δηλώνονται στην πράξη της σύνθεσης δεν παίζει ρόλο και άρα ισχύει η αρχή της αντιμεταθετικότητας, στην πράξη της διαφοράς αυτό δεν ισχύει.

(Σημειώνεται πως αν η βοηθητική πράξη της αντιστοίχισης θεωρηθεί σαν μέρος της πράξης σύνθεσης, και αν θεωρηθεί πως το ένα από τα δύο μοντέλα είναι το μοντέλο αναφοράς για την επίλυση των αντιφάσεων που πιθανόν προκύψουν, τότε ούτε και για την σύνθεση ισχύει η αντιμεταθετικότητα. Στα πλάισια όμως της εργασίας, οι σχέσεις μεταξύ των μοντέλων εισόδου θεωρούνται δεδομένες

και η πράξη της αντιστοίχισης θεωρείται πως έχει εκτελεστεί ξεχωριστά από την πράξη της σύνθεσης).

Όπως αναφέρθηκε, η πράξη της διαφοράς έχει σαν έξοδο έναν μετασχηματισμό. Όπως αναφέρθηκε και στην σχετική παράγραφου του δεύτερου κεφαλαίου της εργασίας, η έννοια του μετασχηματισμού είναι κεντρική στην μοντελοκεντρική μηχανική. Στην γενική περίπτωση, ο μετασχηματισμός ορίστηκε σαν *μια διαδικασία η οποία παίρνει σαν είσοδο ένα μοντέλο και ακολουθώντας συγκεκριμένους κανόνες βγάζει σαν έξοδο κάποιο άλλο μοντέλο*.

Για τις ανάγκες του ορισμού της πράξης της διαφοράς μοντέλων, θα θεωρήσουμε πως οι κανόνες που ακολουθούνται στη διαδικασία του μετασχηματισμού συνίσταναι στην διαδοχική εφαρμογή συγκεκριμένου τύπου επεμβάσεων στο μοντέλο εισόδου. Τις επεμβάσεις αυτές τις θεωρούμε ατομικές και στοιχειώδεις, και αναφέρονται συγκεκριμένα σε μοντέλα τα οποία προδιαγράφονται με βάση το πρότυπο MOF. Σημειώνεται επίσης πως και οι ίδιες οι στοιχειώδεις επεμβάσεις δεν είναι κάτι διαφορετικό από μετασχηματισμοί μοντέλων.

Όπως αναφέρεται στο [36], αυτές οι επεμβάσεις μπορούν να είναι:

- **προσθήκη** κάποιου στοιχείου στο μοντέλο
- **διαγραφή** κάποιου στοιχείου από το μοντέλο
- **μετονομασία** κάποιου ονοματισμένου στοιχείου του μοντέλου
- **μετακίνηση** κάποιου στοιχείου από το εσωτερικό κάποιου στοιχείου κιβώτιο [1] σε κάποιο άλλο. Προαιρετικά, αν το μετακινούμενο στοιχείο είναι το ίδιο στοιχείο κιβώτιο, μπορεί να έχουμε μετακίνηση και των περιεχομένων του.
- **αντιγραφή** ενός στοιχείου σε κάποιο άλλο στοιχείο κιβώτιο
- **τροποποίηση** κάποιας ιδιότητας ενός στοιχείου του μοντέλου (πχ την πολλαπλότητα κάποιας σχέσης)

Η πράξη της διαφοράς, όπως αναφέρεται και στο [4], είναι παρόμοιας φιλοσοφίας με το πρόγραμμα diff που παρέχεται στο λειτουργικό σύστημα UNIX.

[1] Ονοματίζω στοιχείο κιβώτιο (container element), ένα στοιχείο ενός μοντέλου που μπορεί να περιέχει άλλα στοιχεία, όπως για παράδειγμα τα πακέτα (packages) ή τις κλάσεις που περιέχουν εσωτερικές κλάσεις.

Όμως, ενώ το πρόγραμμα diff αναζητά λεκτικού τύπου διαφορές μεταξύ δύο αρχείων κειμένου, κάθε υλοποίηση της πράξης της διαφοράς μοντέλων θα πρέπει να καταλήγει σε ένα πρόγραμμα το οποίο να είναι σε θέση να αναγνωρίζει οντότητες και τις δομικές σχέσεις μεταξύ αυτών των οντοτήτων [5].

Να σημειωθεί σε αυτό το σημείο πως ,όπως μπορεί να διαφανεί και από την φύση των επεμβάσεων που δεχόμαστε πως είναι δυνατόν να παρατηρηθούν σε κάποια εκτέλεση της πράξης της διαφοράς, οι αντιστοιχίσεις που περιλαμβάνονται στη δεδομένη κάθε φορά σχέση μεταξύ των μοντέλων εισόδου, δεν απαιτείται να είναι όσο αυστηρά ορισμένες όσο περιγράφηκε στην περίπτωση της πράξης της σύνθεσης. Για την ακρίβεια, για να υπάρχει η δυνατότητα να αναγνωρίζουμε την επέμβαση της αντιγραφής, πρέπει να μπορούμε να χειριζόμαστε σχέσεις με πολλαπλότητες μεγαλύτερες του ένα.

Για τα πλαίσια της πράξης της διαγραφής δεν χρειάζεται να αλλάξουμε κάτι στον μέχρι τώρα τρόπο με τον οποίο διαχειριζόμαστε αυτές τις περιπτώσεις. Όπως όμως θα δούμε αργότερα, τέτοιου είδους αντιστοιχίσεις μπορούν να εισαχθούν και για την περίπτωση της πράξης της σύνθεσης, αρκεί να δίδεται μαζί και ένας σαφής τρόπος με τον οποίο θα είναι δυνατή η διαχείριση τους υπό το πρίσμα της σύνθεσης μοντέλων.

3.1.3 Άλλες πράξεις

Εκτός από τις πράξεις της σύνθεσης και της διαφοράς μοντέλων που αναφέραμε και που είναι το κύριο αντικείμεο της εργασίας, πρέπει να αναφέρουμε ότι μπορούν να οριστούν και άλλες πράξεις μεταξύ μοντέλων. Για λόγους πληρότητας, ακολουθεί μια συνοπτική παρουσίαση αυτών των πράξεων όπως περιγράφονται στο [4]. Οι πράξεις αυτές είναι η Αντιστοίχιση, η Διαμέριση, η Προβολή, ο Έλεγχος Ιδιότητας, ο Έλεγχος Συνέπειας, η Εφαρμογή και η Προώθηση.

Καταρχήν, είναι δυνατόν να θεωρηθεί η πράξη της αντιστοίχισης όχι σαν βοηθητική, αλλά σαν θεμελιώδης. Όπως αναφέρθηκε και νωρίτερα, η πράξη της αντιστοίχισης παίρνει σαν είσοδο δύο μοντέλα και έχει σαν έξοδο μια σχέση, δηλαδή ένα σύνολο από αντιστοιχίσεις μεταξύ των μοντέλων εισόδου. Στην γενική περίπτωση, η πραξη της αντιστοίχισης μπορεί να έχει σαν έξοδο ένα σύνολο από πιθανές σχέσεις μεταξύ των μοντέλων, οι οποίες στη συνέχεια μπορούν να χρησιμοποιηθούν για την εξερεύνηση από τον μηχανικό των διαφόρων τρόπων να συνδιάσει τα μοντέλα. Γράφουμε:

$$\text{αντιστοίχιση} : \text{μοντέλο} \times \text{μοντέλο} \rightarrow \text{σχέση}$$

Μπορεί επίσης να οριστεί η πράξη της διαμέρισης (split). Πρακτικά, η διαμέριση είναι η αντίστροφη πράξη της σύνθεσης: παίρνει σαν είσοδο κάποιο μοντέλο και βγάζει στην έξοδο δύο μοντέλα και την μεταξύ τους σχέση. Στην γενική περίπτωση, μπορεί μαζί με το μοντέλο εισόδου να της δίδεται και κάποιου είδους κριτήριο διαμέρισης, με βάση το οποίο θα είναι δυνατός ο καθορισμός του τρόπου διαμέρισης. Η σχέση μεταξύ των μοντέλων εξόδου είναι δυνατόν να θεωρηθεί πως αναπαριστά διαπροσωπείες (interfaces) μεταξύ τους. Γράφουμε:

$$\text{διαμέριση} : \text{μοντέλο} \rightarrow \text{μοντέλο} \times \text{μοντέλο} \times \text{σχέση}$$

ή εναλλακτικά

$$\text{διαμέριση} : \text{μοντέλο} \times \text{κριτήριο} \rightarrow \text{μοντέλο} \times \text{μοντέλο} \times \text{σχέση}$$

Συγγενική της πράξης της διαμέρισης είναι η πράξη της προβολής (slice). Η προβολή παίρνει σαν είσοδο ένα μοντέλο και ένα κριτήριο και στην έξοδο της παράγει ένα μοντέλο αποτελούμενο από εκείνα τα στοιχεία του μοντέλου εισόδου που ικανοποιούν το κριτήριο εισόδου. Γράφουμε:

$$\text{προβολή} : \text{μοντέλο} \times \text{κριτήριο} \rightarrow \text{μοντέλο}$$

Στο [4] ορίζονται επίσης πράξεις που αφορούν τον έλεγχο ιδιοτήτων των μοντέλων και των σχέσεων μεταξύ τους. Ορίζεται η πράξη του ελέγχου ιδιότητας που επιστρέφει μια αληθοτιμή ανάλογα με το αν ένα μοντέλο ικανοποιεί μια ιδιότητα και η πράξη του ελέγχου συνέπειας όπου επιστρέφεται μια αληθοτιμή ανάλογα με το κατά πόσο δύο μοντέλα αντιστοιχούν μεταξύ τους με τρόπο συμβατό με μια δεδομένη σχέση εισόδου. Για τις δύο πράξεις γράφουμε:

$$\text{έλεγχος\_ιδιότητας} : \text{μοντέλο} \times \text{ιδιότητα} \rightarrow \text{αληθοτιμή}$$

$$\text{έλεγχος\_συνέπειας} : \text{μοντέλο} \times \text{μοντέλο} \times \text{σχέση} \rightarrow \text{αληθοτιμή}$$

Ορίζονται επιπλέον και δύο πράξεις που αφορούν στην εφαρμογή μετασχηματισμών σε μοντέλα. Η πράξη της εφαρμογής απλά εφαρμόζει σε ένα μοντέλο εισόδου έναν μετασχηματισμό σαν αυτόν που παράγει η πράξη της διαφοράς. Η πράξη της προώθησης παίρνει σαν είσοδο έναν μετασχηματισμό που έχει παραχθεί για ένα μοντέλο εισόδου και ένα δεύτερο μοντέλο εισόδου, μαζί με την μεταξύ τους σχέση και εφαρμόζει στο δεύτερο μοντέλο έναν αντίστοιχο μετασχηματισμό, τέτοιο ώστε να παραχθεί ένα μοντέλο συμβατό με εκείνο που θα προέκυπτε από την εφαρμογή του μετασχηματισμού εισόδου στο πρώτο μοντέλο. Γράφουμε:

$$\text{εφαρμογή} : \text{μοντέλο} \times \text{μετασχηματισμός} \rightarrow \text{μοντέλο}$$

$$\text{προώθηση} : \text{μετασχηματισμός} \times \text{μοντέλο} \times \text{μοντέλο} \times \text{σχέση} \rightarrow \text{μοντέλο}$$

3.2 Αλγόριθμοι θεμελιωδών πράξεων

3.2.1 Σύνθεση μοντέλων

Η διαδικασία που ακολουθείται για την εκτέλεση της πράξης της σύνθεσης μοντέλων και που παρουσιάζεται στα πλαίσια της εργασίας βασίζεται σε αυτήν που περιγράφεται στο [33]. Η θεωρητική βάση των αλγόριθμων που συναποτελούν την υλοποίηση της σύνθεσης, είναι τόσο η θεωρία γραφημάτων, όσο και η θεωρία κατηγοριών.

Όπως έχει αναφερθεί νωρίτερα, αν το μεταμοντέλο στο υποίο υπακούουν τα μοντέλα εισόδου αναπαρίσταται από τον γράφο τύπων $\mathcal{M}$, τότε κάθε μοντέλο περιγράφεται από ένα ζεύγος $\langle G, t\rangle$ όπου G είναι ένας κατευθυνόμενος γράφος και t ένας ομομορφισμός $t : G \to \mathcal{M}$ ο οποίος αναθέτει έναν τύπο σε κάθε στοιχείο του G. Στην περίπτωση μας, ο γράφος τύπων είναι το πρότυπο EMOF.

Επίσης, επαναλαμβάνεται πως είναι $G = (N, E, src, tgt)$ όπου N είναι ένα σύνολο κόμβων, E είναι ένα σύνολο ακμών και $src, tgt : E \to N$ είναι συναρτήσεις που επιστρέφουν το κόμβο αρχής και τέλους αντίστοιχα για κάποια δεδομένη κατευθυνόμενη ακμή του γράφου.

Συνεπώς, για την πράξη της σύνθεσης έχουμε δύο μοντέλα εισόδου,

$$\langle (N_A, E_A, src_A, tgt_A), t_A\rangle$$

$$\langle (N_B, E_B, src_B, tgt_B), t_B\rangle$$

και μια δεδομένη σχέση μεταξύ των δύο μοντέλων εισόδου, η οποία εκφράζεται από έναν ομομορφισμό με τύπους

$$\underline{h} : \langle G_A, t_A\rangle \to \langle G_B, t_B\rangle$$

Συνεπώς, η σύνθεση δύο γράφων είναι κατά βάση θέμα ένωσης των συνόλων N και E με τρόπο τέτοιο που να διατηρούνται οι εξαρτήσεις που ορίζονται από τους ομομορφισμούς με τύπους

$$\langle \underline{h_{node}} : N_A \to N_B, \underline{h_{edge}} : E_A \to E_B\rangle$$

Προχωρούμε παρουσιάζοντας ένα ένα τα βήματα της διαδικασίας.

Σύνθεση συνόλων

Για την σύνθεση συνόλων, μας ενδιαφέρει καταρχήν να μπορούμε να έχουμε έναν τρόπο να συμβολίζουμε τα στοιχεία που προέρχονται από το κάθε σύνολο εισόδου, χωρίς να έχουμε συγκρούσεις ονομάτων. Για τον λόγο αυτό ορίζουμε την έννοια της *μη συμβιβαστής ένωσης* (disjoint union) ως εξής:

Η μη συμβιβαστή ενωση μιας οικογένειας συνόλων $S_1, S_2, \ldots, S_n$, συμβολίζεται ως

$$S_1 \uplus S_2 \uplus \ldots \uplus S_n$$

και ισοδυναμεί με

$$S_1 \times \{1\} \cup S_2 \times \{2\} \cup \ldots \cup S_n \times \{n\}$$

Για συντομία, κατασκευάζουμε την μη συμβιβαστή ενωση προσθέτοντας στα στοιχεία κάθε συνόλου το όνομα του συνόλου σαν δείκτη και κάνοντας την απλή ένωση τους στη συνέχεια. Για παράδειγμα: Αν $S_1 = \{x, y\}$ και $S_2 = \{x, t\}$, γράφουμε ότι $S_1 \uplus S_2$ είναι

$$\{x_{S_1}, y_{S_1}, x_{S_2}, t_{S_2}\}$$

αντί του πιο αναλυτικού

$$\{(x, 1), (y, 1), (x, 2), (t, 2)\}$$

Έχοντας λοιπόν έναν τρόπο αναπαράσασης των στοιχείων των συνόλων εισόδου χωρίς συγκρούσεις ονομάτων, προχωρούμε στον αλγόριθμο σύνθεσης συνόλων. Για την σύνθεση των συνόλων A και B, ακολουθούμε την διαδικασία που περιγράφεται στο [33]. Σύμφωνα με την περιγραφόμενη διαδικασία, αρχικά κατασκευάζουμε έναν μη κατευθυνόμενο γράφο U με κενό σύνολο ακμών $E_U = \emptyset$, του οποίου το σύνολο κόμβων N_U είναι η μη συμβιβαστή ένωση των δύο συνόλων εισόδου $A \uplus B$.

Στη συνέχεια, προσθέτουμε μη κατευθυνόμενες ακμές μεταξύ των κόμβων του γράφου που συνδέονται με την σχέση αντιστοίχισης. Κάνοντας το ίδιο για όλους τους κόμβους του γράφου, καταλήγουμε σε ένα σύνολο από συνεκτικούς υπογράφους του αρχικού γράφου. Κάθε τέτοιος συνεκτικός υπογράφος αναπαριστά ένα στοιχείο του τελικού συνόλου.

Το σύνολο εξόδου P, το οποίο είναι η σύνθεση των δύο συνόλων εισόδου, κατασκευάζεται θεωρώντας ένα στοιχείο για κάθε συνεκτικό υπογράφο του U που κατασκευάστηκε νωρίτερα.

Επειδή μας ενδιαφέρει να μπορούμε να γνωρίζουμε σε ποια στοιχεία των αρχικών συνόλων αντιστοιχεί το κάθε στοιχείο του ενιαίου συνόλου P, επεκτείνουμε τον αλγόριθμο του [33], με σκοπό να κατασκευάσουμε τους ομομορφισμούς ανιχνευσιμότητας $h_A : A \to P$ και $h_B : B \to P$. Κάθε φορά που προστίθεται ένα στοιχείο στο σύνολο εξόδου εξ' αιτίας κάποιου συνεκτικού υπογράφου του γράφου U ελέγχουμε τα στοιχεία του υπογράφου και συμπληρώνουμε τον αντίστοιχο ομομορφισμό (h_A ή h_B) ανάλογα με την προέλευση των στοιχείων.

