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

Δημιουργία περιγραφών Simple Sequencing σε υψηλό επίπεδο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΡΗΣΤΟΣ ΓΙΤΣΗΣ

Επιβλέπων : Νικόλαος Σ. Παπασπύρου
Επίκ. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

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

Δημιουργία περιγραφών Simple Sequencing σε υψηλό επίπεδο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΡΗΣΤΟΣ ΓΙΤΣΗΣ

Επιβλέπων : Νικόλαος Σ. Παπασπύρου
Επίκ. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 22 Ιουλίου 2009.

Νικόλαος Σ. Παπασπύρου	Κώστας Κοντογιάννης	Ανδρέας Παπασαλούρος
Επίκ. Καθηγητής Ε.Μ.Π.	Αναπλ. Καθηγητής Ε.Μ.Π.	Λέκτορας Πανεπιστήμιο Αιγαίου

Αθήνα, Ιούλιος 2009

Χρήστος Γίτσης

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

Περίληψη

Το e-learning είναι ένα είδος εκπαίδευσης υποστηριζόμενης από τις ψηφιακές τεχνολογίες. Κατά τη διαδικασία του e-learning, οι εκπαιδευόμενοι έρχονται σε επαφή με κάποιο εκπαιδευτικό περιεχόμενο, το οποίο είναι συνήθως οργανωμένο με βάση κάποιες προδιαγραφές.

Η προδιαγραφή Simple Sequencing είναι μια προδιαγραφή για την περιγραφή εκπαιδευτικών δραστηριοτήτων οι οποίες παρουσιάζονται με δυναμικό τρόπο στους εκπαιδευόμενους χρήστες, ανάλογα με τις επιδόσεις τους σε συγκεκριμένους εκπαιδευτικούς στόχους κατά τη διάρκεια της πλοήγησής τους μέσα σε κάποιο μαθησιακό υλικό.

Η δημιουργία τέτοιων περιγραφών είναι μια δύσκολη εργασία, καθώς απαιτείται σημαντική επένδυση σε κόπο και σε χρόνο για να μάθει κάποιος να χρησιμοποιεί την προδιαγραφή, ειδικά αν δεν έχει υπόβαθρο στην επιστήμη των υπολογιστών. Η παρούσα εργασία περιγράφει μια προσέγγιση για δημιουργία περιγραφών Simple Sequencing με αποκρύψη των τεχνικών λεπτομερειών της προδιαγραφής. Αυτή η προσέγγιση βασίζεται σε μια αντιστοίχιση μεταξύ των περιγραφών Simple Sequencing και των διαγραμμάτων καταστάσεων.

Με βάση αυτήν την προσέγγιση αναπτύχθηκε ένα εργαλείο συγγραφής περιγραφών Simple Sequencing, το οποίο επίσης παρουσιάζεται. Το εργαλείο αυτό ως εφαρμογή λογισμικού χτίστηκε πάνω στην πλατφόρμα Eclipse Rich Client Platform. Πρόκειται για έναν γραφικό επεξεργαστή για τη δημιουργία του οποίου χρησιμοποιήθηκε το πλαίσιο Graphical Editing Framework.

Λέξεις κλειδιά

ηλεκτρονική εκπαίδευση, εκπαιδευτικές τεχνολογίες, εργαλείο συγγραφής, γραφικός επεξεργαστής

Abstract

Electronic learning (e-learning) is a kind of education supported by the use of digital technologies. During the process of e-learning, learners are presented with some educational content, which is usually organized according to some specifications.

Simple Sequencing is a specification for describing educational activities dynamically presented to learners based upon the achievement of specific goals during their navigation in educational content.

Authoring such descriptions is a difficult task. This paper describes an approach for authoring Simple Sequencing descriptions by hiding underlying details. This approach is based on a mapping between Simple Sequencing descriptions and a well-known formalism for describing hypermedia, that is, statechart diagrams.

Based on this mapping, an authoring tool has been developed, which is also presented. The tool is a graphical editor built on top of the Eclipse Rich Client Platform and the Graphical Editing Framework.

Key words

e-learning, learning technologies, authoring, content packaging, simple sequencing, graphical editor, eclipse

Ευχαριστίες

Ευχαριστώ τους γονείς μου, που η στήριξή τους ήταν απαραίτητη για να μπορέσω να ολοκληρώσω τη φοίτησή μου στο Εθνικό Μετσόβιο Πολυτεχνείο.
Ευχαριστώ τον καθηγητή μου κ. Παπασαλούρο, με τον οποίο συνεργάστηκα πολύ στενά για την εκπόνηση αυτής της εργασίας και τον κ. Παπασπύρου για την πρόθυμη βοήθειά του όποτε τη χρειάστηκα.

Χρήστος Γίτσης

Αθήνα, Ιούλιος 2009.

Η εργασία αυτή είναι επίσης διαθέσιμη ως Τεχνική Αναφορά CSD-SW-TR-6-09, Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών, Εργαστήριο Τεχνολογίας Λογισμικού, Ιούλιος 2009

URL: `http://www.softlab.ntua.gr/techrep/`
FTP: `ftp://ftp.softlab.ntua.gr/pub/techrep/`

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Το *electronic learning* (ή αλλιώς *e-learning*) είναι είδος εκπαίδευσης–εκμάθησης υποστηριζόμενης από την τεχνολογία, στην οποία το μέσο της διδασκαλίας είναι ο ηλεκτρονικός υπολογιστής και ιδιαίτερα οι ψηφιακές τεχνολογίες.
Συχνά στο e-learning δεν υπάρχει καθόλου αλληλεπίδραση του εκπαιδευόμενου πρόσωπο με πρόσωπο με τον διδάσκοντα. Η εξάπλωση και η διάδοση αυτού του τρόπου εκμάθησης προχωρά τα τελευταία χρόνια με γρήγορους ρυθμούς, καθώς χρησιμοποιείται από πολλά πανεπιστήμια και σχολεία ανά τον κόσμο για να παραδίδουν μαθήματα στους φοιτητές και στους μαθητές τους και από πολλές εταιρείες για να εκπαιδεύουν τους εργαζομένους τους. Η ανάπτυξη του διαδικτύου και των τεχνολογιών πολυμέσων είναι οι βασικοί παράγοντες που έκαναν δυνατή την αύξηση της χρήσης του e-learning.
Ο όρος e-learning είναι αρκετά γενικός και καλύπτει ένα ευρύ φάσμα μαθησιακού υλικού, το οποίο μπορεί να διανέμεται στους εκπαιδευόμενους είτε σε έναν οπτικό δίσκο CD-ROM ή DVD, είτε μέσω κάποιου τοπικού δικτύου είτε μέσα από το διαδίκτυο.

Προδιαγραφές

Το εκπαιδευτικό υλικό χρειάζεται να είναι οργανωμένο σε κάποια ηλεκτρονική μορφή για να εξασφαλίζεται η αποτελεσματική και αποδοτική διανομή, διαχείριση και παρουσίασή του.
Αυτόν το σκοπό εξυπηρετούν οι προδιαγραφές για τη μαθησιακή τεχνολογία, η δημιουργία των οποίων έχει γίνει κυρίως από δύο οργανισμούς, τον *Advanced Distributed Learning* και το *IMS Global Learning Consortium* (http://www.adlnet.gov και http://www.imsglobal.org αντίστοιχα).
Από το σύνολο των προδιαγραφών που υπάρχουν, εκείνη που απασχολεί περισσότερο αυτήν την εργασία είναι η προδιαγραφή *Simple Sequencing* του IMS Global Learning Consortium. Πρόκειται για μια προδιαγραφή που ορίζει τον τρόπο και τη σειρά με την οποία διατάσσεται και παρουσιάζεται στους χρήστες το μαθησιακό υλικό. Λίγο πιο συγκεκριμένα, το Simple Sequencing επιτρέπει στο δημιουργό του μαθησιακού υλικού να ορίσει κανόνες, σύμφωνα με τους οποίους το λογισμικό που παρουσιάζει το υλικό στον εκπαιδευόμενο να αποφασίζει, για παράδειγμα, αν πρέπει να παραλείψει ή να επαναλάβει κάποιο τμήμα της ύλης, και αυτό να γίνεται δυναμικά την ώρα της διδασκαλίας, ανάλογα με το ιστορικό των τμημάτων της ύλης που έχει διδαχθεί ο χρήστης και ανάλογα με τις επιδόσεις του σε ορισμένα αντικείμενα.
Η προδιαγραφή Simple Sequencing από μόνη της δεν είναι αρκετή για να ορίσει τον τρόπο οργάνωσης του μαθησιακού υλικού. Γι' αυτό συνήθως χρησιμοποιείται σε συνδυασμό με μια άλλη προδιαγραφή του ίδιου οργανισμού (IMS Global), η οποία ονομάζεται *Content Packaging*.

Εργαλεία συγγραφής μαθησιακού υλικού

Η δημιουργία του ηλεκτρονικού μαθησιακού υλικού το οποίο πρόκειται να παρουσιαστεί και να διδαχθεί στους εκπαιδευόμενους με χρήση της ψηφιακής τεχνολογίας είναι μια περίπλοκη διαδικασία που αποτελείται από πολλά στάδια. Ο ρόλος των *εργαλείων συγγραφής* (authoring tools) του μαθησιακού υλικού είναι ακριβώς να διευκολύνουν το συγγραφέα του μαθησιακού υλικού να ολοκληρώσει με επιτυχία το έργο του, δηλαδή να κατασκευάσει και να οργανώσει το υλικό με τον καταλληλότερο

τρόπο και σύμφωνα με τις προδιαγραφές που έχει επιλέξει να ακολουθήσει. Τα εργαλεία είναι επιθυμητό να το επιτυγχάνουν αυτό χωρίς να απαιτούν από το συγγραφέα μεγάλη προσπάθεια ή κόπο, επιτρέποντάς του να εστιάσει περισσότερο στην υψηλή ποιότητα του περιεχομένου που δημιουργεί, παρά στις τεχνικές λεπτομέρειες των προδιαγραφών.

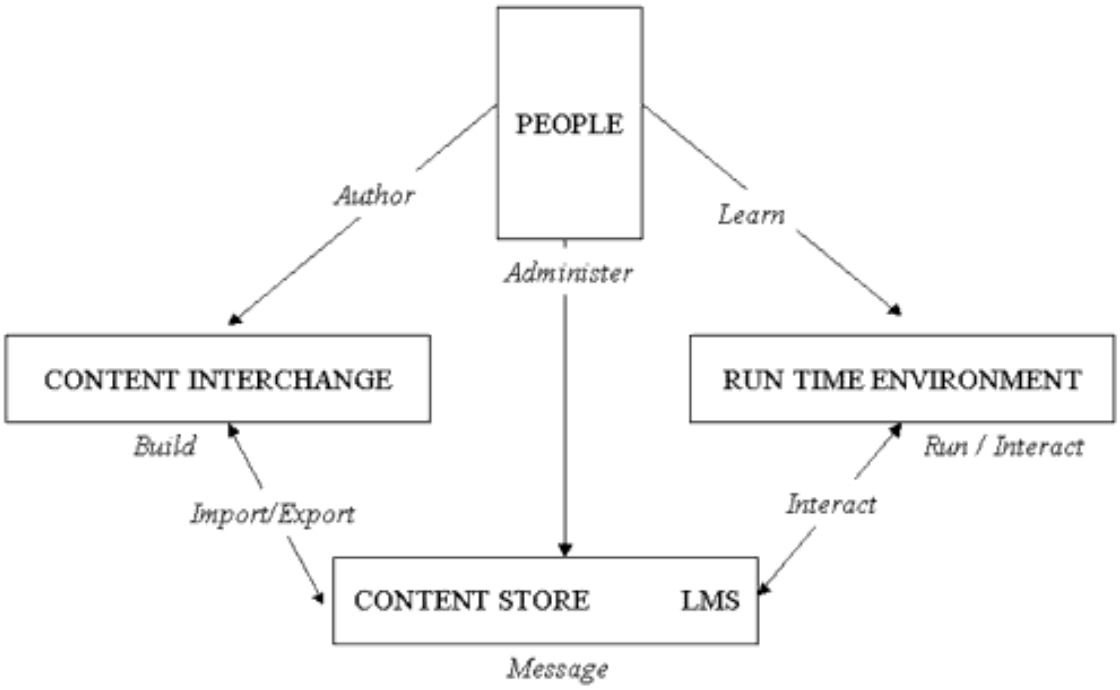
Αντικείμενο της εργασίας και δομή του βιβλίου

Αντικείμενο αυτής της εργασίας είναι η ανάπτυξη ενός εργαλείου συγγραφής μαθησιακού υλικού, το οποίο κατασκευάζει μαθησιακό υλικό οργανωμένο σύμφωνα με τις προδιαγραφές Content Packaging και Simple Sequencing του IMS Global Learning Consortium.
Στο κεφάλαιο 2 γίνεται μια αναλυτική περιγραφή των προδιαγραφών Content Packaging και Simple Sequencing. Επίσης, παρουσιάζεται μια εναλλακτική οπτική αναπαράσταση (με διαγράμματα καταστάσεων) κάποιου μαθησιακού υλικού το οποίο είναι οργανωμένο σύμφωνα με την προδιαγραφή Simple Sequencing. Αυτή η οπτική αναπαράσταση και ο τρόπος μετασχηματισμού της σε περιγραφή Simple Sequencing είναι η βάση στην οποία στηρίζεται η δημιουργία του εργαλείου, αφού ουσιαστικά επιτρέπει στο συγγραφέα να οργανώνει το μαθησιακό υλικό που δημιουργεί σύμφωνα με την προδιαγραφή, χωρίς να χρειάζεται να μάθει πολλές λεπτομέρειες γι' αυτήν.
Η διαδικασία ανάπτυξης της εφαρμογής λογισμικού – εργαλείου συγγραφής περιγράφεται **στο κεφάλαιο 3**. Το εργαλείο δεν αναπτύχθηκε από την αρχή ως ανεξάρτητη εφαρμογή, αλλά βασίζεται στην πλατφόρμα *eclipse* και είναι μια επέκτασή της. Το eclipse είναι μια πλατφόρμα γενικής χρήσης, η οποία έχει σκοπό να χρησιμεύει ως βάση για τη δημιουργία διάφορων εργαλείων. Στην πρώτη ενότητα του κεφαλαίου περιγράφεται η πλατφόρμα και ο τρόπος με τον οποίο μπορεί να επεκταθεί ώστε να προστεθούν σε αυτήν νέες δυνατότητες. Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το πλαίσιο Graphical Editing Framework, το οποίο περιγράφεται στη δεύτερη ενότητα του κεφαλαίου. Στη συνέχεια του κεφαλαίου περιγράφονται οι απαιτήσεις, η οργάνωση του κώδικα και ο έλεγχος της εφαρμογής.
Τέλος, **στο κεφάλαιο 4** γίνεται αξιολόγηση και προτάσεις για μελλοντική δουλειά.

Κεφάλαιο 2

Περιγραφή των προδιαγραφών

Οι ανάγκες όλων των ανθρώπων που συμμετέχουν στη διαδικασία του e-learning (συγγραφέων του μαθησιακού υλικού, διαχειριστών συστημάτων, κατασκευαστών λογισμικού, εκπαιδευομένων) εξυπηρετούνται καλύτερα όταν το μαθησιακό υλικό είναι οργανωμένο σε πακέτα σύμφωνα με ένα γνωστό τρόπο, με μια συγκεκριμένη δομή αρχείων και με αρκετή τεκμηρίωση που να περιγράφει ικανοποιητικά αυτόν τον τρόπο οργάνωσης.
Μια τέτοια πρόταση τρόπου οργάνωσης του υλικού αποτελεί το πλαίσιο IMS Content Framework του IMS Global Learning Consortium. Ο σκοπός του πλαισίου αυτού είναι να κάνει δυνατή την *ενθυλάκωση*, με έναν συνοπτικό και ευανάγνωστο τρόπο, όλου του απαιτούμενου εκπαιδευτικού περιεχομένου, μαζί με τις όλες τις δευτερεύουσες πληροφορίες και με την περιγραφή της δομής του, πράγματα που απαιτούνται για να εξασφαλίζεται μια πλήρης μαθησιακή εμπειρία.



Σχήμα 2.1: Στόχοι του IMS Content framework

Πιο συγκεκριμένα, όπως φαίνεται και στο σχήμα 2.1 [1], το IMS Content framework προσπαθεί να επιτρέπει:

- Στους συγγραφείς να *δημιουργούν* εκπαιδευτικό υλικό σε ηλεκτρονική μορφή.
- Στους διαχειριστές να *διαχειρίζονται* και να διανέμουν το υλικό.

[1] Σχήματα αυτής της ενότητας: Copyright ©2006 IMS Global Learning Consortium (έγγραφα [2, 3])

- Στους εκπαιδευόμενους να *αλληλεπιδρούν* και να μαθαίνουν από το υλικό.

Λόγω του μεγέθους και της πολυπλοκότητας του IMS Content Framework οι δημιουργοί του το διαίρεσαν σε τρία κύρια μέρη (σχήμα 2.2), που περιγράφονται ανεξάρτητα και είναι τα ακόλουθα:

Content Packaging είναι η ενότητα που ασχολείται με τα θέματα της συνάθροισης των πόρων του μαθησιακού υλικού (content resource aggregation, δηλαδή του «πακεταρίσματος» του υλικού), της οργάνωσης της μαθησιακής εμπειρίας και των μετα-δεδομένων.