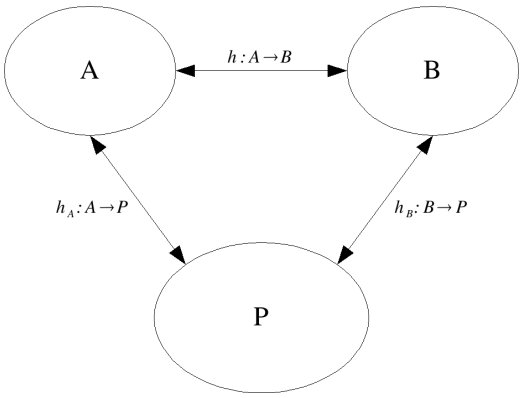
Συνεπώς, για δυο σύνολα A, B και έναν μορφισμό h μεταξύ τους, ο αλγόριθμος σύνθεσης συνόλων είναι:

Algorithm 1 Σύνθεση-Συνόλων(A, B, h):

Require: Έστω $U = (N_U, E_U) = (A \uplus B, \emptyset)$ μη κατευθυνόμενος γράφος
Require: Έστω $P = \emptyset$
for all $x \in A \uplus B$ **do**
  if $\exists y \in A \uplus B$ τέτοιο ώστε $y = h(x)$ **then**
    Πρόσθεσε στο E_U την μη κατευθυνόμενη ακμή (x, y)
  end if
end for
for all G τέτοιο ώστε να είναι συνεκτικός υπογράφος του U **do**
  Πρόσθεσε ένα στοιχείο p στο σύνολο P
  for all κόμβος $e \in G$ **do**
    if $e \in A$ **then**
      $h_A(e) \Leftarrow p$
    else
      $h_{\mathrm{B}}(e) \Leftarrow p$
    end if
  end for
end for
return P, h_A, h_B

Σύνθεση γράφων

Για να συνθέσουμε δύο γράφους G_A και G_B που συνδέονται μεταξύ τους με έναν ομομορφισμό $h : G_A \to G_B$, αρχικά κάνουμε την σύνθεση των συνόλων των κόμβων και των ακμών τους με βάση τον αλγόριθμο Σύνθεσης Συνόλων (1) που περιγράψαμε νωρίτερα. Στη συνέχεια, έχοντας κατασκευάσει τα σύνολα ακμών και κόμβων του ενωμένου γράφου, καθορίζουμε ποιος είναι ο



Σχήμα 3.1: Ομομορφισμοί αντιστοίχισης και ανιχνευσιμότητας.

κόμβος αρχής και τέλους για κάθε ακμή.

Με άλλα λόγια, αν ο γράφος εξόδου αυτού του σταδίου της διαδικασίας είναι ο $P = (N_P, E_P, src_P, tgt_P)$, αρχικά υπολογίζουμε το σύνολο N_P σαν τη σύνθεση των συνόλων N_A και N_B με χρήση της συνάρτησης κόμβων h_{node} του ομομορφισμού h. Αντίστοιχα υπολογίζουμε το σύνολο E_P συνθέτοντας τα σύνολα E_A και E_B με χρήση της συνάρτησης ακμών h_{edge} του ομομορφισμού h. Παράλληλα με τους δύο υπολογισμούς, βρίσκουμε και τους αντίστοιχους ομομορφισμούς ανιχνευσιμότητας $h_{N_A}, h_{N_B}, h_{E_A}$ και h_{E_B}.

Έχοντας υπολογίσει αυτά τα δύο σύνολα, για να ολοκληρωθεί η περιγραφή του κατευθυνομενου γράφου, μένει ο καθορισμός των συναρτήσεων src_P και tgt_P. Τον καθορισμό αυτόν τον κάνουμε στοιχείο προς στοιχείο, ώστε τελικά για κάθε στοιχείο στο πεδίο ορισμού της κάθε συνάρτησης να αντιστοιχιστεί ένα στοιχείο στο πεδίο τιμών της.

Για τον σκοπό αυτό δουλεύουμε ως εξης: ελέγχουμε ένα ένα τα στοιχεία του συνόλου ακμών E_P του ενιαίου γράφου. Για κάθε μία ακμή e του E_P, ελέγχουμε αρχικά αν υπάρχει κάποια ακμή q στο σύνολο ακμών E_A του πρώτου γράφου που να αντιστοιχεί σε αυτήν. Η αντιστοίχιση αυτή ελέγχεται με χρήση του ομομορφισμού ανιχνευσιμότητας h_{E_A}, ελέγχοντας δηλαδή αν $h_{E_A}(q) = e$.

Αν βρεθεί τέτοια ακμή q στον πρώτο γράφο, τότε με χρήση του ομομορφισμού ανιχνευσιμότητας $h_{\mathrm{N_A}}$, οι συναρτήσεις src και tgt του ενιαίου γράφου λαμβάνουν για την ακμή e σαν τιμές τις αντίστοιχες των τιμών των συναρτήσεων src και tgt του γράφου G_A για την ακμή q. Με άλλα λόγια, κάνουμε: $src(e) \Leftarrow h_{N_A}(src(q))$ και $tgt(e) \Leftarrow h_{N_A}(tgt(q))$

Αν δεν βρεθεί η αντίστοιχη ακμή q στον πρώτο γράφο, τότε την αναζητού-

με στον δεύτερο γράφο χρησιμοποιώντας τον ομομορφισμό ανιχνευσιμότητας h_{E_B} και αναθέτουμε τις κατάλληλες τιμές για την ακμή e στις συναρτήσεις src και tgt του ενιαίου γράφου G_P ακολουθώντας την ίδια διαδικασία που αναφέρθηκε παραπάνω, με χρήση του ομομορφισμού ανιχνευσιμότητας h_{N_B}.

Το ότι αρχικά η αντίστοιχη της κάθε ακμής του ενιαίου γράφου αναζητείται στον γράφο G_A, και μόνο όταν δεν βρεθεί προχωρούμε στο να την αναζητήσουμε στον γράφο G_B δεν επηρεάζει την αντιμεταθετικότητα της πράξης αφού όπως έχουμε αναφέρει νωρίτερα, θεωρούμε σαν δεδομένο πως δεν υπάρχουν αντιφάσεις μεταξύ των μοντέλων και πως οι αντιστοιχίες μεταξύ των στοιχείων τους είναι ένα προς ένα.

Σε κάθε περίπτωση, ακολουθώντας επαναληπτικά την διαδικασία που περιγράφηκε παραπάνω για όλες τις ακμές στο σύνολο ακμών E_P του ενιαίου γράφου G_P, καθορίζουμε στοιχείο προς στοιχείο τις τιμές των συναρτήσεων src και tgt του. Συνεπώς, σταδιακά θα οριστεί πλήρως η αντιστοίχιση του πεδίου ορισμού τους με το πεδίο τιμών τους και άρα μπορούμε να θεωρήσουμε πως με την ολοκλήρωση της διαδικασίας θα τις έχουμε ορίσει πλήρως, οπότε πλέον θα έχει οριστεί πλήρως και ο ενιαίος γράφος, που είναι και το ζητούμενο σε αυτό το στάδιο της διαδικασίας της σύνθεσης μοντέλων.

Για λόγους διατήρησης της ανιχνευσιμότητας, μαζί με τον ενιαίο γράφο, ο αλγόριθμος αφήνει στην έξοδο του και τους ομομορφισμούς ανιχνευσιμότητας $h_{N_A}, h_{N_B}, h_{E_A}$ και h_{E_B}. Συνολικά για την σύνθεση των γράφων G_A, G_B που συνδέονται μέσω του ομομορφισμού $h = \langle h_{node}, h_{edge} \rangle$, θα έχουμε τον αλγόριθμο:

Algorithm 2 Σύνθεση-Γράφων (G_A, G_B, h):

Require: Έστω γράφος $P = (N_P, E_P, src_P, tgt_P)$
1: $\langle N_P, h_{N_A}, h_{N_B} \rangle \Leftarrow$ Σύνθεση-Συνόλων$(N_{G_\text{A}}, N_{G_\text{B}}, h_{node})$
2: $\langle E_P, h_{E_A}, h_{E_B} \rangle \Leftarrow$ Σύνθεση-Συνόλων$(E_{G_\text{A}}, E_{G_\text{B}}, h_{edge})$
3: **for all** $e \in E_P$ **do**
4: **if** $q \in E_A$ τέτοιο ώστε $h_{E_A}(q) = e$ **then**
5: $src(e) \Leftarrow h_{N_A}(src(q))$
6: $tgt(e) \Leftarrow h_{N_A}(tgt(q))$
7: **else**
8: {Έχουμε $q \in E_B$ τέτοιο ώστε $h_{E_B}(q) = e$}
9: $src(e) \Leftarrow h_{N_B}(src(q))$
10: $tgt(e) \Leftarrow h_{N_B}(tgt(q))$
11: **end if**
12: **end for**
13: **return** $P, h_{N_A}, h_{N_B}, h_{E_A}, h_{E_B}$

Σύνθεση μοντέλων

Για να συνθέσουμε δυο μοντέλα $M_A = \langle G_A, t_A \rangle$ και $M_B = \langle G_B, t_B \rangle$ τα οποία συνδέονται μέσω ενός ομομορφισμού με τύπους $\underline{h} : \langle G_\text{A}, t_\text{A} \rangle \rightarrow \langle G_\text{B}, t_\text{B} \rangle$ σε ένα ενιαίο μοντέλο $M_P = \langle G_P, t_P \rangle$, δουλέυουμε ως εξής:

Αρχικά συνθέτουμε τους γράφους G_A και G_B σε έναν ενιαίο γράφο G_P, χρησιμοποιώντας τον αλγόριθμο Σύνθεσης Γράφων (2) που περιγράψαμε νωρίτερα. Στη συνέχεια, καθορίζουμε την αντιστοίχιση τύπων t_P του ενιαίου μοντέλου.

Για τον καθορισμό της αντιστοίχησης τύπων t_P του ενιαίου μοντέλου, εξετάζουμε ένα ένα τα στοιχεία του ενιαίου γράφου G_P. Για κάθε ένα τέτοιο στοιχείο x του γράφου (είτε αυτό είναι ακμή είτε κόμβος), αναζητούμε το αντίστοιχο του αρχικά στον γράφο G_A και αν δεν βρεθεί εκεί, στον γράφο G_B, χρησιμοποιώντας τους ομομορφισμούς ανιχνευσιμότητας. Ο τύπος του στοιχείου x καθορίζεται να είναι ίδιος με τον τύπο του στοιχείου του μοντέλου εισόδου από το οποίο προήλθε.

Σημειώνεται πως η σειρά με την οποία αναζητούμε τα στοιχεία του ενιαίου μοντέλου στα μοντέλα εισόδου δεν παίζει ρόλο, καθώς τα δύο μοντέλα εισόδου συνδέονται μέσω του ομομορφισμού με τύπους $\underline{h}$, γνωρίζουμε ότι κάθε στοιχείο του ενός γράφου θα έχει αντιστοιχιοτεί σε κάποιο στοιχείο του ίδιου τύπου στον άλλο γράφο. Θα ισχύει δηλαδή $t_\text{B}(\underline{h}(x)) = t_\text{A}(x), x \in G_A$

Με αυτόν τον τρόπο ορίζουμε την αντιστοίχιση τύπων t_P στοιχείο προς στοιχείο, ώστε τελικά κάθε τιμή στο πεδίο ορισμού της, δηλαδή την ένωση $N_P \cup E_P$ του συνόλου κόμβων του ενιαίου μοντέλου με το σύνολο ακμών του, να έχει αντιστοιχιστεί σε μία τιμή στο πεδίο τιμών της, δηλαδή τον γράφο τύπων $\mathcal{M}$, ο οποίος στα πλαίσια αυτής της εργασίας ισοδυναμεί με το μεταμοντέλο που περιγράφεται από το βασικό MOF (Essential MOF, EMOF).

Αν επιθυμούμε στο αποτέλεσμα της πράξης της σύνθεσης μοντέλων να περιλαμβάνονται και πληροφορίες που να αφορούν την ανιχνευσιμότητα των στοιχείων του ενιαίου μοντέλου, μπορούμε με μια ελάχιστη τροποποίηση της εντολής επιστροφής του αλγόριθμου μας να παίρνουμε στην έξοδο του και τους ομομορφισμούς ανιχνευσιμότητας $h_{N_A}, h_{N_B}, h_{E_A}$ και h_{E_B}, οι οποίοι μπορεί να αποδειχτούν χρήσιμοι για περαιτέρω επεξεργασία του ενιαίου μοντέλου ή για την περίπτωση που θα είναι επιθυμητή η μελέτη της εξέλιξης του, κτλ. Για τις ανάγκες της εργασίας, κάτι τέτοιο δεν είναι απαραίτητο, οπότε στην έξοδο του αλγορίθμου μας απλά εμφανίζεται το ενιαίο μοντέλο.

Συνεπώς, για την σύνθεση των μοντέλων $M_A = \langle G_A, t_A \rangle$ και $M_B = \langle G_B, t_B \rangle$ τα οποία συνδέονται μέσω ενός ομομορφισμού με τύπους $\underline{h} : \langle G_{\mathrm{A}}, t_{\mathrm{A}} \rangle \rightarrow \langle G_{\mathrm{B}}, t_{\mathrm{B}} \rangle$ έχουμε τον αλγόριθμο:

Algorithm 3 Σύνθεση-Μοντέλων ($\langle G_A, t_A\rangle, \langle G_B, t_B\rangle, \underline{h}$):

Require: Έστω μοντέλο $P = \langle G_P = (N_P, E_P, src_P, tgt_P), t_P\rangle$

$\langle G_P, h_{N_A}, h_{N_B}, h_{E_A}, h_{E_B}\rangle \Leftarrow$ Σύνθεση-Γράφων($G_\mathrm{A}, G_\mathrm{B}, \underline{h}$)
for all $n \in N_P$ **do**
  if $q \in N_A$ τέτοιο ώστε $h_{N_A}(q) = n$ **then**
    $t_P(n) \Leftarrow t_A(q)$
  else
    {Έχουμε $q \in N_B$ τέτοιο ώστε $h_{N_B}(q) = n$}
    $t_P(n) \Leftarrow t_B(q)$
  end if
end for
for all $e \in E_P$ **do**
  if $q \in E_A$ τέτοιο ώστε $h_{E_A}(q) = e$ **then**
    $t_P(e) \Leftarrow t_A(q)$
  else
    {Έχουμε $q \in E_B$ τέτοιο ώστε $h_{E_B}(q) = e$}
    $t_P(e) \Leftarrow t_B(q)$
  end if
end for
return P

3.2.2 Διαφορά μοντέλων

Για την υλοποίηση της πράξης της διαφοράς μοντέλων, θα βασιστούμε στην διαδικασία που περιγράφεται στο [36]. Πρόκειται για τον αλγόριθμο UMLDiff$_{cld}$, ο οποίος είναι σχεδιασμένος ώστε να υπολογίζει την διαφορά μεταξύ δυο UML διαγραμμάτων κλάσης (class diagrams, εξ' ου και το *cld* στο όνομα του αλγορίθμου).

Να σημειωθεί σε αυτό το σημείο πως τα μοντέλα εισόδου της πράξης της διαφοράς, δεν είναι κατ' ανάγκη UML διαγράμματα κλάσης. Όμως, όπως έχει ειπωθεί, τα μοντέλα εισόδου είναι μοντέλα τα οποία έχουν εκφραστεί σύμφωνα με το πρότυπο MOF ή έστω υπακούουν σε κάποιο μεταμοντέλο το οποίο έχει προδιαγραφεί με MOF και συνεπώς τα στιγμιότυπα του μπορούν επίσης να γραφούν στην γλώσσα του MOF.

Επιπλέον, όπως αναφέρεται στο [10], το πλήρες MOF (CMOF), απλά χρησιμοποιεί τον φορμαλισμό της UML, ενώ το βασικό MOF (EMOF) είναι ακόμα απλούστερο (βλέπε και την παρουσίαση του μεταμοντέλου του EMOF στο κεφάλαιο 2). Συνεπώς, ένας αλγόριθμος όπως το UMLDiff$_{cld}$, μπορεί να χρησιμοποιηθεί και για μοντέλα MOF.

Για καλύτερη προσαρμογή σε μοντέλα MOF, θα κάνουμε μερικές τροποποιήσεις στον αλγόριθμο και την τροποποιημένη αυτή παραλλαγή του αλγόριθμου UMLDiff$_{cld}$ θα την ονομάσουμε MOFDiff.

Γενική περιγραφή του UMLDiff$_{cld}$

Ο αλγόριθμος UMLDiff$_{cld}$ προσαντατολίζεται στο να καταγράφει τις διαφορές που μπορούν να παρατηρηθούν αναφορικά με τα παρακάτω στοιχεία των διαγραμμάτων κλάσεων:

- Πακέτα (packages)
- Κλάσεις (classes)
- Γενικεύσεις (generalizations)
- Πεδία (attributes)
- Σχέσεις (associations)
- Μεθόδους (operations)

- Παράμετροι (parameters)

Ο αλγόριθμος, βασίζεται στην παρατήρηση πως τα στοιχεία στα διαγράμματα, οργανώνονται σε επίπεδα. Βασική είναι εδώ η έννοια ότι κάποιο στοιχείο του μοντέλου περιέχει άλλα στοιχεία. Ονοματίζουμε ένα τέτοιο στοιχείο, *στοιχείο κιβώτιο* (container element). Όπως μπορεί κανείς να παρατηρήσει και στον ορισμό της γλώσσας του MOF στο δεύτερο κεφάλαιο της εργασίας, ένα μοντέλο συναπαρτίζεται από πακέτα, τα οποία περιέχουν κλάσεις, οι οποίες περιέχουν πεδία και μεθόδους, ενώ συνδέονται μεταξύ τους με σχέσεις και γενικεύσεις.

Ο UMLDiff$_{cld}$ ενεργεί κατά επίπεδα. Τα στοιχεία των δύο μοντέλων εισόδου προσπελάυνονται κατά πλάτος (breadth first) και για κάθε επίπεδο, ενεργοποιείται μια συνάρτηση η οποία αναλαμβάνει να αντιστοιχίσει τα στοιχεία των δύο μοντέλων χρησιμοποιώντας τεχνικές με τις οποίες ποσοτικοποιείται η αντιστοίχιση.

Με τη χρήση της κατά πλάτος προσέγγισης, επιτυγχάνεται καλύτερη αντιστοίχιση μεταξύ των στοιχείων καθώς τα στοιχεία κάθε επιπεδου ελέγχονται μαζί ενώ μια κατά βάθος προσέγγιση θα εξαντλούσε πρώτα ένα στοιχείο κιβώτιο πριν προχωρήσει στο επόμενο, χάνοντας έτσι πιθανές αντιστοιχίσεις. Επίσης θεωρείται πως μια από τις συνηθέστερες επεμβάσεις σε ένα μοντέλο είναι η μετακίνηση στοιχείων από ένα στοιχείο κιβώτιο σε κάποιο άλλο και άρα η κατά πλάτος προσέγγιση έχει καλύτερες πιθανότητες να την ανιχνέυσει.

Σημειώνεται πως ο UMLDiff$_{cld}$ περιλαμβάνει μια βοηθητική διαδικασία αντιστοίχισης, η οποία περιληπτικά περιγράφηκε παραπάνω. Όμως, όπως αναφέρθηκε και νωρίτερα, για τις ανάγκες της εργασίας, θα θεωρήσουμε πως η αντιστοίχιση μεταξύ των μοντέλων εισόδου έιναι δεδομένη κατά την εκκίνηση του αλγορίθμου, έχοντας προσφερθεί σαν είσοδος μαζί με τα δύο μοντέλα εισόδου. Με αυτόν τον τρόπο, μένει ανοιχτή η δυνατότητα να χρησιμοποιηθούν οι διαφορετικές τεχνικές αντιστοίχισης των στοιχείων των μοντέλων εισόδου, που έχουν αναφερθεί στα προηγούμενα.

Αναγνώριση επεμβάσεων

Ανάλογα με τις αντιστοιχίσεις μεταξύ των στοιχείων των μοντέλων εισόδου, ο UMLDiff$_{cld}$ αναγνωρίζει και τις επεμβάσεις που έχουν απαιτηθεί για την ύπαρξη μιας δεδομένη αντιστοίχισης:

- Η μη ύπαρξη αντιστοίχισης κάποιου στοιχείου του μοντέλου εκκίνησης

στο μοντέλο κατάληξης σηματοδοτεί μια επέμβαση αφαίρεσης από το μοντέλο αυτού του στοιχείου.

- Η μη ύπαρξη αντιστοίχισης κάποιου στοιχείου του μοντέλου κατάληξης στο μοντέλο εκκίνησης σηματοδοτεί μια επέμβαση πρόθεσης στο μοντέλο αυτού του στοιχείου.

- Η ύπαρξη πολλαπλών αντιστοιχήσεων μεταξύ ενός στοιχείου του μοντέλου εκκίνησης και στοιχείων του μοντέλου κατάληξης σηματοδοτεί επεμβάσεις αντιγραφής.

- Αν τα αντιστοιχισμένα στοιχεία έχουν διαφορετικά ονόματα στα δύο μοντέλα εισόδου, αυτό σημαίνει πως έχει υπάρξει μια επέμβαση μετονομασίας

- Αν τα αντιστοιχισμένα στοιχεία περιέχονται σε διαφορετικά στοιχεία κιβώτια στα δύο μοντέλα εισόδου, αυτό σημαίνει πως έχει υπάρξει μια επέμβαση μετακίνησης.

- Αν κάποιο στοιχείο στο μοντέλο εισόδου διαφέρει από το αντιστοιχισμένο του στο μοντέλο εξόδου, αυτό σηματοτεί μια επέμβαση τροποποίησης.

Με βάση αυτές τις παραδοχές, ο αλγόριθμος εξετάζει ένα ένα τα στοιχεία κάθε επιπέδου του μοντέλου εξόδου. Για κάθε στοιχείο, αρχικά ελέγχεται αν υπάρχει αντιστοίχιση του από το μοντέλο εισόδου και ανάλογα με το αν και ποια είναι αυτή αναγνωρίζεται αν έχει υπάρξει κάποια επέμβαση στο μοντέλο.

Ο αλγόριθμος MOFDiff

Όπως αναφέρθηκε νωρίτερα, ο αλγόριθμος UMLDiff$_{cld}$ εξετάζει τα στοιχεία του μοντέλου ανά επίπεδο με σκοπό την καλύτερη εκτέλεση της δικής του εσωτερικής διαδικασίας αντιστοίχισης. Μιας και στα πλαίσια της εργασίας θεωρείται πως η ανεύρεση της αντιστοιχίσης των στοιχείων μεταξύ των δύο μοντέλων εισόδου λαμβάνεται έτοιμη από κάποιο εξωτερικό εργαλείο, αυτή η διαδικασία αντιστοίχισης θα μπορούσε να να παρακαμφθεί.

Θα ήταν δηλαδή δυνατόν να τροποποιήσουμε τον αλγόριθμο με τέτοιο τρόπο ώστε να παραβλέπεται αυτή η σταδιακή, κατά πλάτος εξέταση των στοιχείων. Θεωρώντας όμως πως είναι δυνατόν η αντιστοιχίες μεταξύ των στοιχείων να υπολογίζονται από το πρόγραμμα υλοποίησης την ώρα που ζητούνται, και όχι

όλες μαζί εξ' αρχής, και για χάρη της διαλειτουργικότητας, θα διατηρήσουμε την κατά πλάτος προσέγγιση. Άλλωστε, αυτό σημαίνει πως στην χειρότερη περίπτωση απλώς θα αλλάξει η σειρά με την οποία θα ελέγχονται τα στοιχεία.

Καθώς τα μοντέλα τα οποία θα θέλουμε να συγκρίνουμε είναι μοντέλα MOF, δεν έχει νόημα να συγκρίνουμε τις σχέσεις και τις γενικεύσεις (όπως τις διαχωρίζει ο UMLDiff$_{cld}$), καθώς στα μοντέλα MOF, αυτές αναπαρίστανται σαν στιγμιότυπα της MOF μετακλάσης EMOF::Class.

Συνεπώς, στα επόμενα θα περιγραφεί μια ελαφρώς τροποποιημένη εκδοχή του UMLDiff$_{cld}$ η οποία παρακάμπτει την ανακάλυψη των αντιστοιχίσεων μεταξύ των στοιχείων των μοντέλων, αφού θα θεωρείται δεδομένη η ύπαρξη της σχέσης μεταξύ τους, ενώ θα είναι και προσαρμοσμένη στις ανάγκες της σύγκρισης μοντέλων MOF. Αυτήν την τροποποιημένη εκδοχή θα την ονομάσουμε MOFDiff.

Ο πλήρης αλγόριθμος παρουσιάζεται παρακάτω. Ο αλγόριθμος παίρνει σαν είσοδο τα δύο μοντέλα $M_{\text{αρχ}} = \langle G_{\text{αρχ}}, t_{\text{αρχ}} \rangle$ και $M_{\text{τελ}} = \langle G_{\text{τελ}}, t_{\text{τελ}} \rangle$, όπου τα $t_{\text{αρχ}}, t_{\text{τελ}}$ είναι αντιστοιχίσεις τύπων, καθώς και την σχέση μεταξύ των δύο μοντέλων με τη μορφή του ομομορφισμού με τύπους $\underline{h}$.

Για λόγους οικονομίας χώρου, έχει χρησιμοποιηθεί μια υπορουτίνα με το όνομα *Επεξεργασία*, η οποία καλείται διαδοχικά για τους έξι τύπους στοιχείων που ενδιαφέρουν. Στην υπορουτίνα δίδονται σαν παράμετροι ο εκάστοτε τύπος τα στοιχεία του οποίου θέλουμε να εξετάσουμε, τα μοντέλα εισόδου και η μεταξύ τους σχέση, καθώς και το σύνολο επεμβάσεων R στο οποίο αποθηκεύουμε όσες επεμβάσεις εντοπιστούν.

Οι επεμβάσεις αυτές αναπαρίστανται ως:

- Αντιγραφή($e_{\text{αρχ}}, e_{\text{τελ}}$)
- Μετονομασία($e_{\text{αρχ}}, e_{\text{τελ}}$)
- Τροποποίηση($e_{\text{αρχ}}, e_{\text{τελ}}$)
- Μετακίνηση($e_{\text{αρχ}}, e_{\text{τελ}}$)
- Προσθήκη($e_{\text{τελ}}$)
- Διαγραφή($e_{\text{αρχ}}$)

Επιπλέον, θεωρείται πως υπάρχουν οι βοηθητικές υπορουτίνες:

- Όνομα(e)
- Τιμή(e)
- Στοιχείο-Κιβώτιο(e)

Η υπορουτίνα Όνομα(e) κάνει το προφανές. Για μοντέλα MOF αυτό σημαίνει πως επιστρέφεται η τιμή του πεδίου `name`. Η υπορουτίνα Τιμή(e) επιστρέφει την κατάσταση του στοιχείου e, δηλαδή την τιμή των πεδίων του. Για μοντέλα MOF, σε αυτήν την περίπτωση μας ενδιαφέρουν τα υπόλοιπα πεδία, πέραν του ονόματος. Τέλος η υπορουτίνα Στοιχείο-Κιβώτιο(e) επιστρέφει το στοιχείο του μοντέλου το οποίο περιέχει το στοιχείο e.

Algorithm 4 MOFDiff ($\langle G_{\alpha\rho\chi}, t_{\alpha\rho\chi}\rangle, \langle G_{\tau\varepsilon\lambda}, t_{\tau\varepsilon\lambda}\rangle, \underline{h}$):

Require: Σύνολο επεμβάσεων $R = \emptyset$

1: Επεξεργασία(EMOF::Package, $\langle G_{\alpha\rho\chi}, t_{\alpha\rho\chi}\rangle, \langle G_{\tau\varepsilon\lambda}, t_{\tau\varepsilon\lambda}\rangle, \underline{h}, R$)
2: Επεξεργασία(EMOF::Class, $\langle G_{\alpha\rho\chi}, t_{\alpha\rho\chi}\rangle, \langle G_{\tau\varepsilon\lambda}, t_{\tau\varepsilon\lambda}\rangle, \underline{h}, R$)
3: Επεξεργασία(EMOF::Property, $\langle G_{\alpha\rho\chi}, t_{\alpha\rho\chi}\rangle, \langle G_{\tau\varepsilon\lambda}, t_{\tau\varepsilon\lambda}\rangle, \underline{h}, R$)
4: Επεξεργασία(EMOF::Operation, $\langle G_{\alpha\rho\chi}, t_{\alpha\rho\chi}\rangle, \langle G_{\tau\varepsilon\lambda}, t_{\tau\varepsilon\lambda}\rangle, \underline{h}, R$)
5: **return** R

Procedure 5 Επεξεργασία($Type, \langle G_{\alpha\rho\chi}, t_{\alpha\rho\chi}\rangle, \langle G_{\tau\varepsilon\lambda}, t_{\tau\varepsilon\lambda}\rangle, \underline{h}, R$):

1: **for all** $e_{\tau\varepsilon\lambda} \in G_{\tau\varepsilon\lambda}$ τέτοια ώστε $t_{\tau\varepsilon\lambda}(e_{\tau\varepsilon\lambda}) = Type$ **do**
2: **if** $\exists e_{\alpha\rho\chi} \in G_{\alpha\rho\chi}$ τέτοιο ώστε $\underline{h}(e_{\tau\varepsilon\lambda}) = e_{\alpha\rho\chi}$ **then**
3: **if** $\exists e'_{\tau\varepsilon\lambda} \neq e_{\tau\varepsilon\lambda} \in G_{\tau\varepsilon\lambda}$ τέτοιο ώστε $\underline{h}(e_{\alpha\rho\chi}) = e'_{\tau\varepsilon\lambda}$ **then**
4: Πρόσθεσε στο R την επέμβαση *Αντιγραφή*$(e_{\alpha\rho\chi}, e_{\tau\varepsilon\lambda})$
5: **end if**
6: **if** Όνομα$(e_{\alpha\rho\chi}) \neq$ Όνομα$(e_{\tau\varepsilon\lambda})$ **then**
7: Πρόσθεσε στο R την επέμβαση *Μετονομασία*$(e_{\alpha\rho\chi}, e_{\tau\varepsilon\lambda})$
8: **end if**
9: **if** Τιμή$(e_{\alpha\rho\chi}) \neq$ Τιμή$(e_{\tau\varepsilon\lambda})$ **then**
10: Πρόσθεσε στο R την επέμβαση *Τροποποίηση*$(e_{\alpha\rho\chi}, e_{\tau\varepsilon\lambda})$
11: **end if**
12: **if** Στοιχείο-Κιβώτιο$(e_{\alpha\rho\chi}) \neq$Στοιχείο-Κιβώτιο$(e_{\tau\varepsilon\lambda})$ **then**
13: Πρόσθεσε στο R την επέμβαση *Μετακίνηση*$(e_{\alpha\rho\chi}, e_{\tau\varepsilon\lambda})$
14: **end if**
15: **else**
16: Πρόσθεσε στο R την επέμβαση *Προσθήκη*$(e_{\tau\varepsilon\lambda})$
17: **end if**
18: **end for**
19: **for all** $e_{\alpha\rho\chi} \in G_{\alpha\rho\chi}$ τέτοια ώστε $t_{\alpha\rho\chi}(e_{\alpha\rho\chi}) = Type$ **do**
20: **if** $\nexists e_{\tau\varepsilon\lambda} \in G_{\tau\varepsilon\lambda}$ τέτοιο ώστε $\underline{h}(e_{\alpha\rho\chi}) = e_{\tau\varepsilon\lambda}$ **then**
21: Πρόσθεσε στο R την επέμβαση *Διαγραφή*$(e_{\alpha\rho\chi})$
22: **end if**
23: **end for**

3.3 Επέκταση πράξης της σύνθεσης μοντέλων

Προχωρούμε τώρα στην παρουσίαση μιας επέκτασης των θεμελιωδών πράξεων, με σκοπό την παρουσίαση τόσο της γενικότερης λειτουργικότητας τους, όσο και της ευελιξίας τους στο να τροποποιηθούν ώστε να ανταποκρίνονται σε πρακτικές διαφοροποιήσεις της σημασιολογίας και του πεδίου εφαρμογής.

Θα επικεντρωθούμε στην επέκταση της πράξης της σύνθεσης. Για το σκοπό αυτό θα ορίσουμε συνοπτικά έναν τρόπο για την χρήση γενκευμένων αντιστοιχίσεων και κανόνων αντιστοίχισης, με σκοπό να καταστεί δυνατή η έκφραση πιο πολύπλοκων τρόπων με τους οποίους είναι δυνατόν να σχετίζονται μεταξύ τους τα μοντέλα εισόδου.

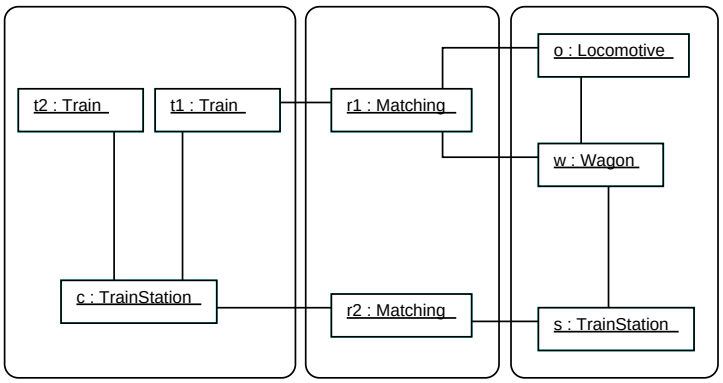
3.3.1 Σύνθετες σχέσεις αντιστοίχισης μοντέλων

Η πράξη της σύνθεσης, όπως ορίστηκε νωρίτερα, λαμβάνει σαν είσοδο, εκτός από τα δύο προς επεξεργασία μοντέλα και ένα σύνολο από αντιστοιχίσεις μεταξύ των στοιχείων τους το οποίο το ονοματίσαμε σαν *σχέση*. Αναφέραμε νωρίτερα πως θεωρούμε πως οι αντιστοιχίσεις μεταξύ των στοιχείων των μοντέλων θα ήταν σχέσεις ένα προς ένα αντιστοιχίας.

Αυτό στην πράξη σημαίνει ο μόνος τρόπος με τον οποίο ένα στοιχείο από το ένα μοντέλο μπορέι να αντιστοιχιστεί σε κάποιο του άλλου, είναι σχέση της ισοδυναμίας. Με άλλα λόγια, το σημασιολογικό περιεχόμενο της απλής ένα προς ένα αντιστοίχισης είναι πως κάποια οντότητα αναπαρίσταται με ένα στοιχείο στο πρώτο μοντέλο και με κάποιο άλλο στοιχείο στο δεύτερο.

Όμως είναι δυνατόν να έχουμε πιο περίπλοκες σχέσεις μεταξύ των στοιχείων δύο μοντέλων. Είναι για παράδειγμα δυνατόν μια οντότητα που αναπαρίσταται με ένα στοιχείο σε κάποιο μοντέλο να αναπαρίσταται από ένα σύνολο στοιχείων σε ένα άλλο μοντέλο. Συνεπώς έχει νόημα να προσπαθήσουμε να συστηματοποιήσουμε έναν τρόπο να χειριζόμαστε σχέσεις μεταξύ των στοιχείων των μοντέλων που δεν είναι ένα προς ένα.

Επίσης, αναφέραμε νωρίτερα πως σε πρακτικές εφαρμογές, πολλές φορές τα δύο μοντέλα που προορίζονται για σύνθεση είναι αντιφατικά. Μέχρι τώρα, θεωρήσαμε πως τα μοντέλα εισόδου δεν εμπεριέχουν αντιφάσεις. Με άλλα λόγια, θεωρήθηκε πως η όποια σημασιολογική ερμηνία των μοντέλων και της σχέσης μεταξύ τους έχει γίνει από πριν και πως οι όποιες αντιφάσεις έχουν



Σχήμα 3.2: Παράδειγμα μοντέλων που συνδέονται με πολλαπλή σχέση.

διορθωθεί.

Όμως στην γενική περίπτωση, μπορούμε να περιγράψουμε ένα περιβάλλον όπου οι σχέσεις μεταξύ των στοιχείων των συνόλων εκτός από τις αντιστοιχίσεις, εμπεριέχουν και σημασιολογικό φορτίο. Για παράδειγμα, είναι δυνατόν να συναντήσουμε περιπτώσεις όπου η παρουσία ενός στοιχείου στο ένα μοντέλο αποκλείει την παρουσία κάποιου άλλου στο δεύτερο μοντέλο. Πιο λεπτεπίλεπτες περιπτώσεις θα μπορούσαν να περιλαμβάνουν σημασιολογικές εξαρτήσεις μεταξύ των τιμών των πεδίων των στοιχείων των μοντέλων, δηλαδή εξαρτήσεις που να αφορούν την κατάσταση των μοντέλων. Για παράδειγμα, θα μπορούσε να υπάρχει ο περιορισμός ότι επιτρέπεται η σύνθεση δύο στοιχείων μόνο αν κάποιο πεδίο τους έχει μια δεδομένη τιμή.