Data Model είναι το τμήμα του πλαισίου ασχολείται με την εισαγωγή, αποθήκευση και διαχείριση του περιεχομένου.

Runtime Environment είναι το τμήμα όπου οι εκπαιδευόμενοι αλληλεπιδρούν με το μαθησιακό υλικό που τους παρουσιάζεται.

2.1 IMS Content Packaging

Το IMS Content Packaging (CP) Specification περιγράφει δομές δεδομένων που χρησιμοποιούνται για την επίτευξη της διαλειτουργησιμότητας ψηφιακού μαθησιακού υλικού με εργαλεία συγγραφής μαθησιακού υλικού, learning management systems (LMS), και περιβάλλοντα χρόνου εκτέλεσης. Σκοπός του IMS CP είναι ο ορισμός ενός προτυποποιημένου συνόλου από δομές που μπορούν να χρησιμοποιηθούν για την ανταλλαγή του μαθησιακού υλικού μεταξύ διαφορετικών συστημάτων.

2.1.1 Τα πακέτα (Content Packages)

Σύμφωνα με την προδιαγραφή, το μαθησιακό υλικό οργανώνεται σε μονάδες που ονομάζονται *πακέτα* (packages). Ένα πακέτο έχει τη δομή που φαίνεται στο σχήμα 2.3 και αποτελείται από δύο κύρια μέρη: Ένα ειδικό αρχείο XML (μια συνοπτική περιγραφή της γλώσσας XML γίνεται στο παράρτημα A□) που περιγράφει την οργάνωση του περιεχομένου και των πόρων μέσα σε ένα πακέτο, και τους ίδιους τους πόρους που περιγράφονται από το XML αρχείο. Το ειδικό αρχείο XML ονομάζεται αρχείο IMS Manifest ("προκήρυξη"). Όταν ένα πακέτο ενσωματωθεί μέσα σε ένα μοναδικό αρχείο για να είναι εύκολη η μεταφορά του, τότε ονομάζεται Package Interchange File. Ακολουθεί μια πιο αναλυτική περιγραφή των τμημάτων του πακέτου που εμφανίζονται στο σχήμα 2.3.

Package Interchange File Ένα μοναδικό αρχείο (π.χ. .zip ή .jar ή .cab) το οποίο συμπεριλαμβάνει ένα αρχείο manifest με όνομα "imsmanifest.xml" καθώς και όλα τα άλλα αρχεία που αναφέρονται μέσα στο αρχείο manifest. Ένα αρχείο Package Interchange είναι ένας συνοπτικός τρόπος διανομής του περιεχομένου, που επιτρέπει να διατηρείται η δομή των πληροφοριών. Ο προτεινόμενος τύπος για τα αρχεία Package Interchange είναι ο PKZip v2.04g (.zip).

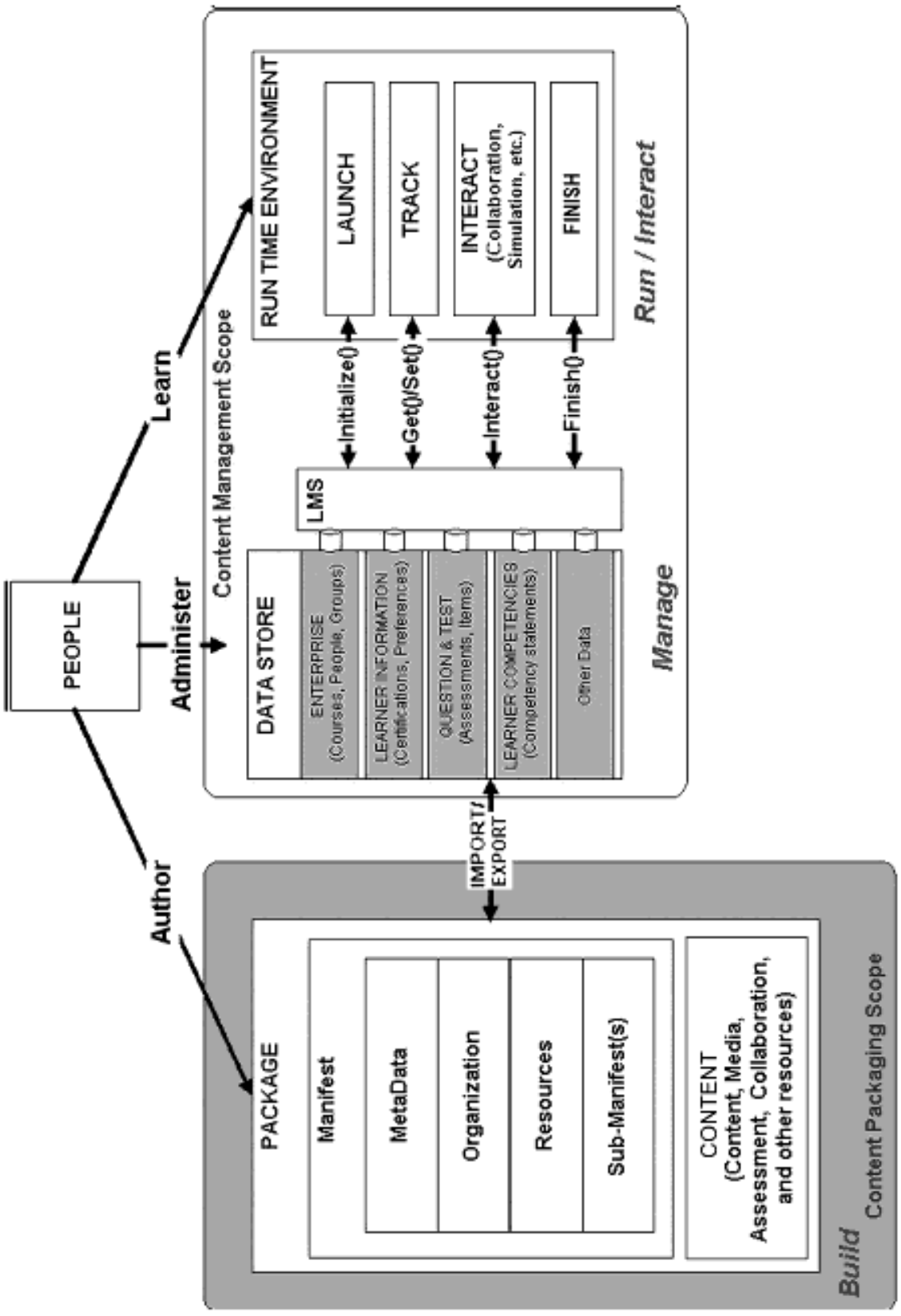
Package Ένας λογικός κατάλογος, που περιλαμβάνει ένα αρχείο XML με συγκεκριμένο όνομα, όποια άλλα αρχεία XML αναφέρονται μέσα του, π.χ. αρχεία DTD ή XSD (βλ. παράρτημα A□), και επίσης τα ίδια τα αρχεία που περιέχουν τους πόρους. Οι πόροι μπορούν να είναι οργανωμένοι σε υποκαταλόγους.

Manifest (κορυφαίου επιπέδου) Ένα υποχρεωτικό στοιχείο XML που περιγράφει το ίδιο το πακέτο. Μπορεί επίσης να περιλαμβάνει προαιρετικά υπό-Manifests. Κάθε στιγμιότυπο ενός Manifest περιλαμβάνει τα ακόλουθα τμήματα:

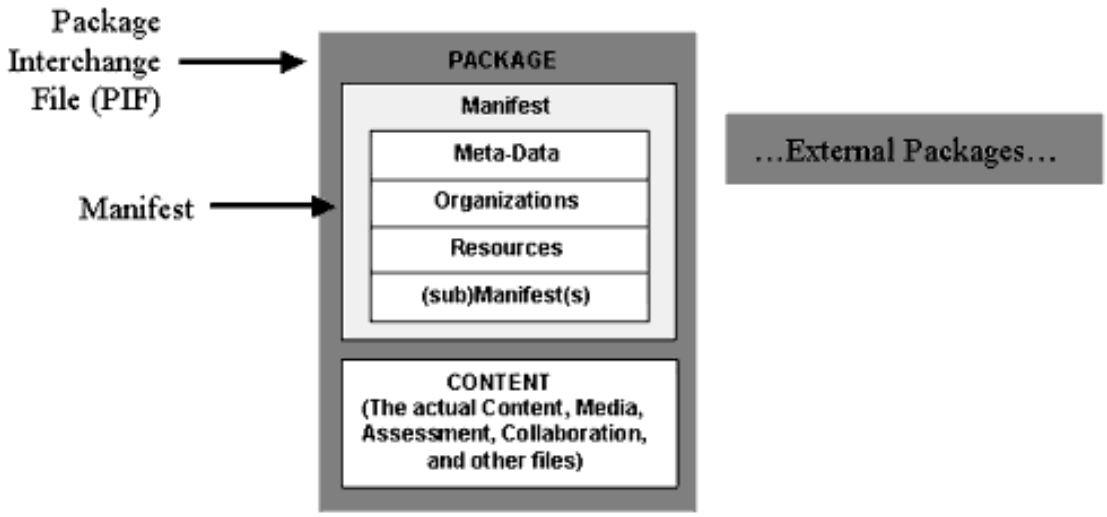
τμήμα Meta-Data Ένα στοιχείο XML που περιέχει δεδομένα για το ίδιο το manifest.

τμήμα Organizations Ένα στοιχείο XML που περιγράφει μηδέν, έναν ή περισσότερους τρόπους οργάνωσης του περιεχομένου του manifest.

τμήμα Resources Ένα στοιχείο XML που περιέχει αναφορές σε όλους τους πόρους και στοιχεία πολυμέσων που χρειάζεται το manifest, και αναφορές σε εξωτερικά αρχεία.



Σχήμα 2.2: το IMS Content framework



Σχήμα 2.3: Ένα Content Package

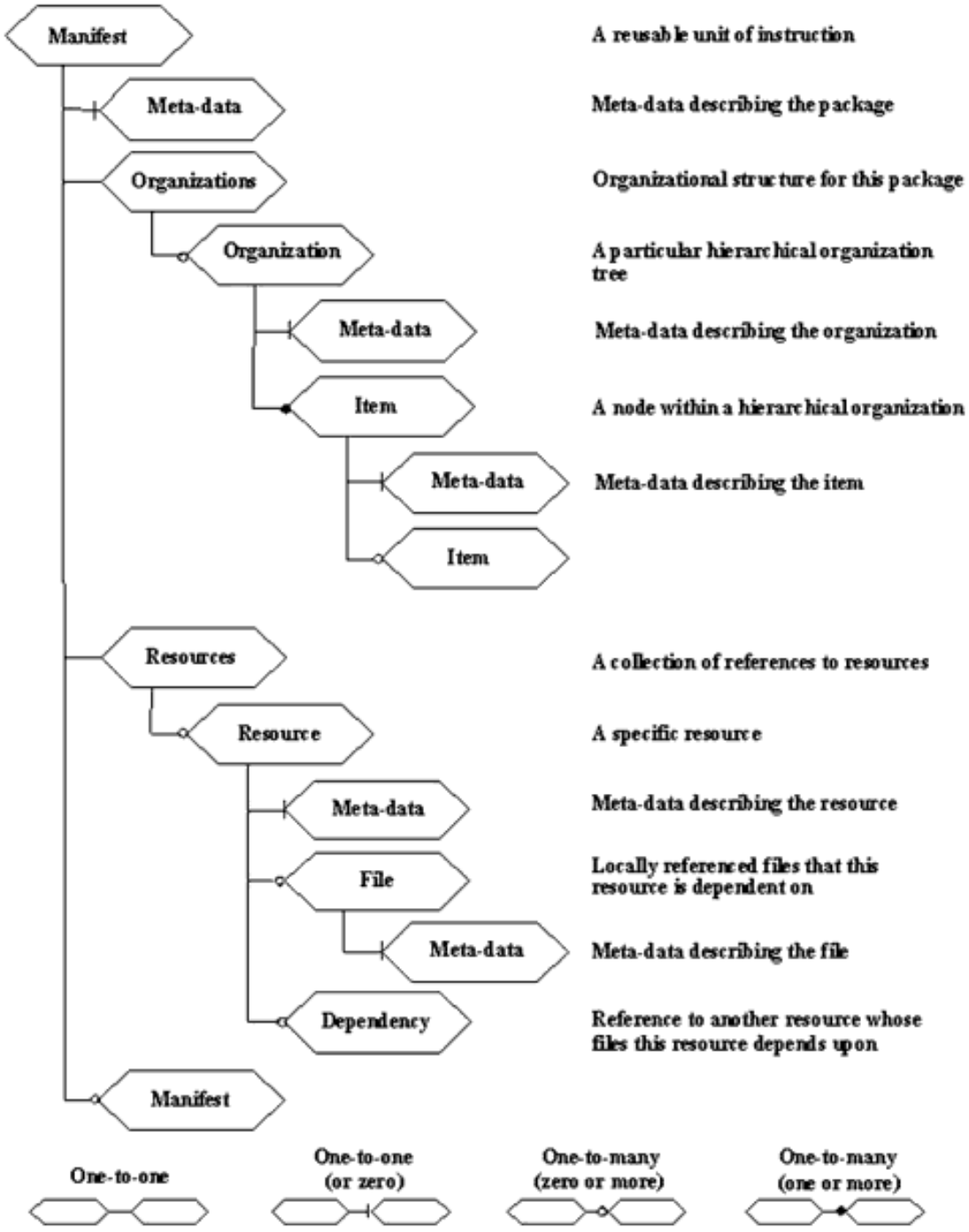
(sub)Manifest(s) Ένα ή περισσότερα, προαιρετικά, λογικά φωλιασμένα manifests.

File Resources (Content) Είναι το πραγματικό περιεχόμενο του πακέτου, δηλαδή αρχεία κειμένου, εικόνες, στοιχεία πολυμέσων και άλλοι πόροι που περιγράφονται μέσα στο manifest. Τα αρχεία αυτά μπορούν να είναι οργανωμένα σε υποκαταλόγους.

Ένα *package* (πακέτο) αποτελεί μια μονάδα χρησιμοποιήσιμου και επαναχρησιμοποιήσιμου περιεχομένου. Θα μπορούσε να περιέχει τμήμα της ύλης ενός μαθήματος, ένα ολόκληρο μάθημα ή μια συλλογή από περισσότερα μαθήματα. Σε κάθε περίπτωση, το package πρέπει να μπορεί να "στέκεται μόνο του" δηλαδή να περιέχει όλη την πληροφορία που απαιτείται για να χρησιμοποιηθούν τα περιεχόμενά του για διδασκαλία.
Δεν είναι απαραίτητο ένα package να ενσωματώνεται σε ένα *αρχείο Package Interchange*. Μπορεί να διανέμεται εναλλακτικά σε ένα CD-ROM ή σε άλλο μέσο αποθήκευσης χωρίς να συμπιέζεται σε ένα αρχείο. Σε κάθε περίπτωση όμως, το αρχείο IMS Manifest με όνομα **imsmanifest.xml** και όσα άλλα XML αρχεία απαιτούνται πρέπει να βρίσκονται στον ριζικό κατάλογο του μέσου διανομής, αλλιώς το πακέτο **δεν** είναι έγκυρο IMS Package.

2.1.2 Η δομή του manifest

Το manifest είναι μια περιγραφή σε γλώσσα XML των πόρων (resources) που συναποτελούν μια διδασκαλία με κάποιο νόημα. Είναι δυνατό να περιέχει επίσης και κανέναν ή περισσότερους πιθανούς τρόπους οργάνωσης των πόρων όταν αυτοί παρουσιάζονται στο μαθητή. Ο δημιουργός του εκπαιδευτικού περιεχομένου έχει την ευχέρεια να αποφασίσει αν θα υπάρχει μόνο ένα αρχείο manifest (που είναι το υποχρεωτικό) στο πακέτο του, ή αν η περιγραφή του πακέτου θα γίνεται μέσα σε περισσότερα αρχεία υπό-manifest. Ο σκοπός των φωλιασμένων υπό-manifests είναι να κάνουν δυνατή τη *συνάθροιση* πολλών πακέτων σε ένα μεγαλύτερο, ή ακόμα και την *αποσύνθεση* ενός σύνθετου πακέτου σε πολλά απλούστερα.
Τα στοιχεία XML που περιέχονται σε ένα manifest, το αν είναι υποχρεωτικά ή προαιρετικά και ο αριθμός τους φαίνονται αναλυτικά στο σχήμα 2.4. Το ριζικό στοιχείο (root element) του manifest πρέπει να είναι ένα στοιχείο <manifest>. Μέσα σε αυτό υπάρχουν τέσσερα υποστοιχεία, που δεν είναι όλα υποχρεωτικά: τα <metadata>, <organizations>, <resources> και τέλος όσα άλλα στοιχεία <manifest> τυχόν χρειάζονται.