Συνεπώς, θα προσπαθήσουμε να περιγράψουμε έναν τρόπο που να μπορεί να εκφράζει τέτοιου είδους σημασιολογικές εξαρτήσεις. Θα προσπαθήσουμε να δώσουμε τη δυνατότητα να εκφραστούν κανόνες που να ορίζουν τρόπους επίλυσης αντιφάσεων και σημασιολογικών εξαρτήσεων. Να σημειωθεί πως ο σκοπός δεν είναι ένα γενικό πλαίσιο επίλυσης αντιφάσεων.

Κανόνες

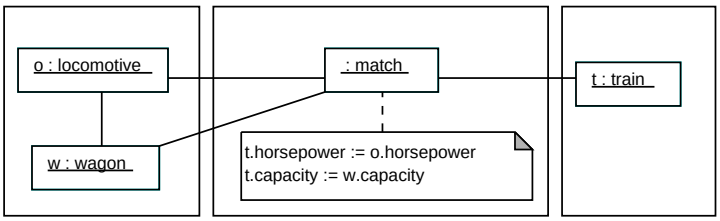
Για την περιγραφή των κανόνων θα χρησιμοποιήσουμε μια απλοποιημένη παραλλαγή των τριπλών γραμματικών γράφων. Όπως περιγράφεται στο [21], οι τριπλές γραμματικές γράφων (triple graph grammars, TGG) είναι μια τεχνική για να δηλώνονται αντιστοιχίσεις μεταξύ δύο μοντέλων. Οι αντιστοιχίσεις μπορούν να μετατραπούν από δηλώσεις σε λειτουργίες και άρα οι TGG μας δίνουν

ένα πλαίσιο που μπορεί να χρησιμοποιηθεί για τον μετασχηματισμό μοντέλων, όπως περιγράφεται στο [22].

Ένας κανόνας γραμμένος σε TGG αποτελείται από τρία μέρη. Στην αριστερή πλευρά ενός κανόνα περιγράφεται το μοτίβο (pattern) το οποίο βρίσκεται σε ένα μοντέλο εισόδου και το οποίο θα μετασχηματιστεί μέσω του κανόνα. Με την εφαρμογή του κανόνα, το μοτίβο που βρίσκεται στην αριστερή πλευρά του κανόνα μετασχηματίζεται στο μοτίβο το οποίο γράφεται στην δεξιά πλευρά του κανόνα. Οι αντιστοιχίσεις μεταξύ των δύο μοτίβων βρίσκονται στο κεντρικό μέρος του κανόνα.

Όπως περιγράφεται στο [21], οι κανόνες μπορούν να περιλαμβάνουν περιορισμούς, ώστε να περιγράφονται σχέσεις μεταξύ των τιμών των πεδίων των μοτίβων εισόδου και εξόδου. Επιπλέον, παρέχεται η δυνατότητα να περιγράφονται πληροφορίες που αφορούν τον περίγυρο των μοτίβων, αλλά και η δυνατότητα να περιγράφονται απαγορευμένες καταστάσεις, συνδέσεις και μοτίβα. Για λεπτομερή περιγραφή όλων των παραπάνω, ο αναγνώστης παρακαλείται να ανατρέξει στην βιβλιογραφία.

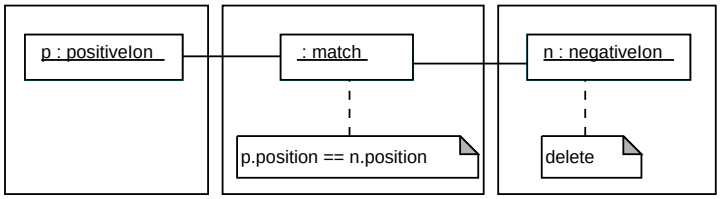
Παρακάτω φαίνεται ένα παράδειγμα ενός κανόνα TGG. Ο κανόνας περιγράφει πώς το μοτίβο ατμομηχανή-συνδεδεμένη-με-βαγόνι μετατρέπεται σε ένα ενιαίο τρένο. Ο περιορισμός (αναπαρίσταται με σχόλιο, όμοια με τον τρόπο συμβολισμού του [22]) καθορίζει πώς τα χαρακτηριστικά της ατμομηχανής και του βαγονιού θα μεταφερθούν στο τρένο.



Σχήμα 3.3: Παράδειγμα κανόνα TGG.

Επίσης, παρακάτω φαίνεται ένα παράδειγμα ενός κανόνα TGG με την περίπτωση μιας απαγορευμένης κατάστασης. Σύμφωνα με τον περιορισμό, αν το θετικό ιόν βρίσκεται στην ίδια θέση με το αρνητικό, τότε το αρνητικό αφαιρείται από το διάγραμμα. Ο περιορισμός αναπαρίσταται με σχόλιο όπως και πριν. Η εντολή διαγραφής του αρνητικού φορτίου αναπαρίσταται επίσης με σχόλιο.

Για τις ανάγκες της εργασίας, θα περιοριστούμε σε σχετικά απλούς κανόνες. Σημειώνεται όμως, πως οι τριπλές γραμματικές γράφων είναι ένα εξαιρετι-



Σχήμα 3.4: Παράδειγμα κανόνα TGG με απαγορευμένη κατάσταση.

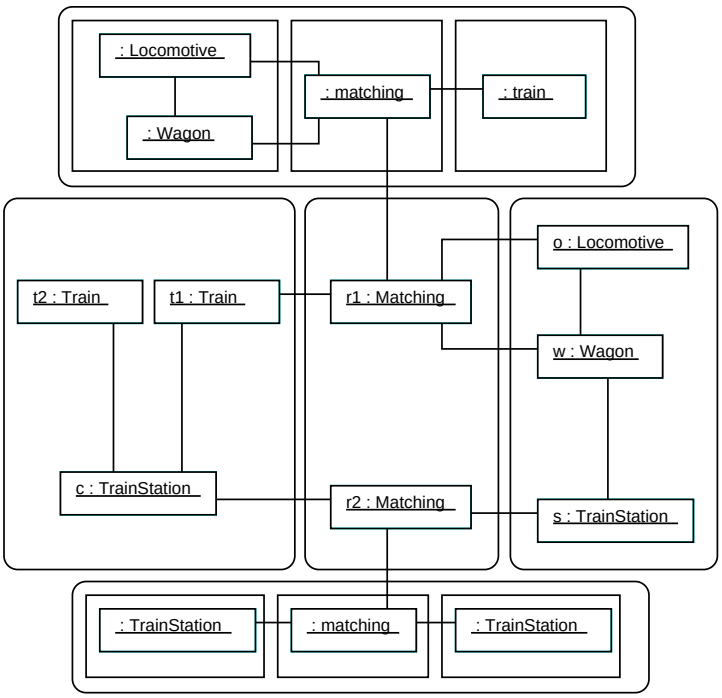
κά ισχυρό και ευέλικτο εργαλείο, το οποίο μπορεί να χρησιμοποιηθεί για να περιγράψει κανόνες πολύ πολυπλοκότερους.

3.3.2 Επεκτεταμένη σύνθεση μοντέλων με κανόνες αντιστοίχισης

Η βασική ιδέα για το πλαίσιο στο οποίο περιγράφουμε τους κανόνες είναι πως ο μηχανικός που ενδιαφέρεται να συνθέσει δυο μοντέλα, μπορεί να περιγράφει τις αντιστοιχίσεις μεταξύ τους σε δύο χρόνους. Σε πρώτο χρόνο, τα στοιχεία στα μοντέλα εισόδου μπορούν να αντιστοιχίζονται με σχέσεις μεταβλητής πολλαπλότητας (ένα προς ένα, ένα προς πολλά, πολλά προς ένα και πολλά προς πολλά). Σε δεύτερο χρόνο, ο μηχανικός καλείται να δώσει κανόνες οι οποίοι να περιγράφουν πώς θα πρέπει να γίνει ο χειρισμός των πολλαπλών σχέσεων.

Μπορούμε να δούμε την παραπάνω διαδικασία και αντίστροφα. Ο μηχανικός καθορίζει κανόνες οι οποίοι περιγράφουν ποιοι είναι οι επιτρεπτοί τρόποι να αντιστοιχίζονται τα στοιχεία των μοντέλων, καθώς και πώς θα γίνει η σύνθεση αν τα στοιχεία αντιστοιχιστούν με αυτόν τον τρόπο. Ακολούθως, οι αντιστοιχίσεις που περιλαμβάνονται στην σχέση μεταξύ των δύο μοντέλων εισόδου της πράξης της σύνθεσης αναγνωρίζονται σαν εφαρμογές των προδιαγεγραμένων κανόνων.

Τα παραπάνω δεν αποκλείουν την πιθανότητα να έχουμε μια αυτοματοποιημένη διαδικασία για την σύνθεση με κανόνες. Μπορούμε να θεωρήσουμε πως οι αντιστοιχίσεις παράγονται από κάποια αυτοματοποιημένη διαδικασία αντιστοίχισης. Το κρίσιμο σημείο για μια αυτοματοποιημένη υλοποίηση της διαδικασίας είναι ο τρόπος με τον οποίο αποφασίζεται ποιοι κανόνες θα εφαρμοστούν σε ποιες αντιστοιχίσεις. Μια λύση σε αυτό το πρόβλημα παρουσιάζεται στο [22], όπου υιοθετείται ένας μηχανισμός απόδοσης προτεραιοτήτων και ένας αλγόριθμος για την επιλογή της σειράς εφαρμογής των κανόνων. Στο [37], παρουσιάζεται μια άλλη μέθοδος που βασίζεται στον σταδιακό συγχρο-



Σχήμα 3.5: Παράδειγμα αναγνώρισης των αντιστοιχίσεων σαν εφαρμογές των κανόνων.

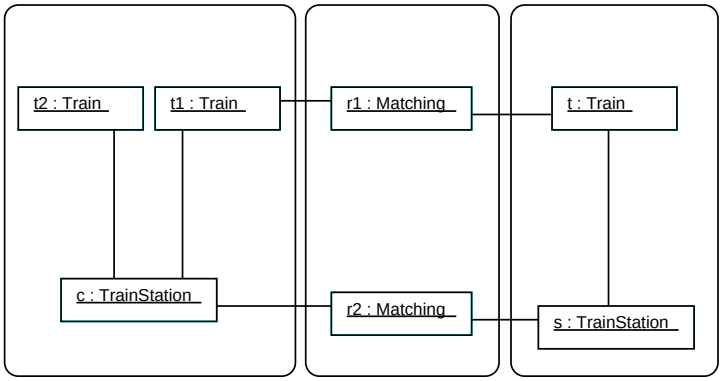
νισμό των μοντέλων εισόδου.

Η περιγραφή των τεχνικών αυτοματοποιημένης επιλογής και εφαρμογής των κανόνων ξεφευγει από τον ορίζοντα αυτης της εργασίας, και για αυτό δεν θα επεκταθούμε περισσότερο. Για τις ανάγκες της πράξης της σύνθεσης, θα θεωρήσουμε πως η επιλογή ποιος κανόνας θα εφαρμοστεί σε ποια αντιστοίχιση είναι δεδομένη. Θα μπορούσε να έχει είτε αποφασιστεί αυτόματα είτε χειροκίνητα. Η προσέγγιση αυτή είναι απλουστευτική, αφού για παράδειγμα, θα μπορούσε η εφαρμογή ενός κανόνα να αποτρέπει την εφαρμογή κάποιου άλλου κτλ). Πιο εκλεπτυσμένες και ισχυρές μεθόδοι παρουσιάζονται στην αναφερθείσα βιβλιογραφία.

Σημειώνουμε πως ο σκοπός της εφαρμογής των κανόνων είναι να μετατραπούν οι σχέσεις με πολλαπλότητα μεγαλύτερη του ένα σε ένα προς ένα αντιστοιχίσεις, ώστε να είναι δυνατή η σύνθεση των μετασχηματισμένων μοντέλων με τον αλγόριθμο (3). Συνεπώς, στον γράφο που μετασχηματίζεται από έναν κανόνα, το μοτίβο που αντιστοιχεί στο αριστερό μέρος του κανόνα μετατρέπεται στο μοτίβο που υπάρχει στο δεξιό μέρος του κανόνα, τα στοιχεία του οποίου αντιστοιχούν ένα προς ένα με στοιχεία του άλλου μοντέλου εισόδου. Άρα,

με την εφαρμογή ενός κανόνα, εκτός από το να μετασχηματίζουμε τον αντίστοιχο υπογράφο του μοντέλου, αναδημιουργούμε και τις ανάλογες ένα προς ένα αντιστοιχίσεις μεταξύ των μετασχηματισμένων μοντέλων εισόδου.

Έχοντας τα παραπάνω υπ' όψη, η σύνθεση μοντέλων στην γενικευμένη περίπτωση αποτελεί μια διαδικασία δύο βημάτων. Στο πρώτο βήμα, τα μοντέλα εισόδου μετασχηματίζονται με βάση τις δοσμένες αντιστοιχίσεις και τους επιλεγμένους κανόνες για κάθε αντιστοίχιση (ή ομάδα αντιστοιχίσεων). Αφού γίνει ο μετασχηματισμός των μοντέλων, το δεύτερο βήμα είναι η σύνθεση των μοντέλων με τον τρόπο που έχει περιγραφεί στα προηγούμενα.



Σχήμα 3.6: Τα μετασχηματισμένα μοντέλα εισόδου.

Συνεπώς έχουμε τις εξής τροποποιήσεις που αφορούν την σύνθεση:

- Η *σχέση* μεταξύ των δύο μοντέλων αποτελείται από ένα σύνολο αντιστοιχίσεων μεταξύ των μοντέλων, ένα σύνολο από κανόνες γραμμένους σύμφωνα με την τεχνική των τριπλών γραμματικών γράφων, καθώς και την επιλογή που αφορά στο ποιος κανόνας εφαρμόζεται σε ποια αντιστοίχιση.
- Η σύνθεση των δύο μοντέλων εισόδου είναι μια διαδικασία δύο βημάτων.
- Κατά τη διάρκεια του πρώτου βήματος της σύνθεσης τα μοντέλα εισόδου μετασχηματίζονται σύμφωνα με τις μεταξύ τους αντιστοιχίσεις. Για τον μετασχηματισμό των μοντέλων χρησιμοποιούνται οι κανόνες που περιλαμβάνονται στην *σχέση* μεταξύ των μοντέλων εισόδου.
- Κατά τη διάρκεια του δεύτερου βήματος της σύνθεσης, τα μετασχηματισμένα μοντέλα που έχουν προκύψει από την εφαρμογή των κανόνων συντίθενται με τον κλασικό αλγόριθμο σύνθεσης μοντέλων (3).

Έχουμε λοιπόν τον αλγόριθμο που παίρνει σαν παραμέτρους τα δύο μοντέλα εισόδου $\mathrm{M_A}$ και $\mathrm{M_B}$, τη μεταξύ τους γενικευμένη σχέση h, ένα σύνολο κανόνων R και την επιλογή της εφαρμογής των κανονων D.

Algorithm 6 Επεκτεταμένη-Σύνθεση-Μοντέλων (M_A, M_B, h, R, D):

Require: Έστω μοντέλα M'_A, M'_B, σχέση $h' : M'_A \rightarrow M'_B$
Require: Έστω μοντέλο M_P
1: Δημιούργησε τα μετασχηματισμένα μοντέλα M'_A, M'_B και τη μεταξύ τους σχέση $h' : M'_A \rightarrow M'_B$, εφαρμόζοντας τους κανόνες R στα μοντέλα M_A και M_B σύμφωνα με την επιλογή D και την γενικευμένη σχέση h
2: $M_P \Leftarrow$ Σύνθεση-Μοντέλων$(\mathrm{M'_A}, \mathrm{M'_B}, h')$
3: **return** M_P

3.4 Αλγεβρικές ιδιότητες των πράξεων

Έχοντας παρουσιάσει τις πράξεις της σύνθεσης και της διαφοράς για την άλγεβρα μοντέλων της εργασίας, έχει ενδιαφέρον να εξετάσουμε τις πράξεις από την πλευρά των αλγεβρικών ιδιοτήτων τους. Καθώς ο ορισμός της άλγεβρας αλλά και των πράξεων δεν έχει γίνει με αυστηρό αλλά με περιγραφικό τρόπο, ομοίως περιγραφικά θα γίνει και η προσέγγιση για τις υπό εξέταση ιδιότητες. Επίσης, η παρουσίαση δεν θα είναι διεξοδική και θα περιοριστεί σε κάποιες από τις ιδιότητες οι οποίες παρουσιάζουν ένα ενδιαφέρον σύμφωνα με το [4], με βάση το οποίο γίνεται και η συνολική παρουσίαση των ιδιοτήτων.

Πρέπει επίσης να σημειωθεί πως οι ιδιότητες αυτές έχουν σε έναν βαθμό χαρακτήρα προδιαγραφής. Υπό μια έννοια δηλαδή, ανεξάρτητα από την υλοποίηση των πράξεων που παρουσιάστηκε στα πλαίσια της εργασίας, οι ιδιότητες αυτές πρέπει να γίνονται αντιληπτές και σαν την αναπαράσταση επιθυμητών χαρακτηριστικών και συμπεριφορών τα οποία πρέπει να βρίσκονται σε μια περιγραφή και υλοποίηση των πράξεων διαχείρισης μοντέλων.

Ταυτοδυναμία

Η πράξη της σύνθεσης είναι ταυτοδύναμη, καθώς η σύνθεση ενός μοντέλου με τον εαυτό του έχει σαν αποτέλεσμα το ίδιο το μοντέλο. Να σημειωθεί πως έμμεσα γίνεται η παραδοχή πως η πράξη της αντιστοίχισης του μοντέλου με τον εαυτό του παράγει μια σχέση ταυτότητας.

Η πράξη της διαφοράς δεν είναι ταυτοδύναμη αφού εξ' ορισμού στην έξοδό της δεν παράγει μοντέλο αλλά μετασχηματισμό. Εξ'αλλου, η διαφορά ενός μοντέλου με τον εαυτό του, θα έχει σαν αποτέλεσμα έναν μετασχηματισμό εξόδου αποτελούμενο από μηδέν επέμβασεις.

Αντιμεταθετικότητα

Η πράξη της σύνθεσης είναι αντιμεταθετική, καθώς η σειρά με την οποία της τροφοδοτούνται τα μοντέλα εισόδου δεν παίζει ρόλο. Σημειώνεται πως αν η βοηθητική πράξη της αντιστοίχισης θεωρηθεί σαν μέρος της πράξης σύνθεσης, και αν θεωρηθεί πως το ένα από τα δύο μοντέλα είναι το μοντέλο αναφοράς για την επίλυση των αντιφάσεων που πιθανόν προκύψουν, τότε δεν ισχύει η αντιμεταθετικότητα.

Η πράξη της διαφοράς δεν είναι αντιμεταθετική αφού τα δύο μοντέλα εισόδου της έχουν καθορισμένους ρόλους: μοντέλο εκκίνησης και μοντέλο κατάληξης. Υπό προϋποθέσεις, το αποτέλεσμα της αντιμετάθεσης των μοντέλων εισόδου μπορεί να είναι ο αντίθετος μετασχηματισμός.

Προσεταιριστικότητα

Η προσαιτεριστικότητα για την πράξη της σύνθεσης εξαρτάται από τον τρόπο με τον οποίο δημιουργούνται οι σχέσεις σε κάθε βήμα, ο οποίος μάλιστα, σύμφωνα με το [4] είναι δυνατόν να είναι πολύ περίπλοκος. Στην πράξη, έχει νόημα ο ορισμός μιας πράξης σύνθεσης που να δέχεται ένα σύνολο μοντέλων και ένα σύνολο σχέσεων.

Για την πράξη της διαφοράς δεν ορίζεται προσαιτεριστικότητα αφού η διαφορά δύο μοντέλων δεν παράγει μοντέλο, αλλά μετασχηματισμό και βέβαι δεν ορίζεται διαφορά μεταξύ του μετασχηματισμού και ενός τρίτου μοντέλου.

Αντίστροφη πράξη

Η πράξη της σύνθεσης έχει αντίστροφη την διαμέριση, όταν η διαμέριση οριστεί με τον έναν από τους δύο τρόπους που περιγράφηκε στην παράγραφο 3.1.3, δηλαδή χωρίς να περιλαμβάνεται κάποιο κριτήριο εισόδου.

Η πράξη της διαφοράς δεν έχει αντίστροφη, αφού υπάρχουν άπειρα μοντέλα

τα οποία μπορούν να διαφέρουν κατά ένα δεδομένο μετασχηματισμό. Μπορεί όμως να κατασκευαστούν τα ελάχιστα μοντέλα τα οποία διαφέρουν κατά τον μετασχηματισμό εισόδου, αλλά η χρησιμότητα μιας τέτοιας πράξης δεν είναι προφανής.

Μονοτονικότητα

Στο [4] περιγράφεται η έννοια της μονοτονικότητας για την πράξη της σύνθεσης μοντέλων ως εξής: αν τα m'_1 και m'_2 αποτελούν την εξέλιξη των m_1 και m_2, τότε η σύνθεση τους θα είναι εξέλιξη της σύνθεσης των αρχικών μοντέλων. Αυτό συμβολίζεται ως

$$m_1 \preceq m'_1 \wedge m_2 \preceq m'_2 \Rightarrow \text{Σύνθεση}(m_1, m_2) \preceq \text{Σύνθεση}(m'_1, m'_2)$$

όπου το $\preceq$ να αναπαριστά μια διάταξη μεταξύ των μοντέλων η οποία καθορίζεται από τις ιδιότητες τους τις οποίες μπορεί να διατηρεί η πράξη της σύνθεσης.

Ολικότητα

Στο [4], η ολικότητα της πράξης της σύνθεσης μοντέλων ορίζεται λέγοντας πως για κάθε ζευγάρι μοντέλων, το αποτέλεσμα της σύνθεσης πρέπει να είναι αποδεκτό σαν μοντέλο. Η σύνθεση δεν μπορεί να θεωρηθεί ολική αν για κάποιους συνδυασμούς μοντέλων δεν παράγεται αποτέλεσμα. Η ιδιότητα αυτή συνδέεται με τη δυνατότητα διαχείρισης των αντιφάσεων που πιθανόν να υπάρχουν μεταξύ των μοντέλων εισόδου, αφού αν η πολιτική σύνθεσης είναι να απορρίπτονται τα αντιφατικά μοντέλα, τότε η σύνθεση δεν λειτουργεί πάντα. Αντίθετα, αν πάντα το αποτέλεσμα είναι ένα δόκιμο μοντέλο, αυτό το αποτέλεσμα μπορεί να αποδεικνύεται χρήσιμο σε συστήματα διαχείρισης αντιφάσεων, όπου το παραγόμενο μοντέλο μπορεί να χρησιμοποιηθεί σαν βάση για την ανακάλυψη και τη διαπραγμάτευση των αντιφάσεων.

Η πράξη της διαφοράς είναι ολική, καθώς για κάθε ζευγάρι μοντέλων είναι δυνατόν να δημιουργηθεί ένα σύνολο από επεμβάσεις που θα μετατρέπουν το ένα στο άλλο. Στην τεττριμένη περίπτωση, αυτές οι επεμβάσεις θα είναι απλά μια σειρά από διαγραφές ακολουθούμενη από μια σειρά απο προσθήκες.

Κεφάλαιο 4

Εφαρμογή

Προχωρούμε τώρα στην παρουσίαση μιας εφαρμογής της επεκτεταμένης πράξης της σύνθεσης. Το πεδίο στο οποίο θα εφαρμοστεί η μέθοδος είναι εκείνο των μοντέλων που υπακούουν στο πρότυπο Common Base Event (CBE), ένα πρότυπο για την προδιαγραφή ενός ενοποιημένου τρόπου ανταλλαγής δεδομένων που αφορούν σε γεγονότα που συμβαίνουν σε κατανεμημένα συστήματα.

Αρχικά θα κάνουμε μια σύντομη παρουσίαση μιας απλοποιημένης εκδοχής του CBE, παρουσιάζοντας τις βασικές σχεδιαστικές του αρχές και τον τρόπο με τον οποίο αυτό μπορεί να χρησιμοποιηθεί για την δημιουργία αρχείων καταγραφών. Στη συνέχεια θα προσεγγίσουμε το πρόβλημα της σύνθεσης δύο αρχείων καταγραφών δομημένων κατά CBE σε ένα ενιαίο αρχείο καταγραφών, χρησιμοποιώντας την επεκτεταμένη πράξη της σύνθεσης που περιγράψαμε στο προηγούμενο κεφάλαιο.

4.1 Το πρότυπο Common Base Event

Το Common Base Event είναι η υλοποίηση από μέρους της IBM του Web Event Format, ένα φορμά που αφορά την απεικόνιση γεγονότων και που προδιαγράφεται από το Web Services Distributed Management, ένα στάνταρ εγκεκριμένο από τον OASIS (Organization for the Advancement of Structured Information Standards) που αφορά στην διαχείριση και παρακολούθηση κατανεμημένων πόρων. Αναλυτικά, για το CBE, ο αναγνώστης μπορεί να αναφερθεί στο [38].

Για τις ανάγκες της εργασίας θα χρησιμοποιηθεί ένα απλοποιημένο υποσύ-

νολο του μοντέλου του CBE. Η χρήση του απλοποιημένου υποσυνόλου του CBE δεν επηρεάζει την συνολική εφαρμογή, καθώς επίδειξη με χρήση του πλήρους μοντέλου του CBE απλά θα προσέθετε περισσότερη πολυπλοκότητα χωρίς να προσφέρει τη δυνατότητα να παρουσιαστεί κάτι ιδιαίτερα ουσιαστικότερο όσον αφορά την επεκτεταμένη μέθοδο σύνθεσης μοντέλων.

Παρακάτω παρουσιάζεται το απλοποιημένο υποσύνολο του μοντέλου Common Base Event που θα χρησιμοποιηθεί στα πλαίσια της εργασίας. Για την κατασκευή του, βασιστήκαμε στο [39].

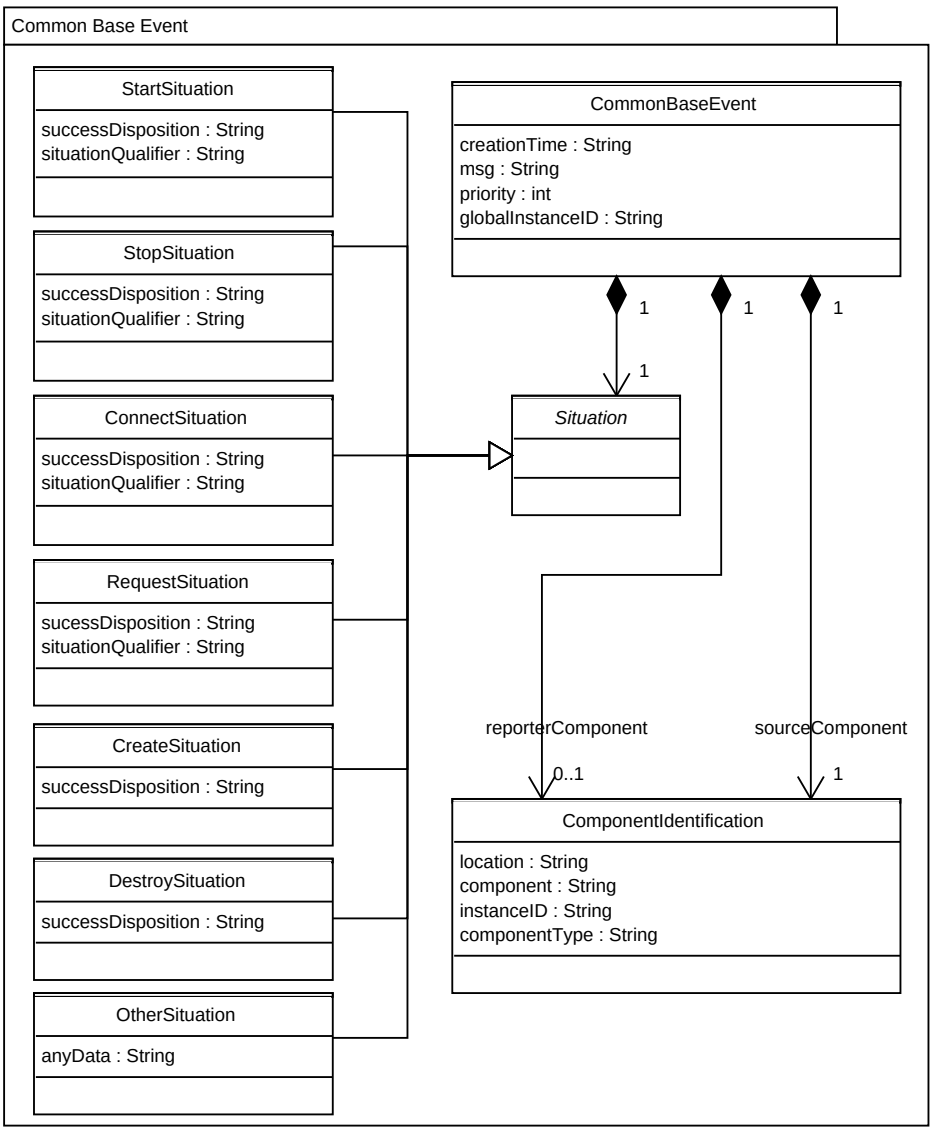
Το CBE όπως είπαμε είναι ένας τρόπος να αναπαρίστανται γεγονότα. Κάθε γεγονός, αποταλείται από μια τριπλέτα: το στοιχείο που κάνει την αναφορά μιας κατάστασης, το στοιχείο που προκάλεσε την κατάσταση και την κατάσταση την ίδια. Στο διάγραμμα κλάσεων παραπάνω, το γεγονός αναπαρίσταται από την κλάση `CommonBaseEvent` η οποία μέσω σχέσεων σύνθεσης αποτελείται από τα στοιχεία της τριπλέτας.

Το ένα μέρος από τα τρία που εμπλέκονται στον ορισμό του κατά CBE γεγονότος, είναι η ίδια η υπό μελέτη κατάσταση. Στο απλοποιημένο μοντέλο CBE που παρουσιάζουμε, αυτή η κατάσταση αναπαρίσταται από την αφηρημένη κλάση `Situation`. Τα επτά είδη καταστάσεων που συμπεριλαμβάνουμε στο απλοποιημένο CBE[1] είναι υποκλάσεις της `Situation`.

Κάθε μια από αυτές περιγράφει ένα συγκεκριμένο είδος κατάστασης που μπορεί να προκύψει κατά τη λειτουργία ενός συστήματος. Οι μεταβλητές τους περιγράφουν λεπτομέρειες της κατάστασης. Όπως περιγράφεται στο [39], οι μεταβλητές `successDisposition` στις διάφορες καταστάσεις παίρνουν τιμές `successful`/`unsuccessful`, οι μεταβλητές `situationQualifier` τιμές `initiated`/`completed`, ενώ η γενική κλάση `OtherSituation` περιλαμβάνει μια μεταβλητή `anyData` στην οποία μπορούν να προστεθούν δεδομένα για καταστάσεις που δεν ταιριάζουν με κάποια από τις υπόλοιπες προβλεπόμενες.

Τα δύο στοιχεία που εμπλέκονται στον ορισμό του γεγονότος, το στοιχείο που αναφέρει την κατάσταση και εκείνο που την προκαλεί είναι στιγμιότυπα της κλάσης `ComponentIdentification`. Το CBE προβλέπει πως αν το στοιχείο που προκαλεί μια κατάσταση είναι και εκείνο που την αναφέρει, τότε δεν πρέπει η ίδια πληροφορία να μεταφέρεται δύο φορές. Αυτό αντικατοπτρίζεται στο διάγραμμα με την πολλαπλότητα της σχέσης `reporterComponent`. Κατά τα άλλα η κλάση `ComponentIdentification` περιλαμβάνει μετα-

[1] Στο πλήρες πρότυπο CBE περιλαμβάνονται δώδεκα είδη καταστάσεων.



Σχήμα 4.1: Απλοποιημένο UML διάγραμμα κλάσεων του Common Base Event.

βλητές που περιγράφουν την διεύθυνση (πχ κάποιο URL), το όνομα, τον τύπο και τον περιγραφητή του εκάστοτε στοιχείου.

Το ίδιο το γεγονός είναι ένα στιγμιότυπο της κλάσης `CommonBaseEvent`. Επιπλέον των τριών συστατικών στοιχείων της τριπλέτας (κατάσταση, αναφέρον στοιχείο και υπαίτιο στοιχείο), η κλάση `CommonBaseEvent` περιλαμβάνει και μεταδεδομένα για το γεγονός, όπως ο περιγραφητής του, η ώρα κατά

την οποία δημιουργήθηκε, το αντίστοιχο μήνυμα κειμένου και η προτεραιότητα του.

4.1.1 Αρχεία καταγραφών

Έχοντας περιγράψει την προδιαγραφή του Common Base Event, ερχόμαστε τώρα να εξετάσουμε πώς αυτό μπορεί να χρησιμοποιηθεί για την καταγραφή γεγονότων.

Το πρότυπο CBE στοχεύει μεταξύ άλλων και στην ικανοποίηση της ανάγκης να υπάρχει ένας σταθερός τρόπος να αναπαρίστανται οι πληροφορίες που αφορούν γεγονότα. Με άλλα λόγια, το CBE στοχεύει στο να αποτελεί ένα πρότυπο σύμφωνα με το οποίο θα είναι δυνατόν να δημιουργούνται αρχεία καταγραφών γεγονότων (log files), με τέτοιο τρόπο ώστε να είναι δυνατή η διαλειτουργικότητα στη συνεργασία μεταξύ των διάφορων εφαρμογών.

Για το σκοπό αυτόν, το CBE ορίζει ένα XML σχήμα χρησιμοποιώντας το πρότυπο XML Schema Definition (XSD). Προϋπάρχοντα συστήματα καταγραφής μπορούν να προσαρμοστούν στο να παράγουν CBE με τη χρήση κάποιου μετατροπέα (adapter). Είναι λοιπόν εμφανές πως το πρότυπο CBE μπορεί να χρησιμοποίηθεί για την προδιαγραφή τόσο του περιεχομένου των αρχείων καταγραφών, όσο και της μορφής τους.

Οι διάφορες εφαρμογές μπορούν λοιπόν να παράγουν αρχεία καταγραφών που να είναι δομημένα με τρόπο που τα διάφορα γεγονότα να περιγράφονται σαν τριπλέτες σύμφωνα με την προδιαγραφή του CBE. Στην κειμενική μορφή των αρχείων καταγραφών, αυτά τα γεγονότα θα αναπαρίστανται σαν κάποια στοιχεία γραμμένα σε XML, που μέσω μιας προγραμματιστικης διεπαφης (application programming interface, API) βρίσκονται σε πλήρη αντιστοιχία με CBE αντικείμενα.

Συνεπώς, βλέπουμε ότι κάθε αρχείο καταγραφής πρόκειται ουσιαστικά για ένα μοντέλο αποτελούμενο από στοιχεία που είναι στιγμιότυπα των κλάσεων του CBE. Έχοντας λοιπόν τη δυνατότητα να αντιμετωπίζουμε τα αρχεία καταγραφών σαν μοντέλα (στιγμιότυπα του μοντέλου-προτύπου), μπορούμε να χρησιμοποιήσουμε τις τεχνικές που περιγράψαμε νωρίτερα, ώστε να τα διαχειριστούμε με συστηματικό τρόπο.

Θα επικεντρωθούμε στην σύνθεση αρχείων καταγραφών.

4.2 Σύνθεση αρχείων καταγραφών

Σε ένα περιβάλλον όπου συνυπάρχουν πολλές συνεργαζόμενες εφαρμογές είναι πολύ πιθανόν κάποιο συμβάν να παρατηρείται, να καταχωρείται και να αναφέρεται από διάφορες πηγές. Ακόμα περισσότερο, σε γενικές γραμμές είναι αναμενόμενο μια χρονική ακολουθία από γεγονότα να συμπεριλαμβάνεται σε διάφορα αρχεία καταγραφών που παράγονται από διάφορες εφαρμογές στο κατανεμημένο περιβάλλον.