Σχήμα 2.4: Τα περιεχόμενα του IMS manifest

Τα meta-data (μετα-δεδομένα) είναι προαιρετικά δεδομένα που περιγράφουν το ίδιο το manifest που τα περιέχει. Συχνά χρησιμοποιούμενα μετα-δεδομένα είναι ο τίτλος, η περιγραφή, λέξεις-κλειδιά, ο ρόλος κάποιου που συνεισέφερε στη δημιουργία του υλικού, ο σκοπός του υλικού (π.χ. αντικείμενο, βαθμός δυσκολίας) και τα πνευματικά δικαιώματα. Το σχήμα XML που χρησιμοποιείται για τα στοιχεία <metadata> πρέπει να ορίζεται σε ένα ειδικό αρχείο τύπου DTD ή XSD και το αρχείο αυτό να δηλώνεται στην αρχή του αρχείου manifest. Τα μετα-δεδομένα πρέπει να προκύπτουν από την προδιαγραφή 1484.12.1-2002 του οργανισμού IEEE για "Learning Object Metadata" (LOM), ενώ και το IMS έχει εκδώσει οδηγίες για να βοηθά στην καλύτερη δυνατή υλοποίηση της προδιαγραφής LOM. Τα meta-data μπορούν να χρησιμοποιηθούν για να περιγράψουν

Τα organizations είναι περιγραφές της στατικής οργάνωσης του περιεχομένου, δηλαδή των πόρων. Είναι υποχρεωτικό να υπάρχει ένα και μοναδικό στοιχείο <organizations>, παιδί του <manifest>, αλλά αν ο δημιουργός του περιεχομένου κρίνει ότι δε χρειάζεται τέτοιο τμήμα που να περιγράφει την οργάνωση των πόρων, μπορεί το στοιχείο να είναι κενό (<organizations/>).
Το μόνο στοιχείο που μπορεί να είναι παιδί του <organizations> είναι το <organization>. Επιτρέπεται να υπάρχουν περισσότερα από ένα τέτοια στοιχεία, καθένα από τα οποία να περιγράφει μια διαφορετική οργάνωση των πόρων. Σε αυτή την περίπτωση όταν το υλικό παρουσιάζεται στον εκπαιδευόμενο, πρέπει να επιλέγεται ένα από τα στοιχεία <organization>. Μπορεί να έχει οριστεί ποιο είναι το προεπιλεγμένο (default) στοιχείο μέσω μιας ιδιότητας του στοιχείου <organizations> ή αλλιώς χρησιμοποιείται το πρώτο στη σειρά στοιχείο ή ακόμα μπορεί το λογισμικό που επεξεργάζεται το content package να κάνει την επιλογή με δικούς του κανόνες ή τέλος να αφήνει το χρήστη να επιλέξει.
Το στοιχείο <organization> περιγράφει μια *ιεραρχική* αναπαράσταση των δεδομένων. Αυτό γίνεται με μια δομή από υποστοιχεία <item>, στοιχεία που μπορούν να εμφανίζονται είτε το ένα μέσα στο άλλο με μια σχέση παιδιού-γονέα σχηματίζοντας μια δενδρική δομή, είτε το ένα στο ίδιο επίπεδο με το άλλο σχηματίζοντας μια επίπεδη δομή. Οι δύο τρόποι μπορούν να χρησιμοποιούνται ταυτόχρονα για να πάρει ο δημιουργός το επιθυμητό μικτό αποτέλεσμα. Κάθε στοιχείο <item> έχει υποχρεωτικά ένα αναγνωριστικό (ιδιότητα identifier) και συνδέεται με πόρους μέσω της ιδιότητας identifierref. Προαιρετικά μπορεί να έχει έναν τίτλο καθώς και μετα-δεδομένα που να περιγράφουν πρόσθετες πληροφορίες (π.χ. για να διευκολύνεται η αναζήτηση).
Ένα παράδειγμα πολλαπλών τρόπων οργάνωσης του ίδιου μαθησιακού υλικού είναι το εξής: Έστω ότι έχουμε υλικό οργανωμένο σε τρία κεφάλαια, καθένα από τα οποία έχει τρεις ενότητες (εισαγωγή, κύριο μέρος, σύνοψη). Ο προεπιλεγμένος τρόπος οργάνωσης θα μπορούσε να είναι απλώς ο ακόλουθος:

1. Κεφάλαιο 1
   (α□) Εισαγωγή 1
   (β□) Κύριο μέρος 1
   (γ□) Σύνοψη 1
2. Κεφάλαιο 2
   (α□) Εισαγωγή 2
   (β□) Κύριο μέρος 2
   (γ□) Σύνοψη 2
3. Κεφάλαιο 3
   (α□) Εισαγωγή 3
   (β□) Κύριο μέρος 3
   (γ□) Σύνοψη 3

Αλλά μετά την παρακολούθηση όλης της ύλης, θα ήταν χρήσιμο να μπορεί να εμφανίζεται στο μαθητή και μια εναλλακτική οργάνωση που να παρουσιάζει μόνο τη σύνοψη κάθε κεφαλαίου, δηλαδή:

1. Σύνοψη 1
2. Σύνοψη 2
3. Σύνοψη 3

Τα resources (πόροι) που περιγράφονται στο manifest είναι στοιχεία όπως ιστοσελίδες, αρχεία κειμένου, αρχεία πολυμέσων, αντικείμενα αξιολόγησης ή άλλα δεδομένα σε μορφή αρχείου. Τα resources μπορούν επίσης να περιλαμβάνουν στοιχεία εξωτερικά του πακέτου που είναι διαθέσιμα μέσω μιας διεύθυνσης URL ή συλλογές από resources που περιγράφονται από τα υπό-manifests. Ο συνδυασμός των resources γενικά χαρακτηρίζεται ως περιεχόμενο (content).
Κάθε πόρος περιγράφεται από ένα στοιχείο XML με όνομα <resource> μέσα στο αρχείο manifest. Το στοιχείο αυτό περιέχει μια λίστα όλων όσων χρειάζονται για να χρησιμοποιηθεί ο πόρος. Τα αρχεία που περιέχονται στο πακέτο τοποθετούνται σε αυτή τη λίστα ως στοιχεία <file> μέσα σε τέτοια στοιχεία <resource>. Κάθε πόρος δεν είναι υποχρεωτικά ένα μόνο αρχείο αλλά μπορεί να είναι και συλλογή αρχείων που υποστηρίζουν την παρουσίαση του αντίστοιχου στοιχείου <item>.
Η τοποθεσία όπου βρίσκεται το αρχείο δηλώνεται μέσω της ιδιότητας href, η οποία είναι μια (τοπική ή απομακρυσμένη) διεύθυνση URL. Για κάθε πόρο υπάρχει η ιδιότητα type, η οποία παίρνει συνήθως την τιμή webcontent, όταν περιγράφει περιεχόμενο που μπορεί να διαβαστεί από έναν φυλλομετρητή διαδικτύου (internet browser), δηλαδή σελίδες HTML, συμπεριλαμβανομένου περιεχομένου που απαιτεί plug-ins όπως flash, πολυμέσα και εκτελέσιμα προγράμματα που μπορούν να εκκινούνται μέσα από τον φυλλομετρητή. Κάθε <resource> και κάθε <file> μπορεί να έχει μετα-δεδομένα που να το περιγράφουν.

Πηγαίος Κώδικας 2.1: Παράδειγμα imsmanifest.xml με πολλαπλές οργανώσεις

```
<?xml version="1.0"?>
<manifest identifier="MANIFEST1"
        xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
        xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_v1p2"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation=
        "http://www.imsglobal.org/xsd/imscp_v1p1
        imscp_v1p1.xsd http://www.imsglobal.org/xsd/imsmd_v1p2
        imsmd_v1p2.xsd">
  <metadata>
    <schema>IMS Content</schema>
    <schemaversion>1.1.3</schemaversion>
    <imsmd:lom>
      <imsmd:general>
        <imsmd:title>
          <imsmd:langstring xml:lang="en-US">
            IMS Content Packaging Sample – Multiple Organizations
          </imsmd:langstring>
        </imsmd:title>
      </imsmd:general>
    </imsmd:lom>
  </metadata>
  <organizations default="TOC1">
    <organization identifier="TOC1" structure="hierarchical">
```

```
    <title>All Lessons</title>
    <item identifier="ITEM1" identifierref="RESOURCE1">
      <title>Lesson 1</title>
      <item identifier="ITEM2" identifierref="RESOURCE2">
        <title>Introduction 1</title>
      </item>
      <item identifier="ITEM3" identifierref="RESOURCE3">
        <title>Content 1</title>
      </item>
      <item identifier="ITEM4" identifierref="RESOURCE4">
        <title>Summary 1</title>
      </item>
    </item>
    <item identifier="ITEM5" identifierref="RESOURCE5">
      <title>Lesson 2</title>
      <item identifier="ITEM6" identifierref="RESOURCE6">
        <title>Introduction 2</title>
      </item>
      <item identifier="ITEM7" identifierref="RESOURCE7">
        <title>Content 2</title>
      </item>
      <item identifier="ITEM8" identifierref="RESOURCE8">
        <title>Summary 2</title>
      </item>
    </item>
    <item identifier="ITEM9" identifierref="RESOURCE9">
      <title>Lesson 3</title>
      <item identifier="ITEM10" identifierref="RESOURCE10">
        <title>Introduction 3</title>
      </item>
      <item identifier="ITEM11" identifierref="RESOURCE11">
        <title>Content 3</title>
      </item>
      <item identifier="ITEM12" identifierref="RESOURCE12">
        <title>Summary 3</title>
      </item>
    </item>
  </organization>
  <organization identifier="TOC2" structure="hierarchical">
    <title>Content Topics</title>
    <item identifier="TOC2_ITEM3" identifierref="RESOURCE3">
      <title>Content 1</title>
    </item>
    <item identifier="TOC2_ITEM7" identifierref="RESOURCE7">
      <title>Content 2</title>
    </item>
    <item identifier="TOC2_ITEM11" identifierref="RESOURCE11">
      <title>Content 3</title>
    </item>
  </organization>
  <organization identifier="TOC3" structure="hierarchical">
```

```
    <title>Summary Topics</title>
    <item identifier="TOC3_ITEM4" identifierref="RESOURCE4">
      <title>Summary 1</title>
    </item>
    <item identifier="TOC3_ITEM8" identifierref="RESOURCE8">
      <title>Summary 2</title>
    </item>
    <item identifier="TOC3_ITEM12" identifierref="RESOURCE12">
      <title>Summary 3</title>
    </item>
  </organization>
</organizations>
<resources>
  <resource identifier="RESOURCE1" type="webcontent"
            href="lesson1.htm">
    <file href="lesson1.htm"/>
  </resource>
  <resource identifier="RESOURCE2" type="webcontent"
            href="intro1.htm">
    <file href="intro1.htm"/>
  </resource>
  <resource identifier="RESOURCE3" type="webcontent"
            href="content1.htm">
    <file href="content1.htm"/>
  </resource>
  <resource identifier="RESOURCE4" type="webcontent"
            href="summary1.htm">
    <file href="summary1.htm"/>
  </resource>
  <resource identifier="RESOURCE5" type="webcontent"
            href="lesson2.htm">
    <file href="lesson2.htm"/>
  </resource>
  <resource identifier="RESOURCE6" type="webcontent"
            href="intro2.htm">
    <file href="intro2.htm"/>
  </resource>
  <resource identifier="RESOURCE7" type="webcontent"
            href="content2.htm">
    <file href="content2.htm"/>
  </resource>
  <resource identifier="RESOURCE8" type="webcontent"
            href="summary2.htm">
    <file href="summary2.htm"/>
  </resource>
  <resource identifier="RESOURCE9" type="webcontent"
            href="lesson3.htm">
    <file href="lesson3.htm"/>
  </resource>
  <resource identifier="RESOURCE10" type="webcontent"
            href="intro3.htm">
```

```
      <file href="intro3.htm"/>
    </resource>
    <resource identifier="RESOURCE11" type="webcontent"
              href="content3.htm">
      <file href="content3.htm"/>
    </resource>
    <resource identifier="RESOURCE12" type="webcontent"
              href="summary3.htm">
      <file href="summary3.htm"/>
    </resource>
  </resources>
</manifest>
```

Ένα από τα σημαντικά χαρακτηριστικά της προδιαγραφής Content Packaging είναι η *επεκτασιμότητα*. Οι δημιουργοί της αναμένουν ότι όσοι επιλέξουν να υλοποιήσουν ένα σύστημα βασισμένο στην προδιαγραφή θα ορίσουν ίσως νέους τύπους <meta-data>, <organizations> και <resources>, αφού θα έχουν αναπτύξει νέες, διαφορετικές προσεγγίσεις γι' αυτά τα στοιχεία. Μια τέτοια επέκταση είναι ουσιαστικά και η προδιαγραφή Simple Sequencing, που θα εξεταστεί στην ενότητα §2.2.

2.2 IMS Simple Sequencing

Η προδιαγραφή Simple Sequencing παρέχει τα μέσα για την αναπαράσταση πληροφοριών που αφορούν την σειρά (ακριβέστερα: ακολουθία, αλληλουχία – sequence) παρουσίασης εκπαιδευτικών δραστηριοτήτων με διάφορους τρόπους.

2.2.1 Μοντέλο

Το sequencing συμβαίνει μέσα σε ένα περιβάλλον χρόνου εκτέλεσης που μεταφράζει συμβάντα πλοήγησης σε αιτήματα πλοήγησης στη μηχανή sequencing.

Εκπαιδευτική δραστηριότητα

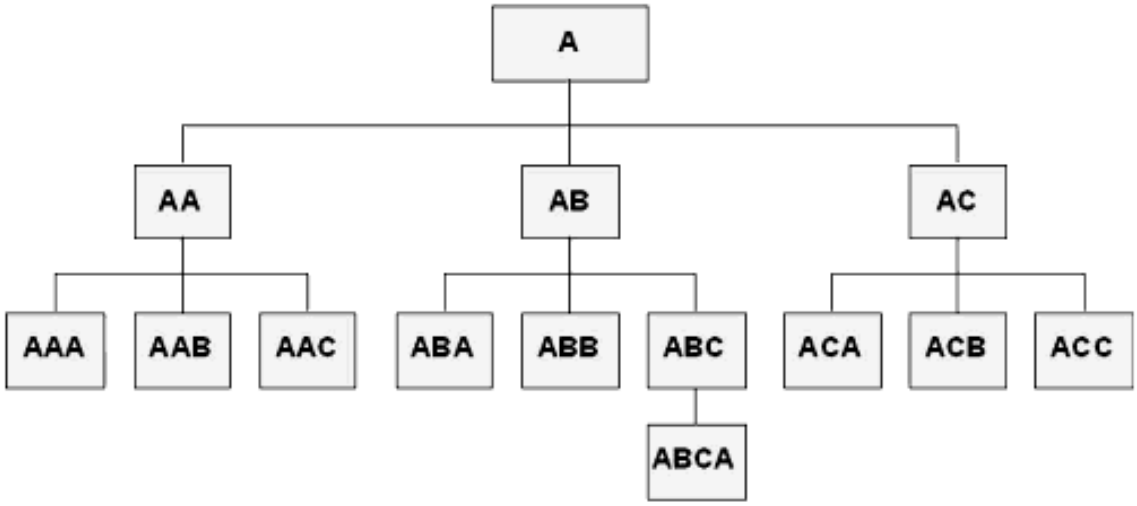
Κεντρικό ρόλο στην προδιαγραφή Simple Sequencing παίζει η έννοια των *εκπαιδευτικών δραστηριοτήτων* (learning activities). Μια εκπαιδευτική δραστηριότητα μπορεί να οριστεί περίπου ως ένα ή περισσότερα συμβάντα διδασκαλίας, ενσωματωμένα μέσα σε ένα πόρο (resource) ή ως μια συνάθροιση δραστηριοτήτων που τελικά αναλύονται σε διακριτούς πόρους, ο καθένας από τους οποίους περιέχει ένα συμβάν διδασκαλίας. Πιο απλά, μια δραστηριότητα μπορεί να είναι μια μονάδα διδασκαλίας, εξέτασης κλπ. Είναι δυνατό η δραστηριότητα να έχει υποδραστηριότητες και να είναι φωλιασμένη σε οποιοδήποτε βάθος. Ο χρήστης μπορεί να δοκιμάσει μια δραστηριότητα όσες φορές θέλει, ή μπορεί ο αριθμός να είναι προκαθορισμένος. Οι δραστηριότητες μπορούν να διακοπούν, να εγκαταλειφθούν, να τερματίσουν κανονικά, κλπ. Όλες οι δραστηριότητες εκτελούνται στο πλαίσιο μιας δραστηριότητας γονέα.

Δέντρο δραστηριοτήτων

Το περιεχόμενο στο Simple Sequencing οργανώνεται με μια ιεραρχική δομή. Κάθε δραστηριότητα μπορεί να έχει μία ή περισσότερες δραστηριότητες–παιδιά, με αποτέλεσμα να σχηματίζεται ένα *δέντρο δραστηριοτήτων* (activity tree), όπως αυτό του σχήματος 2.5 [2].

[2] Σχήματα αυτής της ενότητας: Copyright ©2006 IMS Global Learning Consortium (έγγραφα [4, 5, 6])

Η προδιαγραφή παρέχει τον τρόπο περιγραφής της συμπεριφοράς αλληλουχίας (sequencing behavior), ώστε το LMS να μπορεί να διασχίζει τους κόμβους του δέντρου για να αποφασίζει ποια δραστηριότητα θα παραδώσει στο μαθητή.