Για τις ανάγκες της συντήρησης αλλά και της αποτελεσματικής παρακολούθησης των εκτελούμενων εφαρμογών και του περιβάλλοντος τους, είναι πολλές φορές αναγκαίο να μπορούμε να συνθέτουμε αυτά τα αρχεία καταγραφών σε ένα ενιαίο αρχείο.

Ακόμα όμως και αν οι διάφορες εφαρμογές παρατηρούν και καταγράφουν το ίδιο συμβάν, είναι δυνατόν να το περιγράφουν με διαφορετικό τρόπο. Μπορεί για παράδειγμα ένα συμβάν που περιγράφεται σαν ένα γεγονός από μια εφαρμογή να περιγράφεται σαν μια ολόκληρη χρονοακολουθία από γεγονότα από κάποια εφαρμογή επιφορτισμένη με τη λεπτομερή παρακολούθηση του στοιχείου που προκάλεσε το συμβάν.

Επιπλέον είναι δυνατόν σε πρακτικές εφαρμογές να παρουσιάζονται αντιφάσεις μεταξύ των καταγραφών. Τέτοιες αντιφάσεις μπορεί για παράδειγμα να προκαλούνται σε πραγματικές εφαρμογές από κάποιο λάθος (bug) του πηγαίου κώδικα (οπότε και η αναγνώριση των αντιφάσεων μπορεί να οδηγήσει και σε κάποια διόρθωση) ή μπορεί να προκαλείται από προβληματικό συγχρονισμό λόγω καθυστερήσεων του δικτύου κτλ.

Γίνεται λοιπόν εμφανές, πως για την σύνθεση αρχείων καταγραφών απαιτείται κάτι περισσότερο από την απλή σύνθεση μοντέλων που περιγράφηκε στην παράγραφο 3.2, αφού είναι απαραίτητο να υπάρχει τόσο ένας μηχανισμός να συμβιβάζονται οι πολλαπλές αντιστοιχίσεις, όσο και να μπορούν να κωδικοποιούνται σημασιολογικοί περιορισμοί που να μπορούν να χρησιμοποιηθούν για την επίλυση των αντιφάσεων κτλ. Μια τέτοια λειτουργικότητα προσφέρεται από την επεκτεταμένη πράξη της σύνθεσης με κανόνες αντιστοίχισης η οποία περιγράφηκε στην παράγραφο 3.3.

Στα επόμενα θα παρουσιάσουμε ένα παράδειγμα χρήσης της επεκτεταμένης πράξης της σύνθεσης με κανόνες αντιστοίχισης πάνω σε δύο μοντέλα εισόδου που αναπαριστούν αρχεία καταγραφών κατά CBE.

4.2.1 Τα μοντέλα εισόδου

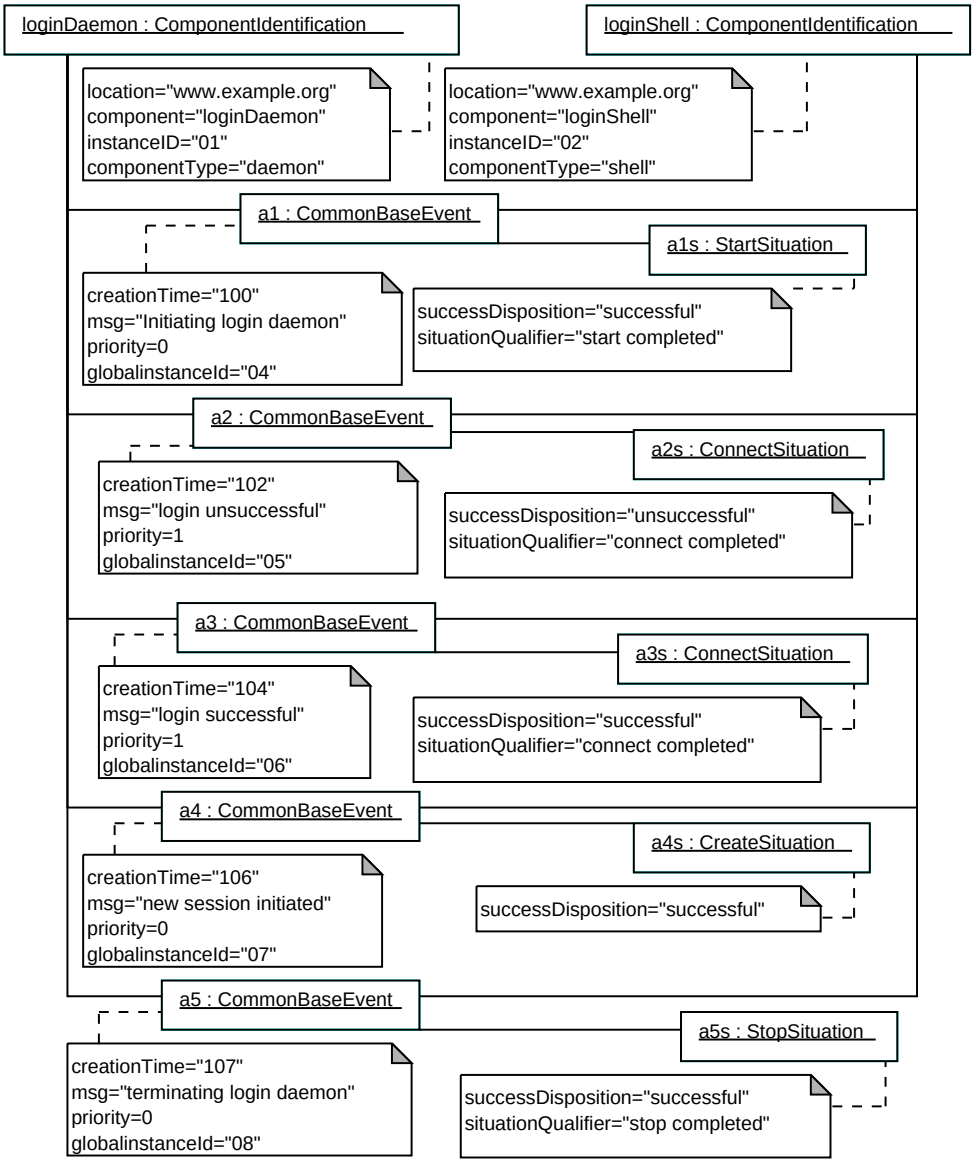
Το σενάριο το οποίο θα χρησιμοποιηθεί για την επίδειξη της εφαρμογής της επεκτεταμένης πράξης της σύνθεσης με κανόνες αντιστοίχισης είναι πως υπάρχουν δύο αρχεία καταγραφών κατά CBE, τα οποία καταγράφουν μια συνεδία όπου όπου ένας χρήστης προσπαθεί να ταυτοποιηθεί και να εισέλθει σε ένα λογισμικό σύστημα με τη χρήση ονόματος χρήστη (username) και κωδικού πρόσβασης (password).

Στο σενάριο της εφαρμογής, ο χρήστης αποτυγχάνει μια φορά πριν καταφέρει να ταυτοποιηθεί. Στην συνέχεια συνδέεται επιτυχώς και δημιουργεί μια συνεδρία με την εφαρμογή της επιλογής του.

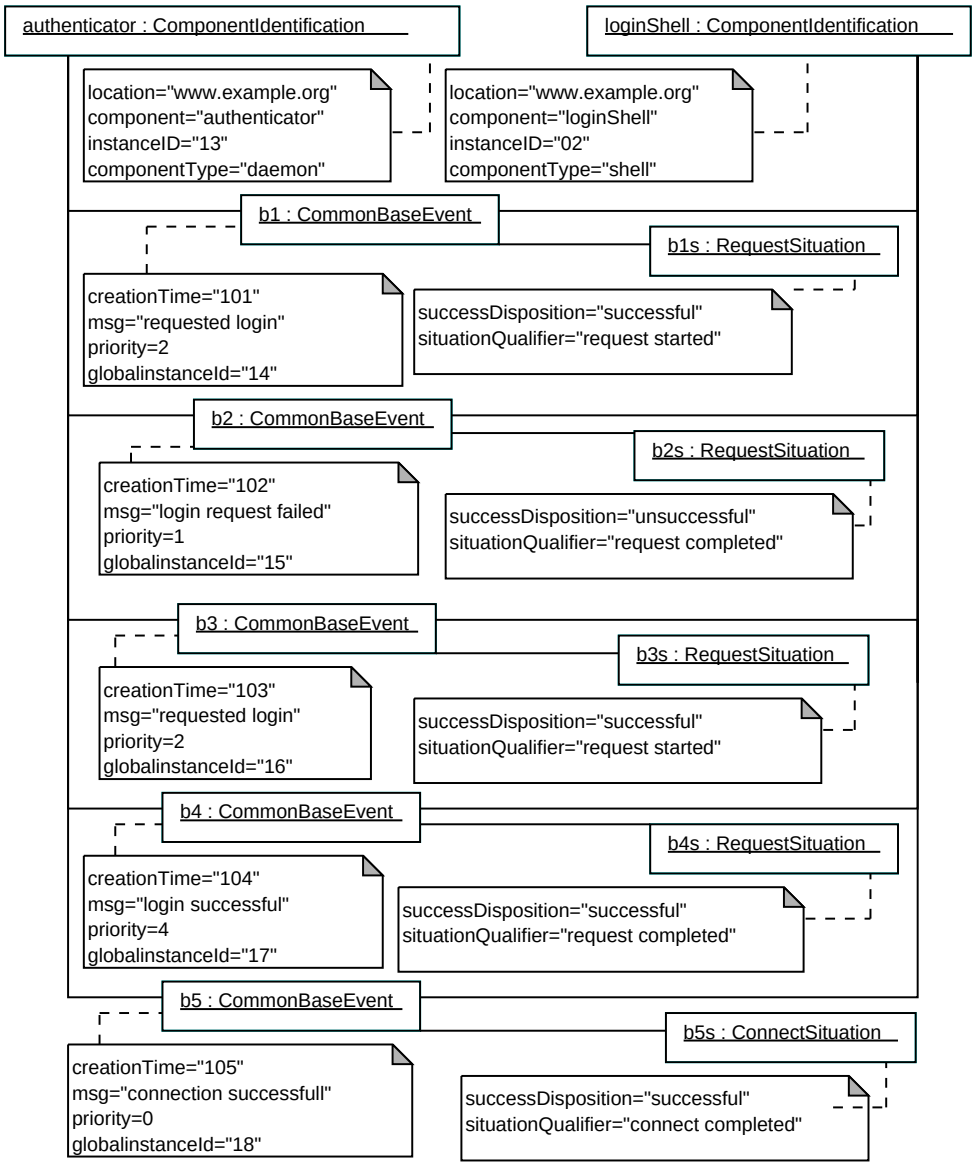
Η συναλλαγή του αυτή με το λογισμικό σύστημα γίνεται μέσα από την εφαρμογή της κονσόλας εισόδου (login shell). Καταγραφείς της συναλλαγής του χρήστη με την κονσόλα εισόδου είναι ο δαίμονας σύνδεσης (login daemon) που χρησιμεύει στην επικοινωνία του χρήστη με την κονσόλα εισόδου και ο ταυτοποιητής (authenticator) που αναλαμβάνει να επαληθεύσει την ταυτοποίηση του χρήστη ελέγχοντας την κρυπτογραφημένη βάση δεδομένων όπου φυλάσσονται τα ζευγάρια username/password.

Ο δαίμονας σύνδεσης είναι ένα νήμα το οποίο εξυπηρετεί ατομικά τον χρήστη και άρα καταστρέφεται μετά την επιτυχή δημιουργία της συνεδρίας. Ο ταυτοποιητής είναι μια εφαρμογή που εξυπηρετεί συνεχώς όλες τις αιτήσεις που προέρχονται από την κονσόλα εισόδου.

Έχουμε συνεπώς δύο μοντέλα που αντιστοιχούν στα δύο αρχεία καταγραφής, ένα για ότι καταγράφει ο login daemon και ένα για ότι καταγράφει ο authenticator. Κάθε γεγονός στα δύο αυτά μοντέλα αποτελείται από μια τριπλέτα, σύμφωνα με το CBE πρότυπο. Στα επόμενα σχήματα, φαίνονται τα δύο μοντέλα των αντοίστιχων αρχείων καταγραφής. Η κατάσταση των στιγμιοτύπων εμφανίζεται σε σχόλια.



Σχήμα 4.2: Μοντέλο του αρχείου καταγραφής που παράγεται από τον δαίμονα σύνδεσης.



Σχήμα 4.3: Μοντέλο του αρχείου καταγραφής που παράγεται από τον ταυτοποιητή.

4.2.2 Σχέση μεταξύ των μοντέλων εισόδου

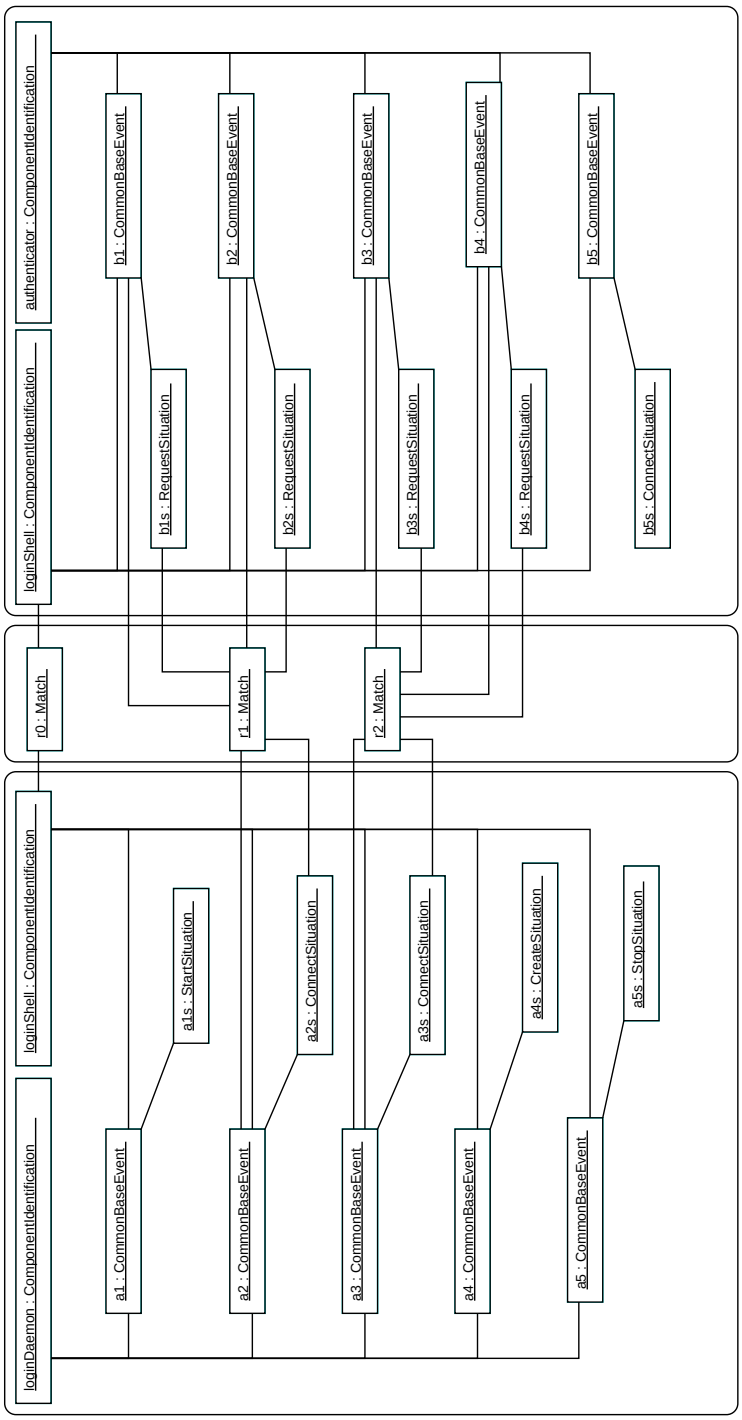
Τα δύο αρχεία καταγραφής περιγράφουν το ίδιο συμβάν με δύο διαφορετικούς τρόπους. Περιέχουν διαφορετικό βαθμό λεπτομέρειας και επικεντρώνονται σε διαφορετικά σημεία της διαδικασίας. Το γεγονός πως τα δύο αρχεία περιγράφουν το ίδιο συμβάν μπορεί να αναπαρασταθεί με τη χρήση αντιστοιχίσεων μεταξύ των στοιχείων των δύο μοντέλων που αναπαριστούν τα δύο αρχεία καταγραφής.

Θεωρούμε πως η σχέση μεταξύ των δύο μοντέλων είναι δοσμένη. Στην συγκεκριμένη περίπτωση, η σχέση αυτή υπολογίστηκε με το χέρι αλλά με κατάλληλες τεχνικές μπορεί αυτή η σχέση να προκύψει και με αυτοματοποιημένο τρόπο (ο αναγνώστης καλείται να ανατρέξει στο κεφάλαιο 3 της εργασίας για την σχετική βιβλιογραφία). Σε κάθε περίπτωση, είτε πρόκειται για χειροκίνητη αντιστοίχιση, είτε για κάποια αυτοματοματοποιημένη διαδικασία, για τις ανάγκες της εργασίας δεν υπάρχει διαφορά.

Στην περίπτωση μας, έχουμε:

- Και στα δύο μοντέλα υπάρχει το στοιχείο `loginShell`, που είναι στιγμιότυπο της CBE κλάσης `ComponentIdentification`. Μεταξύ των στοιχείων στα δύο μοντέλα υπάρχει μια ένα προς ένα αντιστοιχία.
- Τα στοιχεία `a2` και `a2s` του μοντέλου του αρχείου καταγραφής του δαίμονα σύνδεσης και τα στοιχεία `b1`, `b1s`, `b2` και `b2s` στο μοντέλο του αρχείου καταγραφής του ταυτοποιητή αναπαριστούν με δύο διαφορετικούς τρόπους την πρώτη αποτυχημένη απόπειρα του χρήστη να συνδεθεί στο σύστημα. Μεταξύ τους υπάρχει μια αντιστοιχία πολλά προς πολλά.
- Τα στοιχεία `a3` και `a3s` του μοντέλου του αρχείου καταγραφής του δαίμονα σύνδεσης και τα στοιχεία `b3`, `b3s`, `b4` και `b4s` στο μοντέλο του αρχείου καταγραφής του ταυτοποιητή αναπαριστούν με δύο διαφορετικούς τρόπους την δεύτερη, επιτυχημένη απόπειρα του χρήστη να συνδεθεί στο σύστημα. Μεταξύ τους υπάρχει μια αντιστοιχία πολλά προς πολλά.

Οι σχέσεις αυτές φαίνονται παρακάτω. Για λόγους οικονομίας χώρου τα στοιχεία των μοντέλων περιλαμβάνονται μόνο με το όνομα τους.



Σχήμα 4.4: Σχέσεις μεταξύ των δύο αρχείων καταγραφής.

4.2.3 Μετασχηματισμός των μοντέλων εισόδου

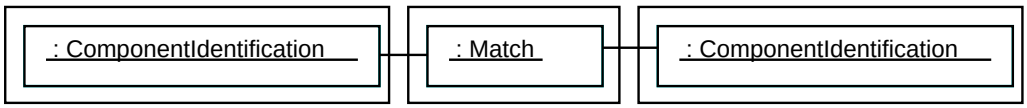
Σκοπός της σύνθεσης είναι να δημιουργηθεί μια ενιαία καταγραφή, που να εξιστορεί ολόκληρο το συμβάν. Ο τρόπος με τον οποίον μπορεί να γίνει η σύνθεση, εξαρτάται από την απόφαση πώς θα αντιστοιχιστούν και πώς θα μετασχηματιστούν αυτές οι αντιστοιχίσεις, με τη χρήση των κανόνων, όπως περιγράφηκε στο προηγούμενο κεφάλαιο.

Για παράδειγμα, αν υποθέσουμε πως η αντιστοίχιση γίνεται χειροκίνητα και ο μηχανικός επιλέξει να παρουσιάσει την λεπτομερή εξιστοριση, τότε θα προτιμούσε στο τελικό μοντέλο να περιλαμβάνεται η λεπτομερέστερη περιγραφή της διαδικασίας ταυτοποίησης και σύνδεσης που περιγράφεται στο μοντέλο που αντιπροσωπεύει το αρχείο καταγραφής του ταυτοποιητή.

Αν αντίθετα επιλέξει να παρουσιάσει πιο συνοπτικά τα γεγονότα, μπορεί να συγχωνεύσει τις λεπτομέρειες που βρίσκονται στο αρχείο καταγραφής του ταυτοποιητή σε μια συνολική καταγραφή της επιτυχημένης ή αποτυχημένης σύνδεσης του χρήστη, όπως περιγράφεται στο μοντέλο που αντιπροσωπεύει το αρχείο καταγραφής του δαίμονα.

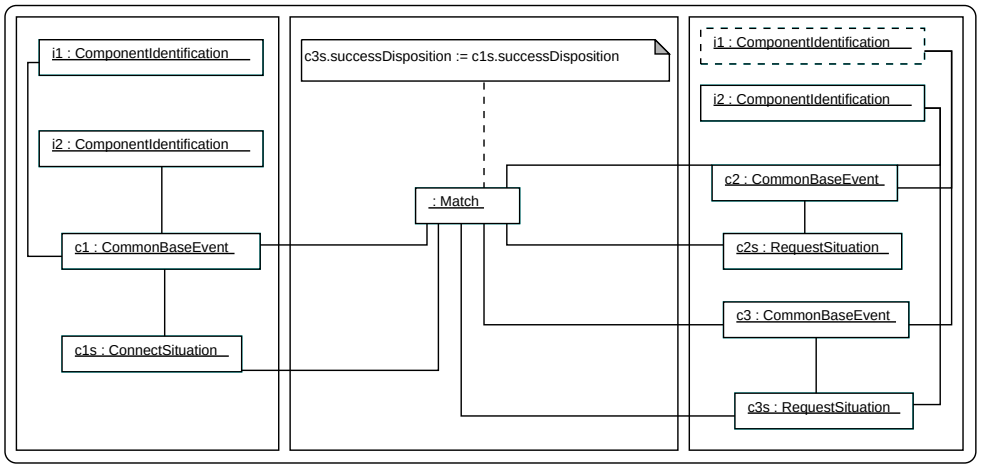
Στην προκείμενη περίπτωση, θα θεωρήσουμε πως το ζητούμενο είναι η κατά το δυνατόν λεπτομερέστερη εξιστόριση. Συνεπώς οι κανόνες που θα επιλέξουμε για την επεξεργασία των αντιστοιχίσεων μεταξύ των δύο αρχείων καταγραφών θα προτιμούν την λεπτομερέστερη περιγραφή.

Ο πρώτος κανόνας χρησιμεύει για το χειρισμό της αντιστοίχισης μεταξύ των δύο εμφανίσεων του το στοιχείου `loginShell`. Καθώς η αντιστοίχιση είναι ένα προς ένα, ο κανόνας δεν κάνει κάτι το ιδιαίτερο.



Σχήμα 4.5: Πρώτος κανόνας.

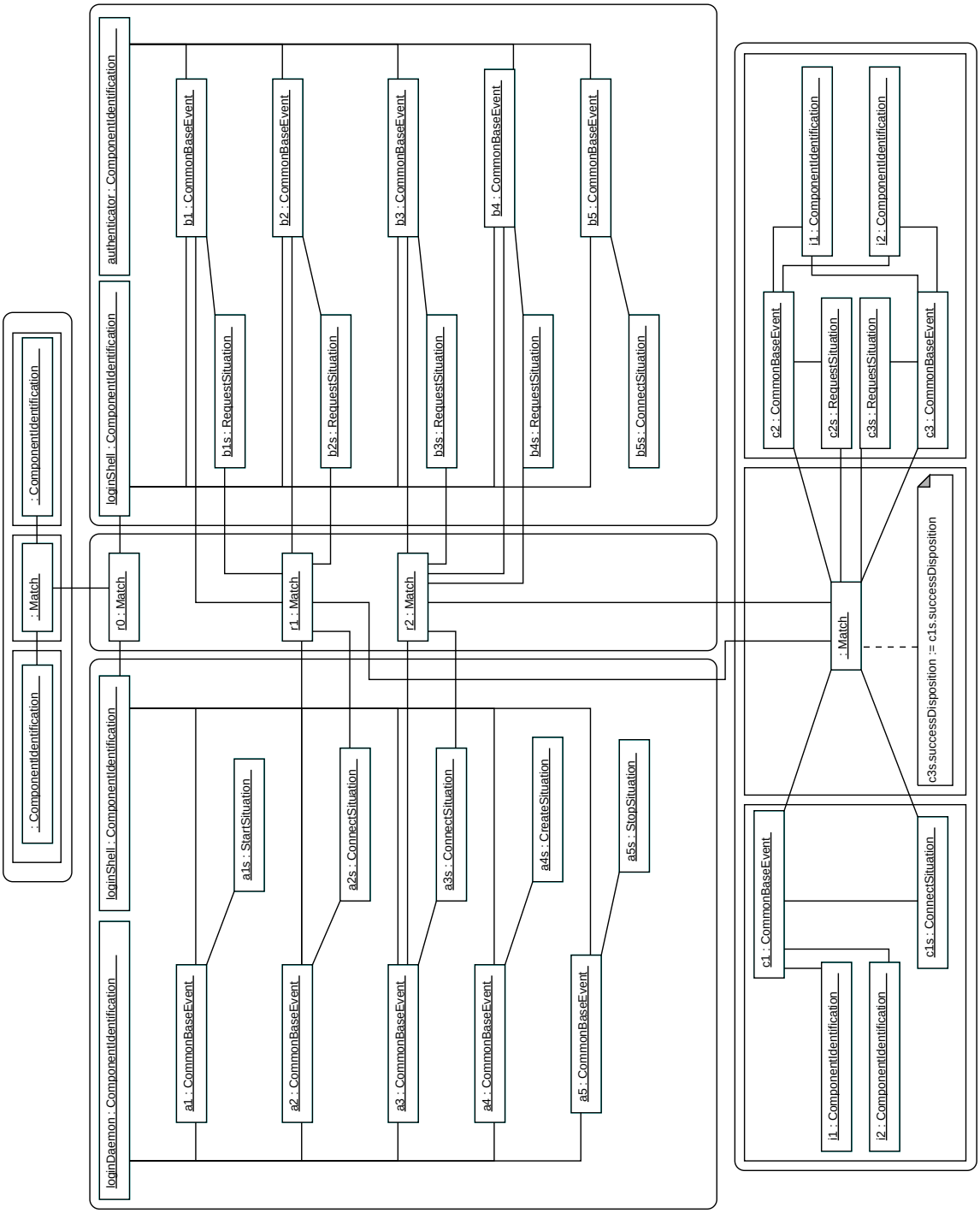
Ο δεύτερος κανόνας χρησιμεύει για τον χειρισμό των δύο αντιστοιχίσεων (της αντιστοίχισης των `a2` και `a2s` με τα `b1`, `b1s`, `b2` και `b2s`, καθώς και της αντιστοίχισης των `a3` και `a3s` με τα `b3`, `b3s`, `b4` και `b4s`) που αφορούν την αναπαράσταση της προσπάθειας σύνδεσης.



Σχήμα 4.6: Δεύτερος κανόνας.

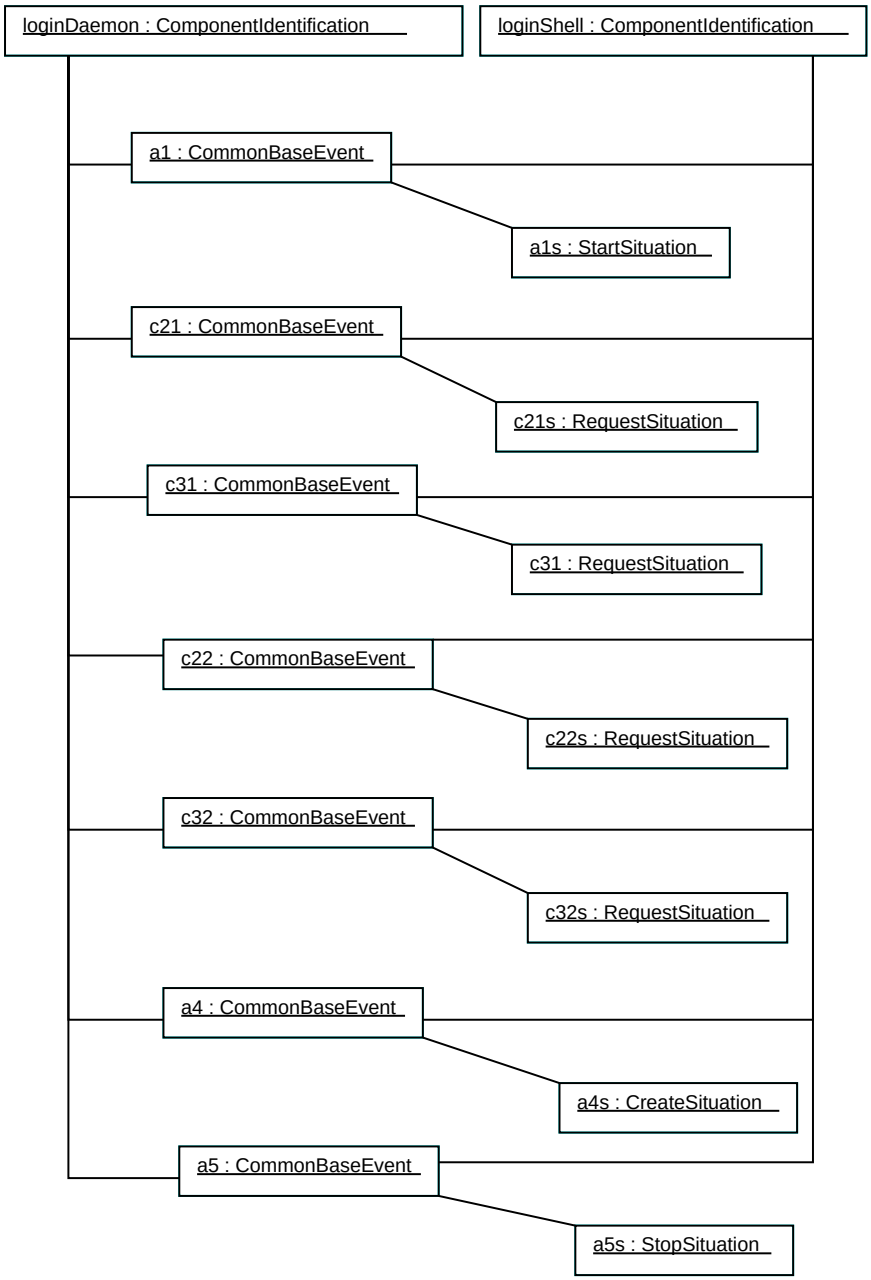
Χρησιμοποιείται ο ίδιος κανόνας και για τις δύο αντιστοιχίσεις αφού αυτές διαφέρουν μόνο κατά το αν η σύνδεση ήταν επιτυχής ή όχι. Η διαφορά αυτή περιγράφεται με τον επιπλέον περιορισμό που πρέπει να εφαρμοστεί ώστε η τιμή του πεδίου `successDisposition` της δεύτερης κατάστασης του μετασχηματισμένου μοντέλου να γίνεται ίση με την αντίστοιχη τιμή του πεδίου της κατάστασης στο αρχικό μοντέλο.

Έχοντας κατά νου αυτούς τους δύο κανόνες, έχουμε την παρακάτω αναγνώριση των αντιστοιχίσεων σαν εφαρμογές των κανόνων για τους δύο γράφους:



Σχήμα 4.7: Αναγνώριση των αντιστοιχίσεων σαν εφαρμογές των δύο κανόνων.

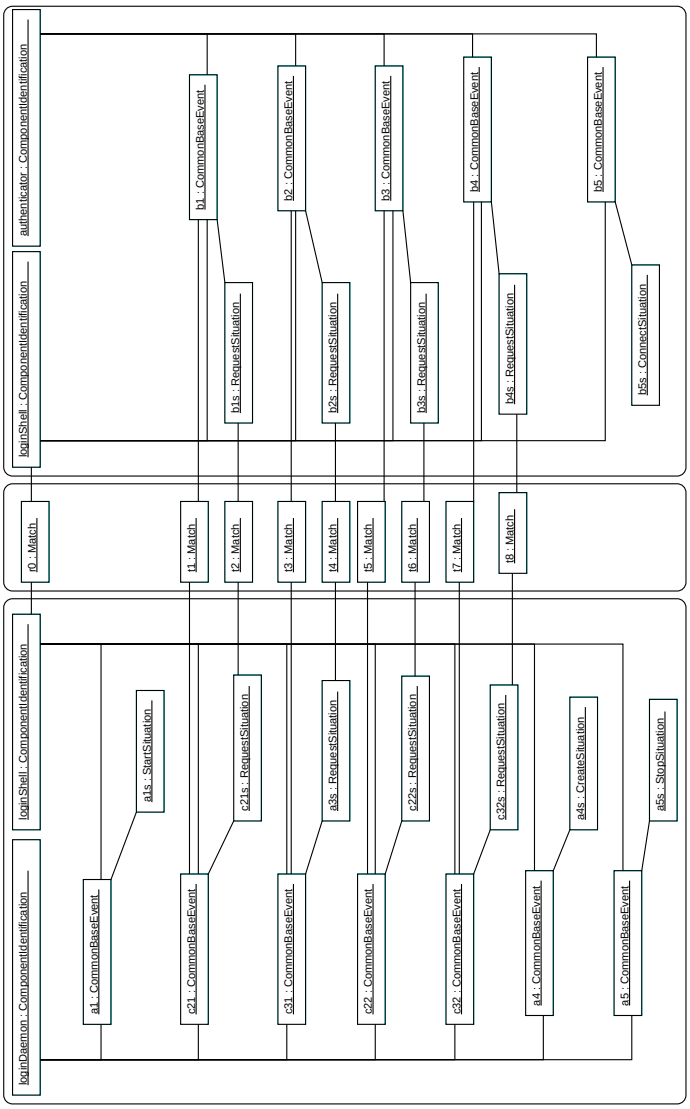
Μπορούμε να συμπεράνουμε από την μορφή των κανόνων πως η εφαρμογή τους θα μετασχηματίσει μόνο το μοντέλο που αναπαριστά το αρχείο καταγραφών του δαίμονα σύνδεσης. Συνεπώς, εφαρμόζοντας τους παίρνουμε το παρακάτω μετασχηματισμένο μοντέλο:



Σχήμα 4.8: Μετασχηματισμένο μοντέλο του αρχείου του δαίμονα σύνδεσης.

4.2.4 Σύνθεση των μετασχηματισμένων μοντέλων

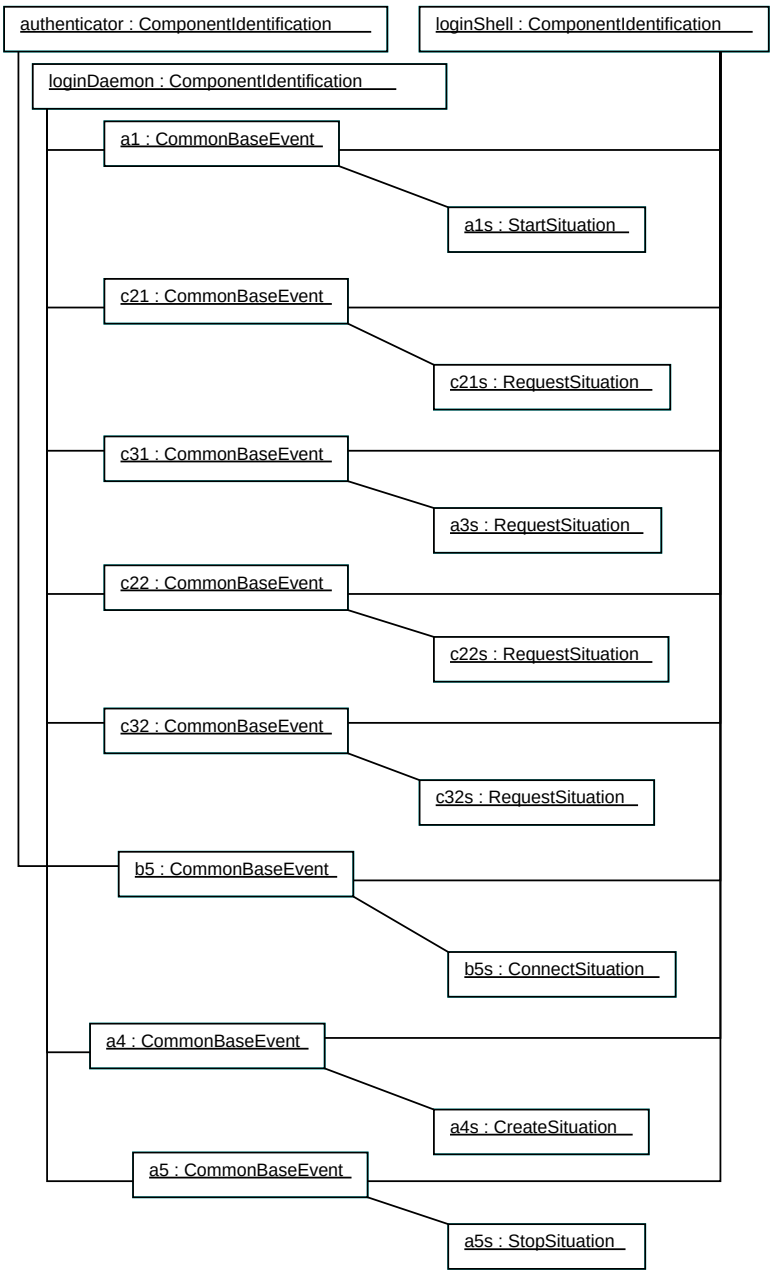
Μετά και τον μετασχηματισμό με χρήση των κανόνων, τα μετασχηματισμένα μοντέλα εισόδου μπορούν να συντεθούν με βάση τον αλγόριθμο (3). Έχουμε τα μετασχηματισμένα μοντέλα:



Σχήμα 4.9: Μετασχηματισμένα μοντέλα εισόδου για τον απλό αλγόριθμο σύνθεσης.