Σχήμα 2.5: Δέντρο δραστηριοτήτων (activity tree)

Το δέντρο δεν είναι απαραίτητα ισορροπημένο, ούτε τα κλαδιά του έχουν όλα ίσο βάθος. Η ιεραρχική οργάνωση σημαίνει για παράδειγμα ότι η δραστηριότητα AB θεωρείται μέρος της δραστηριότητας A, ενώ η δραστηριότητα ABCA περιέχεται στην ABC, η οποία είναι μέρος της AB. Η σημασιολογία των δραστηριοτήτων δεν καθορίζεται από την προδιαγραφή, αλλά ο δημιουργός αναθέτει ό,τι νόημα θέλει σε κάθεμια. Για παράδειγμα, κάποιος θα μπορούσε να πει ότι η A είναι ένα μια σειρά μαθημάτων, η AA ένα μάθημα, και η AAA ένα βήμα σε αυτό το μάθημα. Κάποιος άλλος μπορεί να ορίσει την A σαν ένα πρόγραμμα σπουδών, την AA σαν ένα στόχο του προγράμματος, και την AAA σαν ένα μάθημα. Στην προδιαγραφή Simple Sequencing ο κανονικός τρόπος διάσχισης του δέντρου δραστηριοτήτων είναι η διάσχιση pre-order. Στο παράδειγμα του σχήματος 2.5, αυτό σημαίνει ότι ξεκινώντας από τον κόμβο A, η διάσχιση θα επισκεπτόταν κατά σειρά τους κόμβους AA, AAA, AAB, AAC, AB, ABA, κ.ο.κ.

Το προεπιλεγμένο μονοπάτι διάσχισης που περιγράφηκε μπορεί να τροποποιηθεί με τη δημιουργία *κανόνων αλληλουχίας* (sequencing rules) από το δημιουργό του περιεχομένου. Οι κανόνες αυτοί αποτιμώνται κατά το χρόνο εκτέλεσης και μπορεί να περιέχουν συνθήκες που σχετίζονται με το tracking status (δηλ. την παρακολούθηση της προηγούμενης πορείας του μαθητή). Οι δραστηριότητες "παραδίδονται" στο μαθητή πάντα μία κάθε φορά και μπορεί να έχουν βοηθητικούς πόρους συσχετισμένους με αυτές.

Σημείωση: Το δέντρο δραστηριοτήτων μπορεί να αναπαρασταθεί ως ένα στοιχείο <organization> στην προδιαγραφή IMS Content Packaging. Η δραστηριότητα του κορυφαίου επιπέδου είναι το ίδιο το στοιχείο <organization> και κάθε κόμβος του δέντρου είναι ένα <item> μέσα στο <organization>. Παρ' όλα αυτά, η προδιαγραφή Simple Sequencing δεν κάνει καμιά υπόθεση που να αφορά τον τρόπο πακεταρίσματος του εκπαιδευτικού υλικού. Οι ίδιες συμπεριφορές και κανόνες του Simple Sequencing που ορίζονται στην προδιαγραφή θα μπορούσαν να χρησιμοποιηθούν για εκπαιδευτικό υλικό που συλλέγεται και διοικείται δυναμικά από ένα ευφυές σύστημα διδασκαλίας, για παράδειγμα.

Πόροι δραστηριοτήτων

Ένας *πόρος περιεχομένου* (content resource) μπορεί να συσχετιστεί με οποιονδήποτε κόμβο – φύλλο του δέντρου δραστηριοτήτων. Αυτό σημαίνει ότι όταν γίνεται παράδοση μιας δραστηριότητας στο μαθητή, παραδίδεται αυτός ο πόρος. Η προδιαγραφή Simple Sequencing υποθέτει ότι κάποιοι πόροι

έχουν την ικανότητα να δίνουν αποτελέσματα, και κάποιοι άλλοι όχι. Για παραδειγμα, ένας πόρος μπορεί να είναι ένα "ανόητο" έγγραφο ή σελίδα HTML ή μπορεί να είναι ένα SCO όπως ορίζεται στην προδιαγραφή SCORM (βλ. ενότητα **??**). Αν ένας κόμβος – φύλλο δεν έχει κανέναν πόρο συσχετισμένο μαζί του, τότε η συμπεριφορά του συστήματος είναι απροσδιόριστη από την προδιαγραφή.

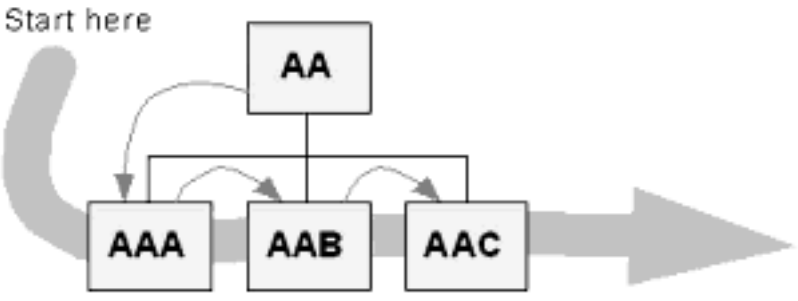
Συστάδες (clusters) δραστηριοτήτων

Οι κανόνες του Simple Sequencing συσχετίζονται με *συστάδες* (clusters) δραστηριοτήτων. Εδώ ο όρος συστάδα αναφέρεται σε έναν κόμβο μαζί με τα άμεσα παιδιά του. Η εμβέλεια ενός συγκεκριμένου κανόνα ποτέ δε φτάνει έξω από τη συστάδα. Για παράδειγμα, πολλοί κανόνες ορίζουν τον τρόπο χειρισμού των υπο-δραστηριοτήτων μιας συστάδας ή τον τρόπο του rollup των αποτελεσμάτων των υπο-δραστηριοτήτων μια συστάδας. Αυτοί οι κανόνες συσχετίζονται με τον κόμβο – γονέα της συστάδας. Άλλοι κανόνες συσχετίζονται μόνο με μια συγκεκριμένη δραστηριότητα και δεν επηρεάζουν τα παιδιά της.

Σχήμα 2.6: Παραδείγματα συστάδων (clusters)

Τρόποι ελέγχου της πλοήγησης (Navigation Control Modes)

Εκτός αν ορίζεται αλλιώς, ένας μαθητής επιτρέπεται να διαλέξει οποιαδήποτε υπο-δραστηριότητα σε μια συστάδα, αλλά είναι δυνατό να επιτρέπεται και μια καθοδηγούμενη *ροή* (flow) μέσα από τη συστάδα. Η ροή μέσα από μια συστάδα είναι η παράδοση καθεμιάς από τις υπο-δραστηριότητες της συστάδας στο μαθητή με τη σειρά όπως φαίνεται στο σχήμα 2.7. Ο κανόνας που ορίζει αν αυτή η ροή είναι ενεργοποιημένη συσχετίζεται με τον κόμβο – γονέα της συστάδας. Αν οι υπο-δραστηριότητες είναι κόμβοι φύλλα με συσχετισμένους πόρους περιεχομένου, τότε η ροή θα παραδόσει καθέναν από αυτούς τους πόρους τον έναν μετά τον άλλο.



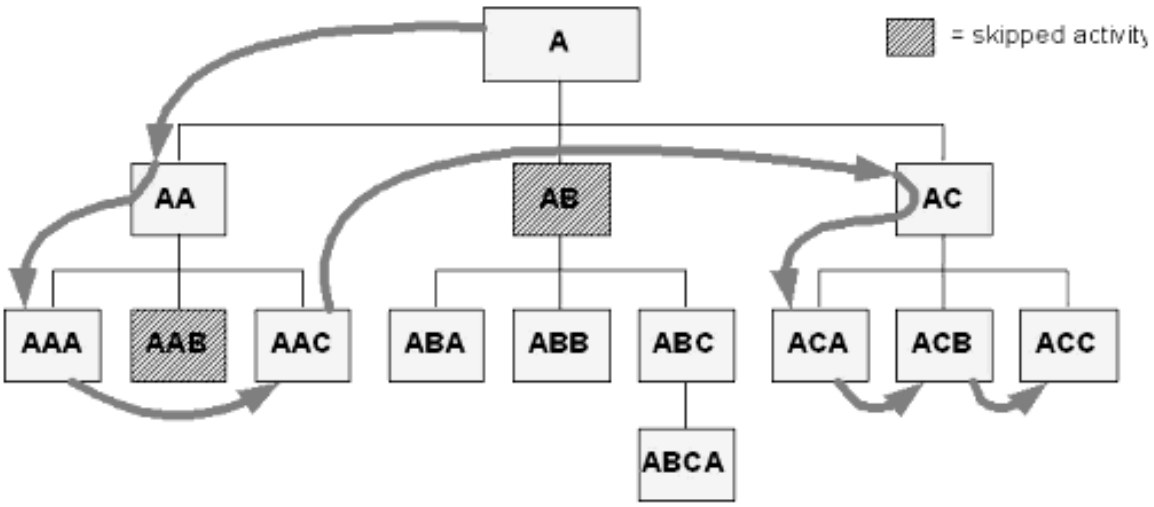
Σχήμα 2.7: Ροή προς τα εμπρός για τη δραστηριότητα AA. Ο εκπαιδευόμενος βλέπει κατά σειρά τις AAA, AAB και τέλος την AAC

Οι βασικοί τρόποι που καθορίζουν αυτή τη συμπεριφορά είναι ο *choice* και ο *flow* καθένας από τους οποίους μπορεί να είναι ενεργοποιημένος ή απενεργοποιημένος. Αν ο choice είναι απενεργοποιημένος, ο χρήστης δεν μπορεί να επιλέξει αλλά πρέπει να ακολουθήσει την καθορισμένη ροή. Αν ο flow

είναι ενεργοποιημένος, ο χρήστης καθοδηγείται μέσα στις δραστηριότητες σύμφωνα με τη δομή του δέντρου. Μπορεί να είναι ενεργοποιημένοι και οι δύο τρόποι ταυτόχρονα, οπότε ο χρήστης μπορεί να επιλέξει οποιαδήποτε δραστηριότητα, αλλά μπορεί επίσης να καθοδηγείται. Όμως δεν μπορεί να είναι και οι δύο τρόποι απενεργοποιημένοι ταυτόχρονα.
Μπορεί να καθοριστούν οι ίδιοι τρόποι ελέγχου της πλοήγησης για όλες τις συστάδες μέσα σε ένα δέντρο δραστηριοτήτων ή μπορεί να είναι διαφορετικοί για διαφορετικές συστάδες. Για παράδειγμα, ένας μαθητής μπορεί να καθοδηγείται με ροή μέσα σε μια αλληλουχία ασκήσεων, μέσα στις οποίες να είναι ελεύθερος να διαλέξει διαφορετικά αντικείμενα εξάσκησης.

Sequencing βασισμένο σε κανόνες – Κανόνες υπό συνθήκη

Είναι δυνατό να επιτευχθούν πιο περίπλοκες συμπεριφορές πλοήγησης με την ανάθεση κανόνων σε διαφορετικούς κόμβους με διαφορετικά βάθη. Το σχήμα 2.8 δείχνει πώς οι κανόνες μπορούν να επηρεάσουν τη διάσχιση του δέντρου όταν ο χρήστης προχωρά από δραστηριότητα σε δραστηριότητα και ο τρόπος flow είναι ενεργοποιημένος. Σε αυτό το παράδειγμα, κάθε συστάδα παίρνει έναν κανόνα που λέγεται "control mode" που ενεργοποιεί τη ροή μέσα από τα παιδιά της, και κάποιοι συγκεκριμένοι κόμβοι παίρνουν έναν υπό συνθήκη κανόνα που ορίζει ότι πρέπει να παραλειφθούν κάτω από συγκεκριμένες προϋποθέσεις.

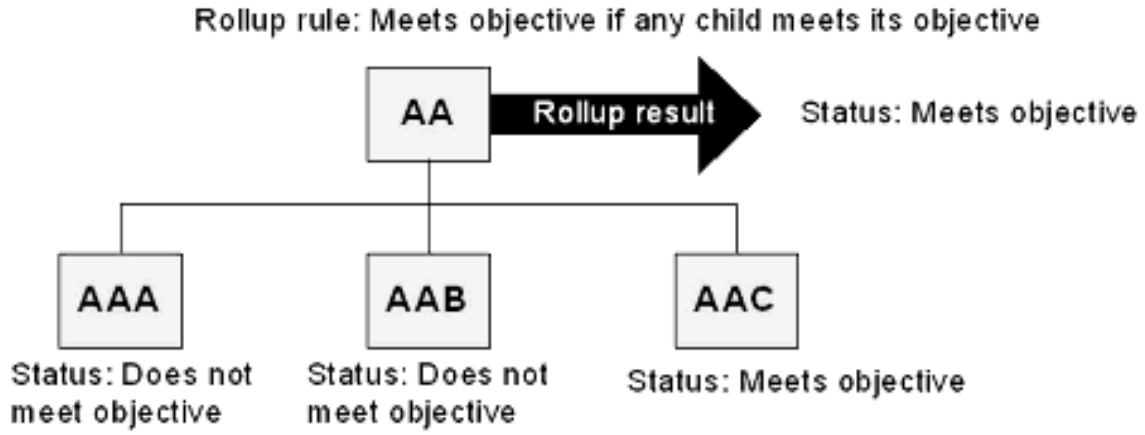


Σχήμα 2.8: Παράδειγμα κανόνων που επηρεάζουν την πλοήγηση για συγκεκριμένους κόμβους δραστηριοτήτων (υπό συνθήκη παράλειψη)

Πολλοί από τους υπό συνθήκη κανόνες που ορίζονται στο Simple Sequencing βασίζονται σε δεδομένα που συλλέγονται από την παρακολούθηση των αποτελεσμάτων των δραστηριοτήτων. Όταν μια δραστηριότητα περιέχει υπο-δραστηριότητες, ή με άλλα λόγια όταν είναι ο κόμβος – γονέας σε μια συστάδα δραστηριοτήτων, τα αποτελέσματα των υπο-δραστηριοτήτων "μαζεύονται" προς τα επάνω (roll up) σε συνοπτικά αποτελέσματα για τη συστάδα. Αυτό δεν είναι πάντα το επιθυμητό. Για παράδειγμα, όταν ένας μαθητής έχει περάσει με επιτυχία μια ελεύθερη εξάσκηση, αυτό δεν πρέπει να επηρεάζει τα αποτελέσματα ενός εξαμηνιαίου μαθήματος. Τα rollups μπορούν να ελέγχονται με δύο τρόπους. Ένας τρόπος είναι με την ανάθεση κανόνων rollup σε μια συστάδα, όπως φαίνεται στο σχήμα 2.9. Ο άλλος τρόπος είναι με την προσθήκη μιας "ετικέττας" σε μια συγκεκριμένη υπο-δραστηριότητα για να διευκρινηστεί ότι αυτή δεν πρέπει να συμπεριλαμβάνεται στα rollups.