Με εκτέλεση του αλγορίθμου (3) παίρνουμε το τελικό, ενιαίο μοντέλο, που αν-

τιστοιχεί σε ένα ενιαίο αρχείο καταγραφής.



Σχήμα 4.10: Ενιαίο αρχείο καταγραφής.

Κεφάλαιο 5

Επίλογος

Καταλήγουμε την διπλωματική εργασία με μια ανακεφαλαιωτική ματιά στα όσα περιγράψαμε και παρουσιάσαμε μέχρι εδώ. Αρχικά θα ανακεφαλαιώσουμε και θα αναφερθούμε στα συμπεράσματα στα οποία καταλήξαμε κατά τη διάρκεια της περιγραφής της άλγεβρας μοντέλων και του γενικότερου οικοσυστήματος μέσα στο οποίο αυτή νοείται.

Η διπλωματική εργασία θα καταλήξει με αναφορά σε εκείνα τα ζητήματα που θεωρήσαμε πως χρήζουν περαιτέρω διερεύνησης, με τη μορφή προτάσεων για μελλοντική έρευνα.

5.1 Συμπεράσματα

Παρακάτω παρουσιάζουμε τα γενικά συμπεράσματα της διπλωματικής εργασίας, αντίστοιχα με τους στόχους τους οποίους θέσαμε στο πρώτο κεφάλαιο.

- Περιγράψαμε ένα οικοσύστημα για την μοντελοκεντρική προσέγγιση στην τεχνολογία λογισμικού. Αναφερθήκαμε στον κεντρικό ρόλο που παίζει το Meta Object Facility σε αυτό το οικοσύστημα και περιγράψαμε με ποιο τρόπο προσεγγίζεται το ζήτημα της παραγωγής λογισμικού, μέσα από τις διαδικασίες μετασχηματισμού μοντέλων και τη χρήση προγραμματιστικών διεπαφών σε υπάρχοντα πρότυπα και τεχνολογίες. Η περιγραφή μας αυτή συνοδεύτηκε και από συνοπτική αναφορά σε υπάρχονα εργαλεία που μπορεί να χρησιμοποιήσει ο μηχανικός. Συνολικά επιχει-

ρήσαμε μια σφαιρική παρουσίαση του περιβάλλοντος στο οποίο νοείται το πρόβλημα της διαχείρισης μοντέλων.

- Με την περιγραφή της άλγεβρας μοντέλων, δώσαμε ένα γενικότερο πλαίσιο στο οποίο μπορεί να αντιμετωπιστεί το πρόβλημα της διαχείρισης μοντέλων με έναν συστηματικό τρόπο. Επικεντρωθήκαμε στις πράξεις της σύνθεσης και της διαφοράς καθώς είδαμε πως αυτές αποτελούν κεντρικοί άξονες για μια τέτοια αλγεβρική προσέγγιση. Δεν παραλείψαμε όμως να αναφερθούμε και σε άλλους τελεστές της άλγεβρας, έστω και περιληπτικά, ενώ κάναμε και μια προσέγγιση στο ζήτημα της μελέτης των πράξεων από τη σκοπιά των αλγεβρικών ιδιοτήτων. Συνολικά δείξαμε πως είναι όντως δυνατόν να περιγραφεί ένας τέτοιος συστηματικός τρόπος μελέτης των μοντέλων και αναδείξαμε βασικά ζητήματα στα οποία θα πρέπει μια τέτοια προσέγγιση να απαντά.

- Δεν μείναμε στην περιγραφή της άλγεβρας μοντέλων. Προχωρήσαμε στην παρουσίαση των αλγορίθμων με τους οποίους είναι δυνατόν να υλοποιηθουν οι πράξεις της Σύνθεσης και της Διαφοράς. Αναφερθήκαμε εκτεταμένα στις προϋποθέσεις που πρέπει να πληρούνται ώστε να είναι δυνατή η χρήση των αλγορίθμων και περιγράψαμε τι είδους περιπτώσεις περιμένουμε να είναι σε θέση να επεξεργαστου οι αλγόριθμοι. Συνολικά δείξαμε την λειτουργική διάσταση των πράξεων της άλγεβρας μοντέλων μας και με την παρουσίαση αλγοριθμικών τρόπων υλοποίησης τους, θέσαμε τις βάσεις για να μπορούν αυτές να αποτελέσουν πρακτικές εφαρμογές σε μοντελοκεντρικά εργαλεία λογισμικού.

- Προχωρήσαμε επιπλέον στην πρακτική εφαρμογή της επεκτεταμένης πράξης της σύνθεσης σε ένα συγκεκριμένο πρόβλημα, που άπτεται ενός προβήματος που δεν συνδέεται αυστηρά με την μοντελοκεντρική προσέγγιση καθαυτή. Εκτός των άλλων, με αυτόν τον τρόπο δείξαμε πως η μοντελοκεντρική προσέγγιση μπορεί να χρησιμοποιηθεί για την επίλυση πρακτικών προβλημάτων, παρά τον εγγενώς αφαιρετικό χαρακτήρα της. Επιπλέον δώσαμε μια λύση στο πρόβλημα της διαχείρισης των καταγραφών που παράγονται από κατανεμημένες εφαρμογές, επικεντρώνοντας στο ζήτημα της σύνθεσης διαφορετικών καταγραφών σε μία ενιαία καταγραφή.

5.2 Προτάσεις για μελλοντική έρευνα

Κατά τη διάρκεια της επεξεργασίας και της συγγραφής αυτής της διπλωματικής εργασίας, συναντήσαμε διάφορα σημεία τα οποία θεωρούμε πως χρήζουν περαιτέρω διερεύνησης.

Ένα βασικό ζήτημα που συναντήσαμε επανειλλημένα στην προσπάθεια να περιγράψουμε τις πράξεις της άλγεβράς μας, ήταν το ζήτημα της αντιστοίχισης μοντέλων. Αναφερθήκαμε με αρκετή έκταση σε υπάρχουσες σχετικές τεχνικές που προσπαθούν να αντιμετωπίσουν το ζήτημα. Η εισαγωγή από μέρους μας των κανόνων πάνω σε αντιστοιχίσεις μεταξύ στοιχείων μοντέλων ανοίγει το πεδίο ακόμα περισσότερο. Θα είχε ιδιαίτερο ενδιαφέρον κατά τη γνώμη μας να μελετηθεί με ποιο τρόπο μπορούν να προκύψουν τέτοιοι κανόνες με όσο το δυνατόν περισσότερο αυτοματοποιημένο τρόπο. Συγκεκριμένα, θεωρούμε πως χρήζει περισότερης μελέτης το πρόβλημα του αν μπορούν να παράγονται κανόνες από κάποιους μέτα-κανόνες, κανόνες δηλαδή τέτοιους που να προδιαγράφονται σε επίπεδο μεταμοντέλου. Οι γραμματικές γράφων, και βέβαια οι τριπλές γραμματικές γράφων (TGG), σε συνδυασμό με την προσέγγιση των μοντέλων σαν γράφους πιστεύουμε πως μπορούν να είναι ένα πολύ χρήσιμο εργαλείο σε αυτήν την κατεύθυνση.

Επιλέον, ένα σημείο που μας κίνησε το ενδιαφέρον ήταν η δυνατότητα να οριστεί η άλγεβρα των μοντέλων με έναν πιο μαθηματικά τυπικό και αυστηρό τρόπο. Στο [6], οι πράξεις συνόλων που ορίζονται έχουν πιο αυστηρά χαρακτηριστικά (είναι για παράδειγμα αυστηρά δυαδικές και είναι κλειστές στο σύνολο των μοντέλων), όμως έχουν και πιο ισχυρές προϋποθέσεις, όπως την απαίτηση οι αντιστοιχίσεις μεταξύ των μοντέλων να είναι αυστηρά ένα προς ένα. Λαμβάνοντας υπ' όψη την επέκταση στην πράξη της σύνθεσης που παρουσιάσαμε, και η οποία επιτρέπει την ύπαρξη πολλαπλότητας στις αντιστοιχίσεις, θα είχε ενδιαφέρον να μελετηθεί με αν μπορεί η πιο αυστηρή αυτή άλγεβρα των πράξεων συνόλων μπορεί να επεκταθεί παρομοίως.

Τέλος, θεωρούμε πως το ζήτημα της μελέτης των αλγεβρικών ιδιοτήτων των πράξεων κάποιας άλγεβρας μοντέλων, επίσης μπορεί να θέσει πολύ ενδιαφέροντα ερωτήματα στον ερευνητή. Άλλωστε, οι ιδιότητες αυτές έχουν και ισχυρό τον χαρακτήρα της προδιαγραφής, οπότε και τίθενται σημαντικά ερωτήματα κατά πόσο μπορεί να χρησιμοποιηθεί αυτή η προσέγγιση των τελεστών και των αλγεβρικών ιδιοτήτων τους για την προδιαγραφή επιθυμητών χαρακτηριστικών για εργαλεία. Επιπλέον τίθεται και το ενδιαφέρον ζήτημα αν είναι δυνατόν αυτές οι προδιαγραφές να είναι πάντοτε πρακτικά υλοποιή-

σιμες τόσο για τους αλγεβρικούς τελεστές που περιγράψαμε, αλλά και για τη δημιουργία βιομηχανικά χρήσιμων εργαλείων λογισμικού.

Βιβλιογραφία

[1] J. Bezivin, M. Barbero, and F. Jouault, ``On the applicability scope of model driven engineering," *Model-Based Methodologies for Pervasive and Embedded Software, 2007. MOMPES '07. Fourth International Workshop on*, pp. 3--7, March 2007.

[2] Object Management Group, Inc., *MDA Guide Version 1.0.1*. http://www.omg.org/docs/omg/03-06-01.pdf.

[3] ``The fast guide to model driven architecture, the basics of model driven architecture (mda)," January 2006.

[4] G. Brunet, M. Chechik, S. Easterbrook, S. Nejati, N. Niu, and M. Sabetzadeh, ``A manifesto for model merging," in *GaMMa '06: Proceedings of the 2006 international workshop on Global integrated model management*, (New York, NY, USA), pp. 5--12, ACM, 2006.

[5] Z. Xing and E. Stroulia, ``Umldiff: an algorithm for object-oriented design differencing," in *ASE '05: Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, (New York, NY, USA), pp. 54--65, ACM, 2005.

[6] P. Selonen, ``Set operations for the unified modeling language," in *Proceedings of the 8th Symposium on Programming Languages and Tools (SPLST'03)* (P. Kilpeläinen and N. Päivinen, eds.), pp. 70--81, University of Kuopio, June 2003.

[7] B.-Y. Koo, W. Simmons, and E. Crawley, ``Algebra of systems: An executable framework for model synthesis and evaluation," *Systems Engineering and Modeling, 2007. ICSEM '07. International Conference on*, pp. 42--49, March 2007.

[8] Object Management Group, Inc., *OMG-Meta Object Facility, Version 1.4*, April 2002.

[9] Object Management Group, Inc., *UML Infrastructure Specification, Version 2.1.2*, November 2007.

[10] Object Management Group, Inc., *Meta Object Facility (MOF) Core Specification Version 2.0*, January 2006.

[11] Object Management Group, Inc., *MOF To IDL Mapping, Version 2.0*, January 2006.

[12] Java Community Process, *Java Metadata Interface(JMI) Specification, JSR 040, Version 1.0*, June 2002.

[13] Object Management Group, Inc., *MOF 2.0/XMI Mapping, Version 2.1.1*, December 2007.

[14] Παπαϊωάννου, *Θεωρία Γραφημάτων*. Εθνικό Μετσόβιο Πολυτεχνείο, 2004.

[15] A. Corradini, U. Montanari, and F. Rossi, ``Graph processes,'' *Fundam. Inf.*, vol. 26, no. 3-4, pp. 241--265, 1996.

[16] T. Mens, K. Czarnecki, and P. V. Gorp, ``04101 discussion -- a taxonomy of model transformations,'' in *Language Engineering for Model-Driven Software Development* (J. Bezivin and R. Heckel, eds.), no. 04101 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005. <http://drops.dagstuhl.de/opus/volltexte/2005/11> [date of citation: 2005-01-01].

[17] J. Bézivin, ``From object composition to model transformation with the mda,'' in *TOOLS '01: Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS39)*, (Washington, DC, USA), p. 350, IEEE Computer Society, 2001.

[18] Object Management Group, Inc., *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification*, July 2007.

[19] J. Kovse and T. Härder, ``Generic xmi-based uml model transformations,'' in *OOIS '02: Proceedings of the 8th International Conference on Object-Oriented. Information Systems*, (London, UK), pp. 192--198, Springer-Verlag, 2002.

[20] ``Atl starter's guide, version 0.1,'' December 2005. http://www.eclipse.org/m2m/atl/doc/ATL_Starter_Guide.pdf.

[21] E. Kindler and R. Wagner, ``Triple graph grammars: Concepts, extensions, implementations, and application scenarios,'' Tech. Rep. tr-ri-07-284, Software Engineering Group, Department of Computer Science, University of Paderborn, June 2007.

[22] A. Königs, ``Model Transformation with Triple Graph Grammars,'' in *Model Transformations in Practice Satellite Workshop of MODELS 2005, Montego Bay, Jamaica*, 2005.

[23] F. Hermann, H. Ehrig, and G. Taentzer, ``A typed attributed graph grammar with inheritance for the abstract syntax of uml class and sequence diagrams,'' *Electron. Notes Theor. Comput. Sci.*, vol. 211, pp. 261--269, 2008.

[24] F. Hermann, ``A typed attributed graph grammar for syntax-directed editing of uml sequence diagrams,'' Master's thesis, Faculty for Electrical Engineering and Computer Science Technical University Berlin, October 2005.

[25] J. J. Vereijken, ``Graph grammars and operations on graphs,'' Master's thesis, Leiden University, May 1993.

[26] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. J. Grose, *Eclipse Modeling Framework*. Addison-Wesley Professional, August 2003.

[27] Obeo Inc., *Acceleo User Guide*. http://www.acceleo.org/doc/obeo/en/acceleo-2.2-user-guide.pdf.

[28] M. Matula, ``Netbeans metadata repository,'' March 2003. http://mdr.netbeans.org/MDR-whitepaper.pdf.

[29] P. Selonen, K. Koskimies, and M. Sakkinen, ``How to make apples from oranges in uml,'' *hicss*, vol. 03, p. 3054, 2001.

[30] D. Richards, ``Merging individual conceptual models of requirements,'' *Requirements Engineering*, vol. 8, no. 4, pp. 195--205, 2003.

[31] D. Ratiu, M. Feilkas, and J. Jürjens, ``Extracting domain ontologies from domain specific apis,'' in *Proceedings of the 12th European Conference on Software Maintenance and Reengineering (CSMR'08)* (K. Kontogiannis, C. Tjortjis, and A. Winter, eds.), (Washington, DC, USA), pp. 203--212, IEEE Computer Society, April 2008.

[32] I. Ivkovic and K. Kontogiannis, ``Towards automatic establishment of model dependencies using formal concept analysis.,'' *International Journal of*

Software Engineering and Knowledge Engineering, vol. 16, no. 4, pp. 499--522, 2006.

[33] M. Sabetzadeh and S. Easterbrook, ``An algebraic framework for merging incomplete and inconsistent views,'' in *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering*, (Washington, DC, USA), pp. 306--318, IEEE Computer Society, 2005. urlhttp://dx.doi.org/10.1109/RE.2005.8.

[34] T. Mens and R. V. D. Straeten, ``Incremental resolution of model inconsistencies,'' in *WADT*, pp. 111--126, 2006.

[35] N. Aizenbud-Reshef, B. T. Nolan, J. Rubin, and Y. Shaham-Gafni, ``Model traceability,'' *IBM Systems Journal*, vol. 45, no. 3, 2006.

[36] M. Girschick, ``Difference detection and visualization in uml class diagrams,'' tech. rep., Technical University of Darmstadt, 2006. TUD-CS-2006-5.

[37] H. Giese and R. Wagner, ``Incremental model synchronization with triple graph grammars,'' in *Proc. of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS), Genova, Italy* (O. Nierstrasz, J. Whittle, D. Harel, and G. Reggio, eds.), vol. 4199 of *Lecture Notes in Computer Science (LNCS)*, pp. 543--557, Springer Verlag, October 2006.

[38] IBM Corporation, *Canonical Situation Data Format: The Common Base Event V1.0.1*, April 2003.

[39] D. Bridgewater, ``Standardize messages with the common base event model,'' February 2004. http://www-128.ibm.com/developerworks/library/ac-cbe1/.