Objectives

Οι εκπαιδευτικοί στόχοι (learning objectives) είναι διακεκριμένοι από τις εκπαιδευτικές δραστηριότητες στο Simple Sequencing. Τα objectives αντιπροσωπεύουν ένα σύνολο από δεδομένα τοπικής ή

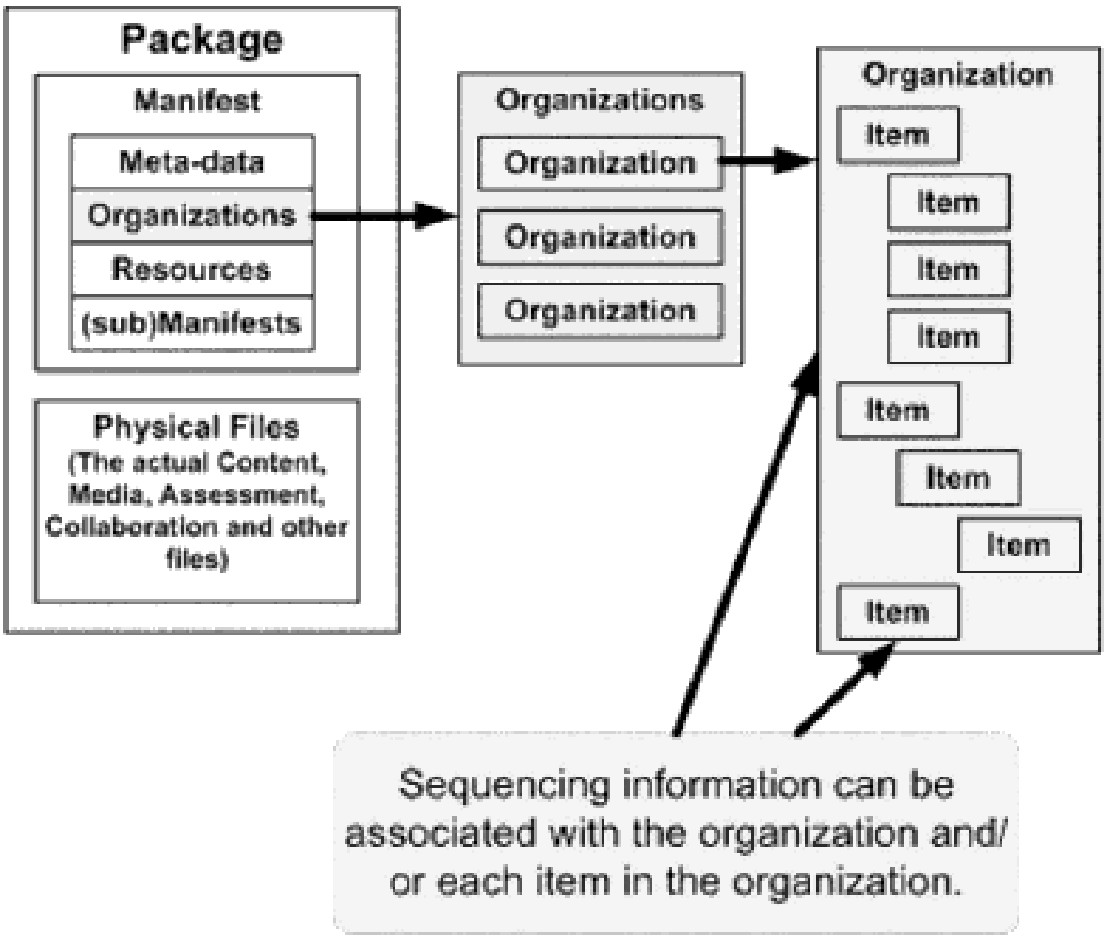


Σχήμα 2.9: Παράδειγμα ενός κανόνα που ελέγχει το rollup των αποτελεσμάτων από τις υπό-δραστηριότητες στον γονέα

καθολικής εμβέλειας, το καθένα με μια κατάσταση ικανοποίησης (μεταβλητή boolean) έναν αριθμό μέτρο ικανοποίησης. Η προδιαγραφή δεν κάνει υποθέσεις σχετικά με το πώς πρέπει να ερμηνεύεται το objective (π.χ. μπορεί να είναι μέτρο ικανότητας ή απλώς μια άλλη τιμή). Οι δραστηριότητες μπορούν να έχουν περισότερα από ένα τοπικά objectives και μπορεί να έχουν αναφορές σε πολλά καθολικά ορισμένα διαμοιραζόμενα objectives. Πολλές δραστηριότητες μπορούν να αναφέρονται στο ίδιο καθολικά ορισμένο objective, με αποτέλεσμα το διαμοιρασμό των τιμών των δεδομένων.

2.2.2 Σχέση με την προδιαγραφή IMS Content Packaging

Όπως προαναφέρθηκε, η προδιαγραφή Simple Sequencing στηρίζεται στην έννοια των *εκπαιδευτικών δραστηριοτήτων* (learning activities). Η προδιαγραφή Content Packaging παρέχει μια έτοιμη δομή για να συσχετίζεται μια εκπαιδευτική δραστηριότητα με έναν πόρο – το στοιχείο *item* και τη σχέση του με το στοιχείο *resource*. Επιπλέον, τα στοιχεία *item* μπορούν να συνθέσουν συλλογές, με τέτοιες συλλογές να περιέχονται σε ένα στοιχείο – γονέα *organization*, ακριβώς όπως οι εκπαιδευτικές δραστηριότητες μπορούν να σχηματίζουν μαζί συστάδες μέσα σε μια δραστηριότητα – γονέα. Επομένως, η προδιαγραφή Simple Sequencing αντιστοιχίζει την έννοια της εκπαιδευτικής δραστηριότητας σε ένα στοιχείο *item* ή σε μια συλλογή στοιχείων *item* μέσα σε ένα στοιχείο *organization* και στο ίδιο το στοιχείο *organization*, όπως αυτό ορίζεται από την προδιαγραφή Content Packaging. Η περιγραφή σε γλώσσα XML του Content Packaging επεκτείνεται από αυτήν την προδιαγραφή για να οριστεί το πώς η πληροφορία που αφορά την αλληλουχία συσχετίζεται με το πακεταρισμένο περιεχόμενο.
Όπως φαίνεται στο σχήμα 2.10, το στοιχείο *organization* και κάθε στοιχείο *item* μέσα σε αυτό μπορεί να έχει ορισμένη συμπεριφορά αλληλουχίας. Μπορούν να υπάρχουν αναφορές από περισσότερα από ένα items σε στοιχεία sequencing definition, ώστε τα στοιχεία αυτά είναι επαναχρησιμοποιήσιμα.
Όλη η πληροφορία που ορίζει συμπεριφορά αλληλουχίας για ένα στοιχείο organization είναι παρούσα στο εσωτερικό του στοιχείου αυτού. Αν σε ένα manifest υπάρχουν πολλαπλές οργανώσεις, κάθε μια από αυτές μπορεί να έχει το δικό της σύνολο κανόνων αλληλουχίας. Μπορεί, για παράδειγμα, η μια οργάνωση να περιέχει απλώς ένα σύνολο περιεχομένου που παρουσιάζεται με την προκαθορισμένη σειρά, ενώ μια άλλη να επιτρέπει πλουσιότερη αλληλεπίδραση με το χρήστη αν περιέχει δεδομένα sequencing.
Η προδιαγραφή IMS Simple Sequencing περιγράφει μόνο το πώς η πληροφορία αλληλουχίας σχετίζεται με την προδιαγραφή IMS Content Packaging, αλλά δεν αποκλείει τη χρήση αυτής της πληροφορίας σε άλλα πλαίσια.
Η προδιαγραφή δε χρησιμοποιεί ούτε και απαγορεύει τη χρήση sub-Manifests, που ορίζονται στην προδιαγραφή Content Packaging.



Σχήμα 2.10: Τρόπος συσχέτισης της πληροφορίας sequencing με ένα Content Package

2.2.3 Περιγραφή σε XML και παραδείγματα

Αυτή η υποενότητα θα δείξει πιο αναλυτικά την περιγραφή σε γλώσσα XML της προδιαγραφής Simple Sequencing, τον τρόπο με τον οποίο τα διάφορα στοιχεία του sequencing μπορούν να συγκεντρωθούν σε ομάδες στοιχείων οι οποίες περιγράφουν την επιθυμητή αλληλουχία των δραστηριοτήτων, όπως και τον τρόπο που η πληροφορία sequencing ενσωματώνεται μέσα σε ένα manifest του IMS Content Packaging. Μια και στην εφαρμογή που αναπτύχθηκε στο πλαίσιο της εργασίας δεν υλοποιήθηκε το σύνολο της προδιαγραφής Simple Sequencing, η έμφαση δίνεται στο υποσύνολο που υλοποιήθηκε. Για μια πληρέστερη περιγραφή της προδιαγραφής, ο αναγνώστης μπορεί να ανατρέξει στα [4, 5].
Σύμφωνα και με αυτά που περιγράφηκαν στην ενότητα §2.2.2, η προαναφερθείσα ενσωμάτωση γίνεται με την συμπερίληψη ενός μοναδικού στοιχείου <sequencing>, είτε μέσα σε ένα στοιχείο <item>, είτε μέσα σε ένα στοιχείο <organization>.

sequencingCollection

Η πληροφορία του sequencing μπορεί να επαναχρησιμοποιηθεί από πολλά διαφορετικά items. Αυτό γίνεται με χρήση της ιδιότητας “IDRef” του στοιχείου <sequencing> και συγκεκριμένα με ανάθεση της ιδιότητας “ID” του άλλου στοιχείου <sequencing> που θέλουμε να χρησιμοποιήσουμε στην ιδιότητα “IDRef”. Τα στοιχεία <sequencing> στα οποία αναφερόμαστε με αυτόν το μηχανισμό πρέπει να αποθηκεύονται σε ένα στοιχείο <sequencingCollection>. Το στοιχείο αυτό είναι απλώς ένα

container για στοιχεία <sequencing> και (αν υπάρχει) τοποθετείται πάντα στο κορυφαίο επίπεδο του manifest, όπως φαίνεται στο παράδειγμα του κώδικα 2.2. Στο παράδειγμα αυτό φαίνεται ένα ελάχιστο imsmanifest.xml με δείκτη από το στοιχείο organization σε πληροφορία sequencing αποθηκευμένη στο κορυφαίο επίπεδο του manifest.
Υπάρχει επίσης η δυνατότητα ένα στοιχείο <sequencing> να έχει και δείκτη (με χρήση του "IDRef") σε στοιχείο που βρίσκεται μέσα σε συλλογή <sequencingCollection> και τοπικά ορισμένες πληροφορίες μέσα του. Σε αυτήν την περίπτωση, χρησιμοποιούνται όλες οι πληροφορίες, αλλά αν υπάρχει σύγκρουση μεταξύ των δύο, τότε οι τοπικοί ορισμοί υπερσκελίζουν τους αντίστοιχους εξωτερικούς.

Πηγαίος Κώδικας 2.2: Παράδειγμα στοιχείων <sequencing> και <sequencingCollection>

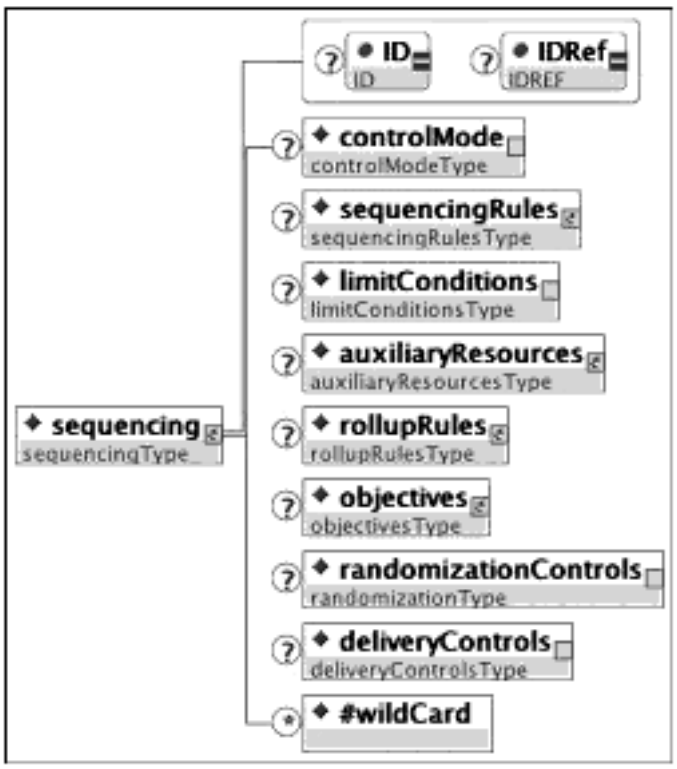
```
<manifest identifier = "IMSSS.TestManifest.1">
  <organizations default = "IMSSS.TestManifest.1.Org.1">
    <organization identifier = "IMSSS.TestManifest.1.Org.1">
      <item identifier = "IMSSS.TestManifest.1.Item.1"/>
      <imsss:sequencing IDRef = "IMSSS.TestManifest.1.Seq.1"/>
    </organization>
  </organizations>
  <resources/>
  <imsss:sequencingCollection>
    <imsss:sequencing ID = "IMSSS.TestManifest.1.Seq.1">
      <!-- Sequencing info goes here -->
    </imsss:sequencing>
  </imsss:sequencingCollection>
</manifest>
```

sequencing

Το στοιχείο <sequencing> εμφανίζεται μία ή περισσότερες φορές μέσα στο στοιχείο <sequencingCollection>, αν αυτό υπάρχει. Επίσης εμφανίζεται καμία ή μία φορά σε κάθε στοιχείο <item> και <organization> μέσα σε ένα IMS content package. Σε κάθε περίπτωση, το στοιχείο αυτό είναι το πιο βασικό στοιχείο της προδιαγραφής, γιατί σε αυτό ενθυλακώνεται όλη η πληροφορία της αλληλουχίας.
Η δομή ενός στοιχείου <sequencing> φαίνεται στο σχήμα 2.11 [3]. Η πληροφορία του sequencing αποτελείται από ένα προαιρετικό αναγνωριστικό ID και μια προαιρετική αναφορά (δείκτη) IDREF, η χρησιμότητα των οποίων εξηγήθηκε παραπάνω, και επιπλέον οχτώ άλλα επίσης προαιρετικά στοιχεία που περιέχουν πληροφορίες του μοντέλου ορισμού, ενώ το τελευταίο προαιρετικό στοιχείο (wildcard) είναι ο μηχανισμός επέκτασης της προδιαγραφής. Το στοιχείο controlMode ορίζει τον τρόπο πλοήγησης για αυτόν τον κόμβο (π.χ. flow ή choice). Οι κανόνες αλληλουχίας (sequencingRules) παρέχουν τις λεπτομέρειες των κανόνων που χρησιμοποιούνται για συμπεριφορά αλληλουχίας. Το στοιχείο

[3] Υπόμνημα για τα σχήματα περιγραφής XML:

- Ορθογώνιο με τετραγωνισμένες γωνίες: στοιχείο XML
- Ορθογώνιο με στρογγυλεμένες γωνίες: ιδιότητα XML
- Έντονα γραμμένο όνομα στο επάνω μέρος του ορθογώνιου: όνομα στοιχείου/ιδιότητας
- Όνομα στο κάτω μέρος του ορθογωνίου: όνομα τύπου
- Ένδειξη πολλαπλότητας (μέσα σε μικρό κύκλο):
  - ? = 0..1
  - * = 0..ν
  - + = 1..ν
  - D = 0..1 με προεπιλεγμένη τιμή, αν λείπει (μόνο για ιδιότητες)
  - (χωρίς ένδειξη) = 1



Σχήμα 2.11: Στοιχείο <sequencing>

limitConditions επιτρέπει τον ορισμό ανώτατων ορίων στον αριθμό προσπαθειών, στο χρόνο ενασχόλησης κτλ., σε αυτόν τον κόμβο. Το στοιχείο auxiliaryResources παρέχει έναν τρόπο για δημιουργία συνδέσμων με άλλους βοηθητικούς πόρους όπως γλωσσάρια, κείμενα βοήθειας, διαδραστικά ηλεκτρονικά τεχνικά εγχειρίδια, κτλ. Οι κανόνες rollup (rollupRules) ελέγχουν το πώς η πληροφορία από τις υπο-δραστηριότητες συνοψίζεται (μαζεύεται) προς τα επάνω σε κόμβους υψηλότερων επιπέδων μέσα στο δέντρο δραστηριοτήτων. Το στοιχείο objectives είναι μια λίστα με εκπαιδευτικούς στόχους που συσχετίζονται με αυτόν τον κόμβο. Το στοιχείο randomizationControls ορίζει πώς γίνεται η επιλογή και η τυχαιοποίηση των δραστηριοτήτων. Το στοιχείο deliveryControls χρησιμοποιείται κατά το χρόνο παράδοσης της δραστηριότητας στο μαθητή.

sequencingRules

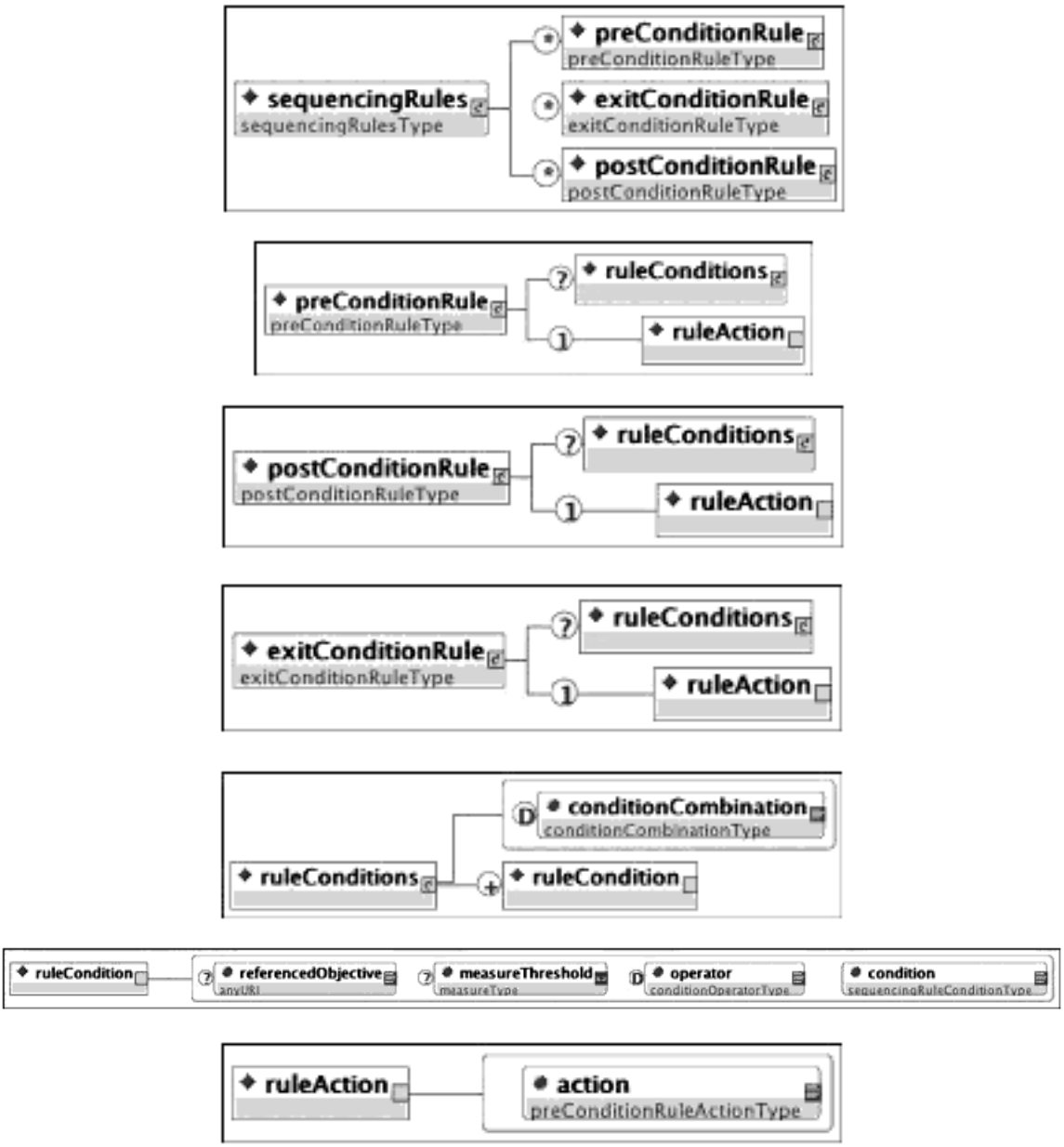
Κάθε δραστηριότητα μπορεί να έχει συσχετισμένο μαζί της έναν απεριόριστο αριθμό κανόνων sequencing. Καθένας από αυτούς τους κανόνες έχει τη γενική μορφή: *Εάν συνθήκη τότε ενέργεια*.
Μπορεί να υπάρχουν πολλαπλές συνθήκες, οι οποίες να συνδυάζονται είτε με έναν συνδυασμό *και* (όλες οι συνθήκες πρέπει να είναι αληθείς) είτε με έναν συνδυασμό *ή* (αρκεί να είναι μόνο μια συνθήκη αληθής). Κάθε συνθήκη μπορεί να αντιστρέφεται πριν συνδυαστεί με τις άλλες για να αποτιμηθεί συνολικά ο κανόνας, αλλά δεν μπορούν να οριστούν άλλοι τελεστές ή συνθήκες. Κάθε συνθήκη αναφέρεται σε ένα στοιχείο του tracking model για μια δραστηριότητα ή σε ένα συσχετισμένο objective. Αν οι συνθήκες του κανόνα αποτιμηθούν αληθείς, ο κανόνας ορίζει μια ενέργεια (action) που πρέπει να εκτελείται ως αποτέλεσμα. Οι ενέργειες διαιρούνται σε τρεις ομάδες:

- Preconditions – Ενέργειες που ελέγχουν αποφάσεις της αλληλουχίας σχετικά με την παράδοση. Κανόνες που περιέχουν τέτοιες ενέργειες χρησιμοποιούνται για να καθορίσουν αν μια συγκεκριμένη δραστηριότητα θα παραδοθεί ή όχι.

- Post Conditions – Ενέργειες που ελέγχουν τη ροή του sequencing δημιουργώντας αιτήματα

αλληλουχίας. Κανόνες που περιέχουν τέτοιες ενέργειες εφαρμόζονται όταν μια δραστηριότητα τερματίζεται.

- Exit Actions – Ενέργειες που τερματίζουν μια δραστηριότητα. Κανόνες που περιέχουν τέτοιες ενέργειες εφαρμόζονται όταν ένας απόγονος μιας δραστηριότητας τερματίζεται.



Σχήμα 2.12: Στοιχείο <sequencingRules>

Στον παράδειγμα του κώδικα 2.3 φαίνεται ένας κανόνας precondition που προκαλεί την απενεργοποίηση (action="disabled") του αντίστοιχου item αν μία από τις δύο συνθήκες (conditionCombination="any") "completed" και "satisfied" είναι αληθής. Αν θέλαμε η απενεργοποίηση να γίνεται όταν και οι δύο συνθήκες είναι αληθείς, θα μπορούσαμε να θέσουμε conditionCombination="all", που είναι και η προεπιλεγμένη τιμή.

Πηγαίος Κώδικας 2.3: Παράδειγμα κανόνα sequencing

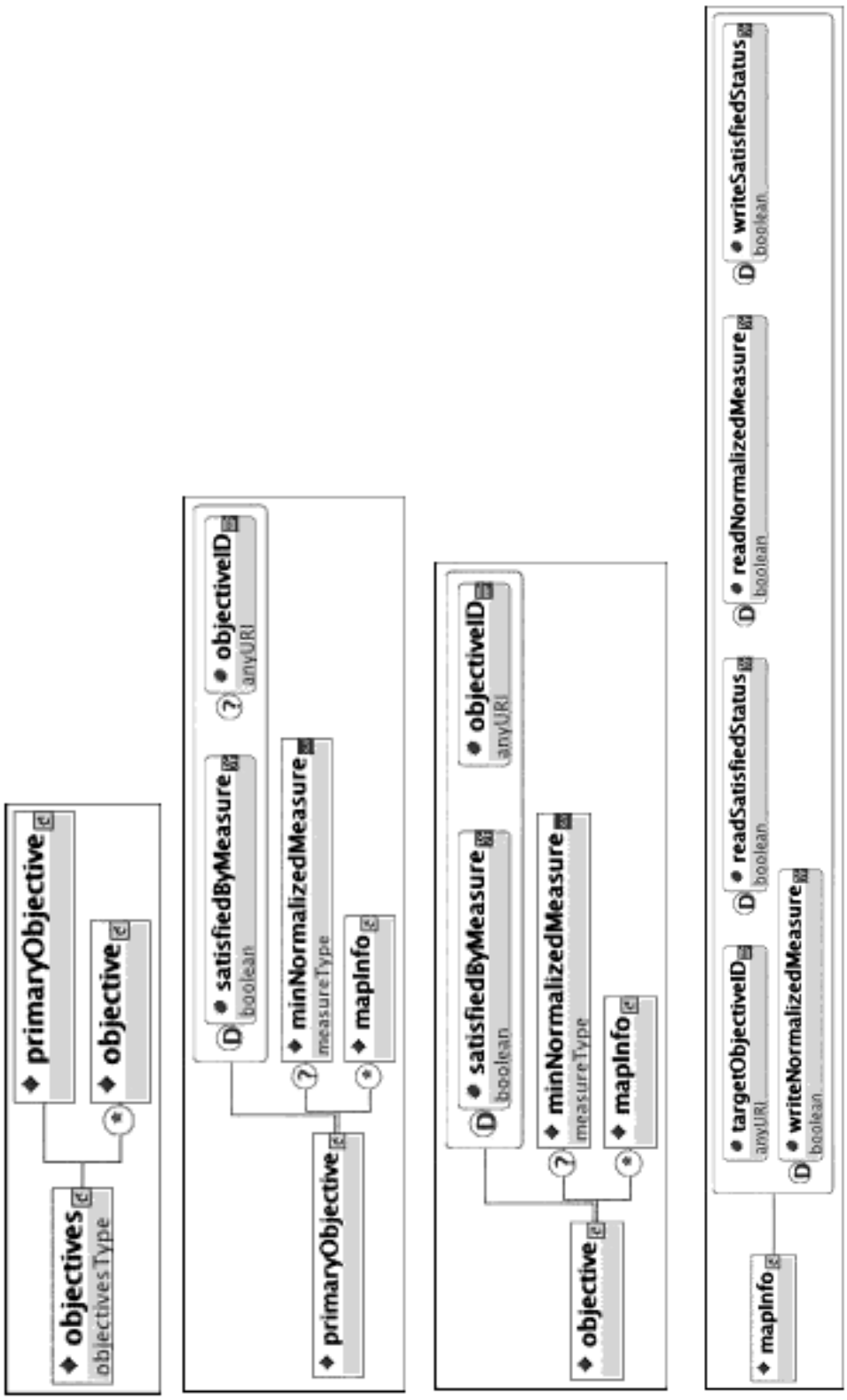
```
<sequencing>
  <sequencingRules>
    <preConditionRule>
      <ruleConditions conditionCombination = "any">
        <ruleCondition condition = "completed"/>
        <ruleCondition condition = "satisfied"/>
      </ruleConditions>
      <ruleAction action = "disabled"/>
    </preConditionRule>
  </sequencingRules>
</sequencing>
```

objectives

Η περιγραφή των εκπαιδευτικών στόχων που συσχετίζονται με κάποια δραστηριότητα γίνεται με το στοιχείο <objectives>, η δομή του οποίου σε γλώσσα XML φαίνεται στο σχήμα 2.13.
Κάθε δραστηριότητα μπορεί να έχει απεριόριστο αριθμό εκπαιδευτικών στόχων. Το μοντέλο παρακολούθησης (tracking model) ορίζει ένα σύνολο δεδομένων που κρατά μια κατάσταση ικανοποίησης (satisfaction status, π.χ. επιτυχία/αποτυχία) και ένα μέτρο (π.χ. βαθμολογία) για κάθε objective. Κάθε δραστηριότητα πρέπει να έχει τουλάχιστον ένα objective, και ακόμα και αν δεν ορίζεται κανένα μέσα στο στοιχείο <sequencing>, θα δημιουργηθεί ένα κατά το χρόνο εκτέλεσης. Ένα μοναδικό objective, το οποίο ονομάζεται *πρωτεύον* (primary objective), συνεισφέρει στο rollup. Αν η δραστηριότητα έχει περισσότερα από ένα objective ή αν η πληροφορία που περιέχεται σε ένα objective πρόκειται να διαμοιραστεί και σε άλλες δραστηριότητες, τότε αυτό απαιτείται να έχει και ένα αναγνωριστικό (objectiveID). Το ελάχιστο μέτρο ικανοποίησης του objective (minNormalizedMeasure) κανονικοποιείται ώστε να παίρνει πραγματικές τιμές ανάμεσα στο στις τιμές -1..1, με ακρίβεια τουλάχιστον τεσσάρων δεκαδικών ψηφίων. Η boolean μεταβλητή satisfied by measure δείχνει αν πρέπει να χρησιμοποιείται το μέτρο για να αποφασίζεται αν το objective έχει ικανοποιηθεί (υπάρχουν και άλλες μέθοδοι).
Οι διεργασίες του Simple Sequencing μπορούν να αναφέρονται στις τοπικές πληροφορίες των objectives για κάθε δραστηριότηττα. Η χρήση των *Objective Maps* επιτρέπει στις διεργασίες να χρησιμοποιούν επίσης και objectives καθολικής εμβέλειας που διαμοιράζονται ανάμεσα στις δραστηριότητες. Το στοιχείο mapInfo περιγράφει την αντιστοίχιση των τοπικών πληροφοριών που αφορούν το objective από (διάβασμα των τιμών satisfaction status και measure) ή πρός (γράψιμο των τιμών) ένα καθολικά διαμοιραζόμενο objective – στόχο, του οποίου το αναγνωριστικό πρέπει να δίνεται (targetObjectiveID).

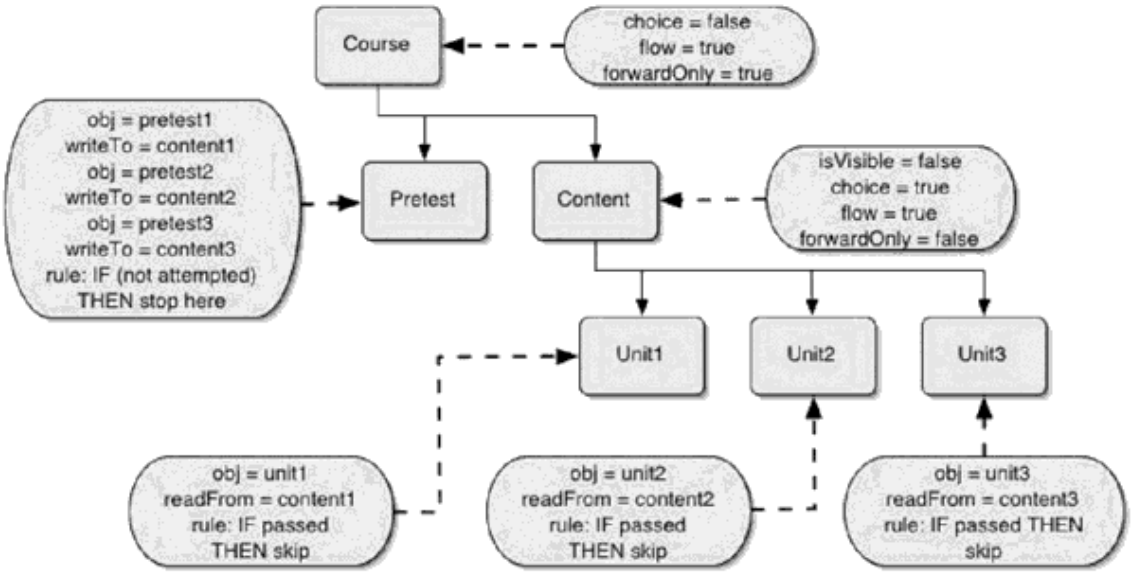
Ένα ολοκληρωμένο παράδειγμα

Δείχνουμε εδώ ένα ολοκληρωμένο παράδειγμα ενός IMS manifest που περιέχει πληροφορίες sequencing. Σε αυτό το παράδειγμα χρησιμοποιούνται κανόνες sequencing τύπου precondition για να παραδοθεί στο μαθητή διαφορετική ύλη ανάλογα με το αποτέλεσμα ενός προκαταρκτικού τεστ. Συγκεκριμένα, στο παράδειγμα έχουμε τα εξής: Υπάρχει μια δραστηριότητα Pretest, την οποία ο μαθητής πρέπει να προσπαθήσει, αλλιώς δεν μπορεί να προχωρήσει παρακάτω. Με βάση το αποτέλεσμα του τεστ αποφασίζεται ποια δραστηριότητα θα παραδοθεί στη συνέχεια: αν το σκορ είναι μικρότερο του 25%, πρέπει να παραδοθεί η δραστηριότητα Remediation και μετά η General, αν είναι μεταξύ 25% και 75% η General, αν είναι μεταξύ 75% και 90% η General και μετά η Advanced, και τέλος αν είναι μεγαλύτερο του 90% η Advanced. Ο τρόπος που επιτυγχάνεται αυτή η συμπεριφορά είναι με τη χρήση του objective content1, στο οποίο γράφεται το αποτέλεσμα του Pretest, και το οποίο διαβάζεται από τις δραστηριότητες Remediation, General και Advanced και συγκρίνεται κατά περίπτωση με τις τιμές 0.25, 0.75 και 0.90 για να αποφασιστεί αν θα παραλειφθούν (skip) ή όχι.



(α□) objectives (β□) primaryObjective (γ□) objective (δ□) mapInfo

Σχήμα 2.13: Στοιχείο <objectives>



Σχήμα 2.14: Διάγραμμα κανόνων sequencing

Πηγαίος Κώδικας 2.4: Παράδειγμα imsmanifest.xml με κανόνες sequencing που υλοποιεί τη συμπεριφορά του σχήματος 2.14

```
<?xml version = "1.0" encoding = "UTF-8"?>
<manifest xmlns = "http://www.imsglobal.org/xsd/imscp_v1p1"
 xmlns:imsss = "http://www.imsglobal.org/xsd/imsss"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation = "http://www.imsglobal.org/xsd/
        imsss Schemas/imsss_v1p0.xsd
        http://www.imsglobal.org/xsd/imscp_v1p1
        Schemas/imscp_v1p1p3.xsd"
 identifier = "BBT_TLCExamplePostTest">
  <!-- This manifest contains a solution to the following ID Problem:
       * Require pretest before content
       * Deliver material based on score in pretest
       * N < 25% => Remediation then General
       * 25% < N < 75% => General only
              * 75% < N < 90% => General then Advanced
                     * N > 90% => Advanced only
       * Allow student to choose any material, regardless of score
    -->
  <organizations>
    <organization identifier = "Organization">
      <item identifier = "Course">
        <item identifier = "Pretest">
          <imsss:sequencing>
            <imsss:sequencingRules>
              <imsss:preConditionRule>
                <imsss:ruleConditions>
                  <imsss:ruleCondition condition = "attempted"
                                       operator = "not"/>
                </imsss:ruleConditions>
```

```
            <imsss:ruleAction action = "stopForwardTraversal"/>
          </imsss:preConditionRule>
        </imsss:sequencingRules>
        <imsss:objectives objectiveID = "pretest1">
          <imsss:primaryObjective>
            <imsss:mapInfo targetObjectiveID = "content1"
                           writeSatisfiedStatus = "true"/>
          </imsss:primaryObjective>
        </imsss:objectives>
      </imsss:sequencing>
    </item>
    <item identifier = "Content" isvisible = "false">
      <item identifier = "Remediation">
        <imsss:sequencing>
          <imsss:sequencingRules>
            <imsss:preConditionRule>
              <imsss:ruleConditions>
                <imsss:ruleCondition condition =
                            "objectiveMeasureGreaterThan"
                             referencedObjective = "remediation1"
                             measureThreshold = "0.25"/>
              </imsss:ruleConditions>
              <imsss:ruleAction action = "skip"/>
            </imsss:preConditionRule>
          </imsss:sequencingRules>
          <imsss:objectives>
            <imsss:primaryObjective objectiveID = "remediation1">
              <imsss:mapInfo targetObjectiveID = "content1"
                             readSatisfiedStatus = "true"/>
            </imsss:primaryObjective>
          </imsss:objectives>
        </imsss:sequencing>
      </item>
      <item identifier = "General">
        <imsss:sequencing>
          <imsss:sequencingRules>
            <imsss:preConditionRule>
              <imsss:ruleConditions>
                <imsss:ruleCondition condition =
                             "objectiveMeasureGreaterThan"
                             referencedObjective = "general1"
                             measureThreshold = "0.90"/>
              </imsss:ruleConditions>
              <imsss:ruleAction action = "skip"/>
            </imsss:preConditionRule>
          </imsss:sequencingRules>
          <imsss:objectives>
            <imsss:primaryObjective objectiveID = "general1">
              <imsss:mapInfo targetObjectiveID = "content1"
                             readSatisfiedStatus = "true"/>
            </imsss:primaryObjective>
```

```
            </imsss:objectives>
          </imsss:sequencing>
        </item>
        <item identifier = "Advanced">
          <imsss:sequencing>
            <imsss:sequencingRules>
              <imsss:preConditionRule>
                <imsss:ruleConditions>
                  <imsss:ruleCondition condition =
                              "objectiveMeasureLessThan"
                              referencedObjective = "advanced1"
                              measureThreshold = "0.75"/>
                </imsss:ruleConditions>
                <imsss:ruleAction action = "skip"/>
              </imsss:preConditionRule>
            </imsss:sequencingRules>
            <imsss:objectives>
              <imsss:primaryObjective objectiveID = "advanced1">
                <imsss:mapInfo targetObjectiveID = "content1"
                              readSatisfiedStatus = "true"/>
              </imsss:primaryObjective>
            </imsss:objectives>
          </imsss:sequencing>
        </item>
        <imsss:sequencing>
          <imsss:controlMode choice = "true"
                    flow="true" forwardOnly = "false"/>
          </imsss:sequencing>
      </item>
      <imsss:sequencing>
        <imsss:controlMode choice = "false"
              flow = "true" forwardOnly = "true"/>
      </imsss:sequencing>
    </item>
  </organization>
 </organizations>
 <resources/>
</manifest>
```

2.3 Εναλλακτική οπτική αναπαράσταση

Τα εργαλεία authoring που σκοπεύουν να βοηθούν στη δημιουργία εκπαιδευτικού περιεχομένου το οποίο να περιγράφεται σύμφωνα με κάποια προδιαγραφή, μπορούν να χωριστούν σε δύο κατηγορίες:

Στα εργαλεία χαμηλού επιπέδου, στα οποία το εργαλείο ακολουθεί από κοντά την προδιαγραφή, που σημαίνει ότι ο συγγραφέας πρέπει να γνωρίζει τις τεχνικές λεπτομέρειες της συγκεκριμένης προδιαγραφής.

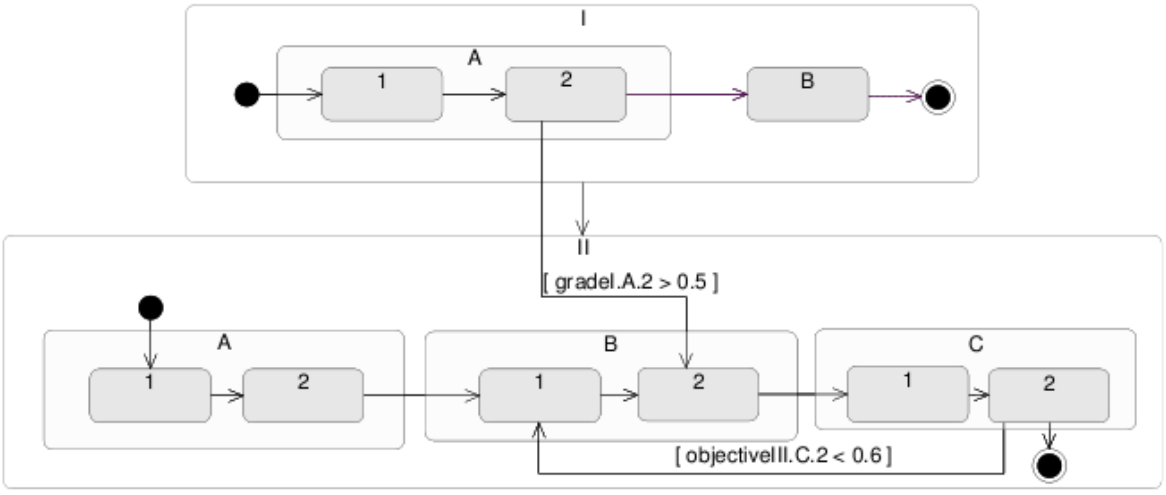
Στα εργαλεία υψηλού επιπέδου, στα οποία το εργαλείο φροντίζει να αποκρύπτει τις λεπτομέρειες της προδιαγραφής από το δημιουργό του εκπαιδευτικού περιεχομένου.

Η προδιαγραφή Simple Sequencing, όπως είναι φανερό και από την περιγραφή της στην ενότητα §2.2, δεν είναι τόσο απλή όσο υποδηλώνει το όνομά της. Αντίθετα, είναι μάλλον περίπλοκη και το μοντέλο δεδομένων που χρησιμοποιεί δεν είναι ιδιαίτερα διαισθητικό, με αποτέλεσμα να απαιτείται σημαντική επένδυση σε χρόνο και σε προσπάθεια για να τη μάθει και να τη χρησιμοποιήσει κανείς. Επομένως, για τη δημιουργία εκπαιδευτικού περιεχομένου σύμφωνου με την προδιαγραφή Simple Sequencing από τους περισσότερους εκπαιδευτικούς, το καταλληλότερο μέσο είναι ένα εργαλείο υψηλού επιπέδου.

2.3.1 Περιγραφή με διάγραμμα καταστάσεων

Στο [7] προτείνεται ένας τρόπος οπτικής αναπαράστασης ενός εγγράφου Simple Sequencing ο οποίος χρησιμοποιεί διαγράμματα καταστάσεων για να περιγράψει τη δενδρική δομή των εκπαιδευτικών δραστηριοτήτων (activities) και τους πιθανούς τρόπους πλοήγησης μέσα στο δένδρο αυτό, καθώς και για να αποκρύψει από το συγγραφέα του εκπαιδευτικού περιεχομένου τις λεπτομέρειες της υλοποίησης, δηλαδή κυρίως την περιγραφή με γλώσσα XML της προδιαγραφής.

Το σχήμα 2.15 δείχνει ένα παράδειγμα τέτοιου διαγράμματος καταστάσεων το οποίο καθορίζει τη συμπεριφορά κάποιου εκπαιδευτικού περιεχομένου. Όπως φαίνεται στο σχήμα, υπάρχουν δύο είδη μεταβάσεων στα διαγράμματα καταστάσεων. Το ένα είδος περιγράφει την προεπιλεγμένη σειρά των κόμβων, η οποία υπονοείται από τη δομή του δέντρου και υποδεικνύεται από μια pre-order διάσχιση του δέντρου. Ονομάζουμε αυτές τις μεταβάσεις *ακολουθιακές μεταβάσεις* (order transitions). Το δεύτερο είδος μεταβάσεων είναι αυτές στις οποίες ανατίθενται κάποιες συνθήκες, και εκτελούνται όταν αυτές οι συνθήκες αποτιμώνται ως αληθείς. Αυτές οι μεταβάσεις ονομάζονται *υπο συνθήκη μεταβάσεις* (conditional transitions). Η αρχική κατάσταση μιας υπο συνθήκη μετάβασης πρέπει να είναι μια βασική κατάσταση, δηλαδή μια κατάσταση που δεν έχει υπο-καταστάσεις. Οι υπο συνθήκη μεταβάσεις μπορούν να είναι είτε *μεταβάσεις προς τα εμπρός* (forward transitions), στην περίπτωση που ο κόμβος προορισμού έπεται του αρχικού κόμβου καθώς διασχίζουμε το δέντρο των activities με διάσχιση pre-order, είτε *μεταβάσεις προς τα πίσω* (backwards transitions), στην αντίθετη περίπτωση.



Σχήμα 2.15: Παράδειγμα διαγράμματος καταστάσεων

Η αναπαράσταση με διαγράμματα καταστάσεων είναι από τη φύση της διαφορετική από το Simple Sequencing στη σημασιολογία της πλοήγησης. Γι αυτό το λόγο χρειάζεται και ένας αλγόριθμος μετασχηματισμού του διαγράμματος καταστάσεων σε ένα έγκυρο έγγραφο Simple Sequencing, ο οποίος περιγράφεται επίσης στο [7].

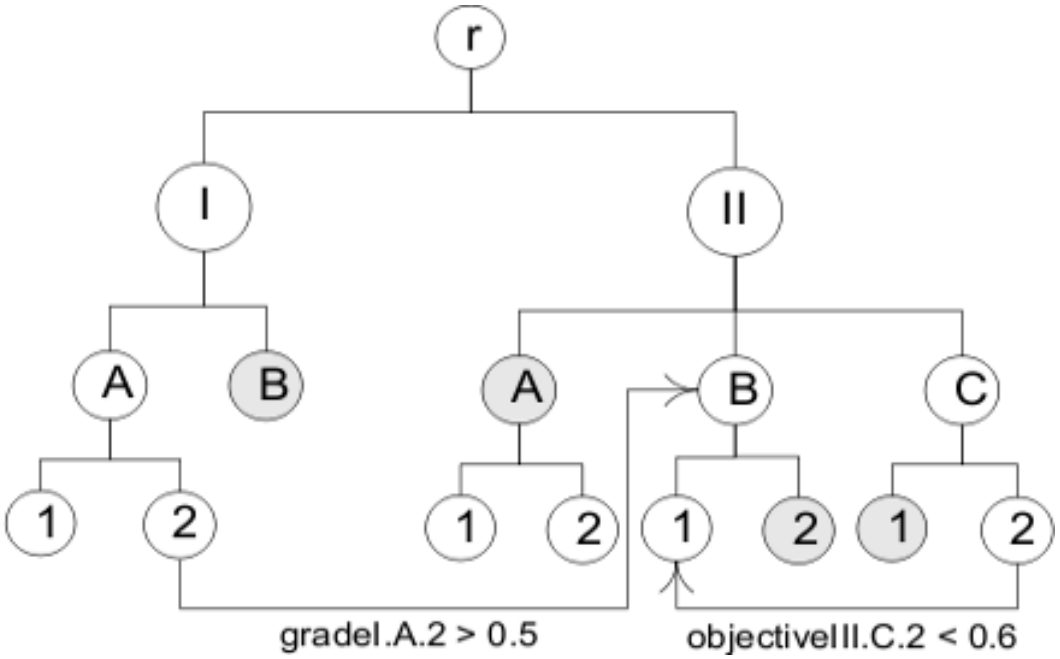
2.3.2 Μετασχηματισμός διαγράμματος καταστάσεων σε περιγραφή Simple Sequencing

Σε αυτήν την ενότητα θα δείξουμε πώς το διάγραμμα του σχήματος 2.15 πρόκειται να μετασχηματιστεί σε μια περιγραφή Simple Sequencing.
Ο αλγόριθμος μετασχηματισμού παίρνει ως είσοδο το δέντρο του διαγράμματος καταστάσεων. Αυτό το δέντρο ορίζει μια ιεραρχία κόμβων οι οποίοι αντιστοιχούν σε καταστάσεις και στις υπό-καταστάσεις τους. Σε κάθε κόμβο είναι δυνατό να ανατεθούν ένα ή περισσότερα *objectives*. Κάθε objective σχετίζεται με ένα *measure* (μέτρο), το οποίο είναι μια αριθμητική τιμή. Το διάγραμμα περιέχει επίσης μεταβάσεις, τόσο υπό συνθήκη όσο και χωρίς συνθήκες. Οι υπό συνθήκη μεταβάσεις περιέχουν *λογικές εκφράσεις* καθεμιά από τις οποίες αναφέρεται σε κάποιο ήδη υπάρχον μέτρο ενός objective. Το δέντρο Simple Sequencing σχηματίζεται από το δέντρο του διαγράμματος καταστάσεων με τον τρόπο που περιγράφεται στον αλγόριθμο 1.

Αλγόριθμος 1 Μετασχηματισμός διαγράμματος καταστάσεων σε περιγραφή Simple Sequencing

1: **για κάθε** κόμβο στο διάγραμμα καταστάσεων **κάνε**
2: Δημιούργησε έναν αντίστοιχο κόμβο στο δέντρο των Simple Sequencing activities
3: Πρόσθεσε τις υπό-καταστάσεις του κόμβου ως παιδιά του αντίστοιχου activity
4: {Η σειρά των παιδιών ενός activity καθορίζεται από τις ακολουθιακές μεταβάσεις του διαγράμματος καταστάσεων.}
5: **τέλος για**
6: **για κάθε** υπο συνθήκη μετάβαση **κάνε**
7: Α ← activity εκκίνησης
8: Β ← activity προορισμού
9: ΚΚΠ ← *κοντινότερος κοινός πρόγονος* (Α, Β)
10: σύνολο Μ1 ← τα δεξιά αδέλφια των κόμβων που ανήκουν στο μονοπάτι από τον κόμβο Α έως τον κόμβο πριν από τον ΚΚΠ, με εξαίρεση τον τελευταίο.
11: σύνολο Μ2 ← τα αριστερά αδέλφια των κόμβων που ανήκουν στο μονοπάτι από τον κόμβο Β έως τον κόμβο πριν από τον ΚΚΠ, με εξαίρεση τον τελευταίο.
12: σύνολο Π ← τα παιδιά του ΚΚΠ που βρίσκονται μεταξύ των μονοπατιών των κόμβων Α και Β {τα οποία περιγράφηκαν παραπάνω}
13: **για κάθε** κόμβο ∈ (Μ1∪Μ2∪Π) **κάνε**
14: Πρόσθεσε έναν κανόνα precondition στον κόμβο
15: {Η συνθήκη του κανόνα προκύπτει από τη συνθήκη της υπό συνθήκη μετάβασης. Η αντίστοιχη ενέργεια είναι η *skip*.}
16: **τέλος για**
17: **εάν** η υπό συνθήκη μετάβαση είναι μετάβαση προς τα πίσω **τότε**
18: Πρόσθεσε έναν κανόνα postcondition στον κόμβο Α
19: {Η συνθήκη του κανόνα προκύπτει από τη συνθήκη της αντίστοιχης υπό συνθήκη μετάβασης και η ενέργεια είναι η *previous*.}
20: **τέλος εάν**
21: **τέλος για**

Στο σχήμα 2.16 φαίνεται μια εναλλακτική αναπαράσταση του διαγράμματος καταστάσεων του σχήματος 2.15. Σε αυτήν την αναπαράσταση, η δομή των κόμβων καταστάσεων είναι ίδια με τη δομή των activities στην περιγραφή Simple Sequencing. Ο κόμβος *r* είναι η ρίζα του αντίστοιχου δέντρου των activities.
Σε αυτό το δέντρο απεικονίζονται δύο υπό συνθήκη μεταβάσεις. Η πρώτη είναι η μετάβαση προς τα εμπρός από τον κόμβο *r.I.A.2* στον κόμβο *r.II.B*. Η εφαρμογή του παραπάνω αλγόριθμου δίνει ως αποτέλεσμα για τη συγκεκριμένη μετάβαση τον ορισμό κανόνων *skip* στους σκιασμένους κόμβους *r.I.B* και *r.II.A*. Η δεύτερη μετάβαση είναι η μετάβαση προς τα πίσω από τον κόμβο *r.II.C.2* στον κόμβο *r.II.B.1*. Η εφαρμογή του αλγόριθμου ορίζει κανόνες *skip* στους σκιασμένους κόμβους *r.II.B.2*



Σχήμα 2.16: Δενδρική αναπαράσταση του διαγράμματος καταστάσεων

και *r.II.C.1* και επιπλέον έναν κανόνα postcondition με ενέργεια *previous* στον κόμβο *r.II.C.2*, ως αποτέλεσμα αυτής της μετάβασης.

Κεφάλαιο 3

Σχεδιασμός, ανάπτυξη, έλεγχος της εφαρμογής

3.1 Η πλατφόρμα Eclipse

Το Eclipse είναι μια κοινότητα λογισμικού ανοιχτού κώδικα, τα έργα της οποίας εστιάζουν στην κατασκευή μιας ανοιχτής πλατφόρμας ανάπτυξης, η οποία αποτελείται από επεκτάσιμα πλαίσια, εργαλεία και περιβάλλοντα χρόνου εκτέλεσης για την κατασκευή και τη διαχείριση λογισμικού κατά τη διάρκεια ολόκληρου του κύκλου ζωής του. Το ίδρυμα Eclipse είναι ένας μη κερδοσκοπικός οργανισμός που υποστηρίζεται από τα μέλη του και φιλοξενεί τα έργα Eclipse.
Το έργο Eclipse δημιουργήθηκε αρχικά από την IBM το Νοέμβριο του 2001 και υποστηρίζεται από μια κοινοπραξία εταιρειών πώλησης λογισμικού.

3.1.1 Αρχιτεκτονική της πλατφόρμας

Το Eclipse είναι μια πλατφόρμα που έχει σχεδιαστεί από την αρχή για να χτίζονται πάνω της ολοκληρωμένα εργαλεία ανάπτυξης εφαρμογών διαδικτύου και εφαρμογών λογισμικού. Από το σχεδιασμό της, η πλατφόρμα δεν παρέχει μεγάλο επίπεδο χρηστικότητας από μόνη της. Η αξία της όμως βρίσκεται στο γεγονός ότι ενθαρρύνει την ανάπτυξη επιπλέον δυνατοτήτων βασισμένων σε ένα μοντέλο από plug-ins.
Το eclipse προσφέρει μια κοινή διαπροσωπεία προς το χρήστη (User Interface) για να δουλεύει με εργαλεία. Είναι σχεδιασμένο να λειτουργεί σε πολλά διαφορετικά λειτουργικά συστήματα ενώ ταυτόχρονα παρέχει στιβαρή ολοκλήρωση με το κάθε λειτουργικό σύστημα που βρίσκεται κάτω από αυτό. Τα plug-ins μπορούν να στηρίζονται στα φορητά APIs του eclipse και έτσι να λειτουργούν αμετάβλητα σε οποιοδήποτε από τα υποστηριζόμενα λειτουργικά συστήματα.
Στον πυρήνα του eclipse βρίσκεται μια αρχιτεκτονική για δυναμική αναζήτηση, φόρτωση και εκτέλεση των plug-ins. Η πλατφόρμα χειρίζεται τις λειτουργίες της εύρεσης και εκτέλεσης του σωστού κώδικα. Η διαπροσωπεία χρήστη της πλατφόρμας παρέχει ένα τυποποιημένο μοντέλο πλοήγησης. Κάθε plug-in μπορεί στη συνέχεια να εστιάσει στην σωστή υλοποίηση ενός μικρού αριθμού εργασιών. Παραδείγματα τέτοιων εργασιών είναι η κωδικοποίηση, ο έλεγχος, η μεταγλώττιση, η αποσφαλμάτωση, η δημιουργία διαγραμμάτων, κτλ.

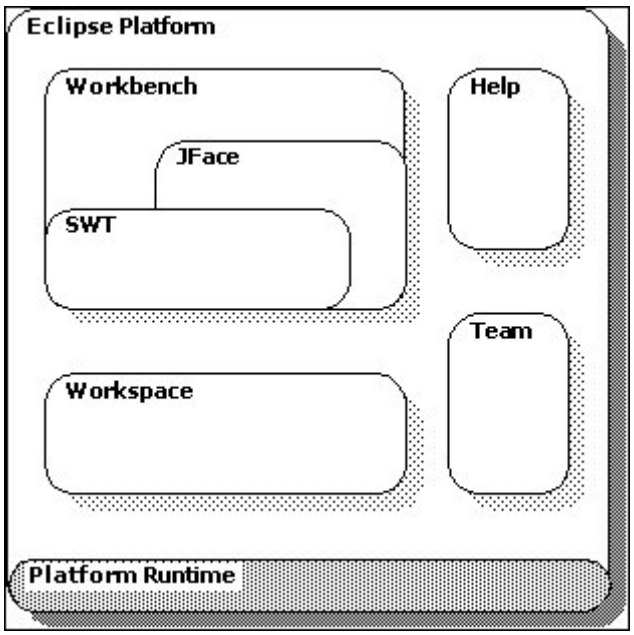
Ανοιχτή αρχιτεκτονική

Η πλατφόρμα eclipse ορίζει μια ανοιχτή αρχιτεκτονική έτσι ώστε η κάθε ομάδα ανάπτυξης plug-in να μπορεί να εστιάζει στη δική της περιοχή εξειδίκευσης. Η ιδέα είναι ότι οι ειδικοί στον τομέα τους χτίζουν το οπίσθιο μέρος και οι ειδικοί στην χρηστικότητα χτίζουν τα εργαλεία που θα χρησιμοποιεί ο τελικός χρήστης (τα plug-ins). Αν η πλατφόρμα είναι καλά σχεδιασμένη, τότε σημαντικές νέες δυνατότητες και νέα επίπεδα ολοκλήρωσης μπορούν να προστεθούν σε αυτήν χωρίς να επηρεάζονται τα υπόλοιπα εργαλεία.
Η πλατφόρμα eclipse χρησιμοποιεί το μοντέλο του πάγκου εργασίας (workbench) για να ενσωματώνει τα εργαλεία από τη σκοπιά του τελικού χρήστη. Εργαλεία που αναπτύσσει ο καθένας μπορούν να ενσωματωθούν στον πάγκο εργασίας με τη χρήση καλά ορισμένων διεπαφών που ονομάζονται σημεία επέκτασης (extension points).

Και η ίδια η πλατφόρμα είναι χτισμένη σε στρώματα από plug-ins, καθένα από τα οποία ορίζει επεκτάσεις πάνω σε σημεία επεκτάσεων των plug-ins του χαμηλότερου στρώματος. Με τη σειρά τους και αυτά τα plug-ins ορίζουν τα δικά τους σημεία επέκτασης για να παρέχουν τη δυνατότητα περαιτέρω προσαρμογής από plug-ins ανώτερων στρωμάτων. Αυτό το μοντέλο επέκτασης επιτρέπει σε ομάδες ανάπτυξης plug-ins να προσθέσουν πολλές και διάφορες λειτουργίες στη βασική πλατφόρμα εργαλείων. Τα διάφορα τεχνήματα για κάθε εργαλείο, όπως αρχεία και άλλα δεδομένα, συντονίζονται από ένα κοινό για όλη την πλατφόρμα μοντέλο διαχείρισής τους.
Η πλατφόρμα δίνει στους χρήστες έναν κοινό τρόπο να δουλεύουν με τα εργαλεία και παρέχει διαχείριση των πόρων (resources) που αυτοί δημιουργούν με τα plug-ins. Οι ομάδες ανάπτυξης των plug-ins επωφελούνται επίσης από αυτή την αρχιτεκτονική, καθώς η πλατφόρμα διαχειρίζεται την περιπλοκότητα των διαφορετικών περιβαλλόντων εκτέλεσης, για παράδειγμα των διαφορετικών λειτουργικών συστημάτων.

Η δομή της πλατφόρμας

Η πλατφόρμα είναι δομημένη σαν υποσυστήματα που υλοποιούνται από ένα ή περισσότερα plug-ins το καθένα. Τα υποσυστήματα είναι χτισμένα πάνω σε μια μικρή μηχανή χρόνου εκτέλεσης (runtime engine). Το σχήμα 3.1 δείχνει μια απλοποιημένη όψη.

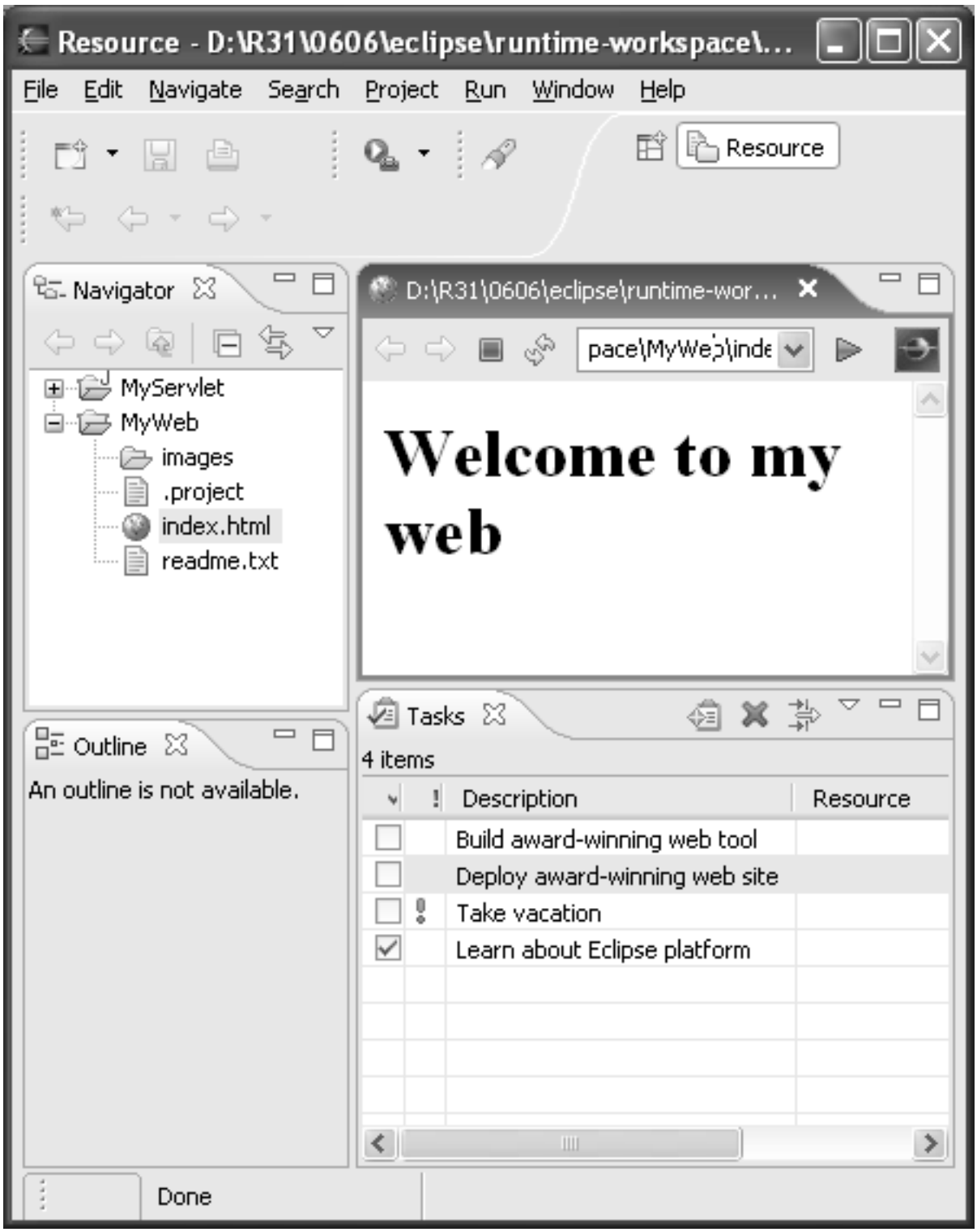


Σχήμα 3.1: Δομή της πλατφόρμας eclipse

Τα plug-ins που απαρτίζουν ένα υποσύστημα ορίζουν σημεία επέκτασης για πρόσθεση νέας συμπεριφοράς στο σύστημα. Ο πίνακας 3.1 περιγράφει τις κύριες ψηφίδες της πλατφόρμας που είναι υλοποιημένες σαν ένα ή περισσότερα plug-ins.

Αρχική μορφή της πλατφόρμας

Η βασική πλατφόρμα που παρέχεται στο χρήστη αρχικά είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment, IDE) για οτιδήποτε και τίποτα συγκεκριμένο (σχήμα 3.2).
Τα plug-ins είναι αυτά που καθορίζουν την τελική λειτουργικότητα της πλατφόρμας. Τα νέα plug-ins μπορούν να παρέχουν υποστήριξη για επεξεργασία και διαχείριση επιπρόσθετων τύπων πόρων, όπως αρχεία Java, προγράμματα C, σελίδες HTML και αρχεία JSP.



Σχήμα 3.2: Αρχική μορφή του πάγκου εργασίας του eclipse

Platform Runtime	Ορίζει το μοντέλο των σημείων επέκτασης και το μοντέλο των plug-ins. Ανακαλύπτει δυναμικά τα plug-ins και διατηρεί τις πληροφορίες για αυτά και για τα σημεία επέκτασης σε ένα μητρώο (registry) της πλατφόρμας. Τα plug-ins εκκινούνται όταν χρειάζεται, σύμφωνα με τις ενέργειες του χρήστη. Το περιβάλλον χρόνου εκτέλεσης υλοποιείται σύμφωνα με την τεχνολογία OSGi.
Resource management (workspace)	Ορίζει API για δημιουργία και διαχείριση των πόρων (resources), όπως έργα, αρχεία και φάκελοι, οι οποίοι παράγονται από τα εργαλεία και αποθηκεύονται στο σύστημα αρχείων.
Workbench UI	Υλοποιεί τη "θέση οδήγησης" του χρήστη για να περιηγείται μέσα στην πλατφόρμα. Ορίζει σημεία επέκτασης για προσθήκη ψηφίδων διαπροσωπείας χρήστη όπως όψεις ή ενέργειες μενού. Παρέχει πρόσθετες εργαλειοθήκες (JFace και SWT) για δημιουργία διεπαφών χρήστη. Οι υπηρεσίες διαπροσωπείας χρήστη είναι δομημένες έτσι ώστε ένα υποσύνολο των plug-ins μπορεί να χρησιμοποιηθεί για να δημιουργηθούν εφαρμογές rich client οι οποίες να είναι ανεξάρτητες από το μοντέλο του workspace. Τα plug-ins που σχετίζονται με εφαρμογές ολοκληρωμένου περιβάλλοντος ανάπτυξης (Integrated Development Environment, IDE) ορίζουν επιπλέον τρόπους πλοήγησης και διαχείρισης των πόρων.
Help system	Ορίζει σημεία επέκτασης που τα plug-ins μπορούν να χρησιμοποιούν για να παρέχουν βοήθεια ή άλλου είδους τεκμηρίωση στο χρήστη.
Team support	Ορίζει ένα μοντέλο προγραμματισμού ομάδας, για τη διαχείριση και την προσθήκη αριθμών έκδοσης στους πόρους.
Debug support	Ορίζει ένα μοντέλο αποσφαλμάτωσης και κλάσεις διαπροσωπείας χρήστη για το χτίσιμο αποσφαλματωτών. Το μοντέλο και οι κλάσεις αυτές είναι ανεξάρτητες από τη γλώσσα προγραμματισμού.
Other utilities	Άλλα plug-ins παρέχουν διάφορες άλλες χρήσιμες λειτουργίες όπως αναζήτηση και σύγκριση πόρων, μεταγλώττιση με χρήση αρχείων ρυθμίσεων σε γλώσσα XML, και δυναμική ανανέωση της πλατφόρμας από έναν server.

Πίνακας 3.1: Περιγραφή των ψηφίδων της πλατφόρμας

3.1.2 Επέκταση της πλατφόρμας με plug-ins

Όπως αναφέρθηκε και παραπάνω, η πλατφόρμα eclipse είναι δομημένη ως εξής: στον πυρήνα υπάρχει μια μηχανή χρόνου εκτέλεσης (core runtime engine) και πάνω της υπάρχει ένα σύνολο επιπρόσθετων λειτουργιών που εγκαθίστανται σαν plug-ins. Τα plug-ins συνεισφέρουν λειτουργικότητα στην πλατφόρμα συνεισφέροντας σε προκαθορισμένα σημεία επέκτασης. Ένα παράδειγμα ενός τέτοιου plug-in είναι και ο πάγκος εργασίας. Η γλώσσα στην οποία είναι υλοποιημένο το Eclipse είναι η Java. Όταν όμως εκκινείται ο πάγκος εργασίας, δεν εκκινείται ένα απλό πρόγραμμα Java. Ενεργοποιείται η μηχανή εκτέλεσης, η οποία μπορεί δυναμικά να ανακαλύπτει εγκατεστημένα plug-ins και να τα εκκινεί όποτε χρειάζεται.

Από τα παραπάνω προκύπτει ότι όταν κάποιος θέλει να γράψει κώδικα ο οποίος επεκτείνει την πλατφόρμα, η διαδικασία που πρέπει να ακολουθήσει είναι διαφορετική από τη διαδικασία ανάπτυξης μιας απλής εφαρμογής Java. Συγκεκριμένα, θα πρέπει να ορίσει επεκτάσεις μέσα στο plug-in του, οι οποίες να συνδέονται σε κάποια από τα σημεία επεκτάσεων που έχει ορίσει η πλατφόρμα. Από τη σκοπιά της πλατφόρμας, το plug-in που γράφει κάποιος τρίτος δε διαφέρει από τα βασικά plug-ins σαν αυτό που χειρίζεται τους πόρους (resources) ή τον ίδιο τον πάγκο εργασίας.

Τα βήματα που πρέπει να ακολουθήσουμε είναι:

- Απόφαση για τον τρόπο με τον οποίο το plug-in θα ενσωματωθεί στην πλατφόρμα.
- Εύρεση των κατάλληλων σημείων επέκτασης στα οποία χρειάζεται να συνεισφέρουμε ώστε να ενσωματωθεί το plug-in.
- Υλοποίηση αυτών των επεκτάσεων σύμφωνα με την προδιαγραφή για τα σημεία επέκτασης.
- Δημιουργία ενός αρχείου manifest (MANIFEST.MF) το οποίο περιγράφει το πακετάρισμα και τα προαπαιτούμενα για τον κώδικά μας, και ενός plug-in manifest (plugin.xml) το οποίο περιγράφει τις επεκτάσεις που ορίζουμε.

Ένα ελάχιστο plug-in

Το πιο απλό πρόγραμμα γραμμένο σε γλώσσα Java χωρίς τη χρήση άλλων βιβλιοθηκών διαπροσωπείας χρήστη μοιάζει με τον κώδικα 3.1.

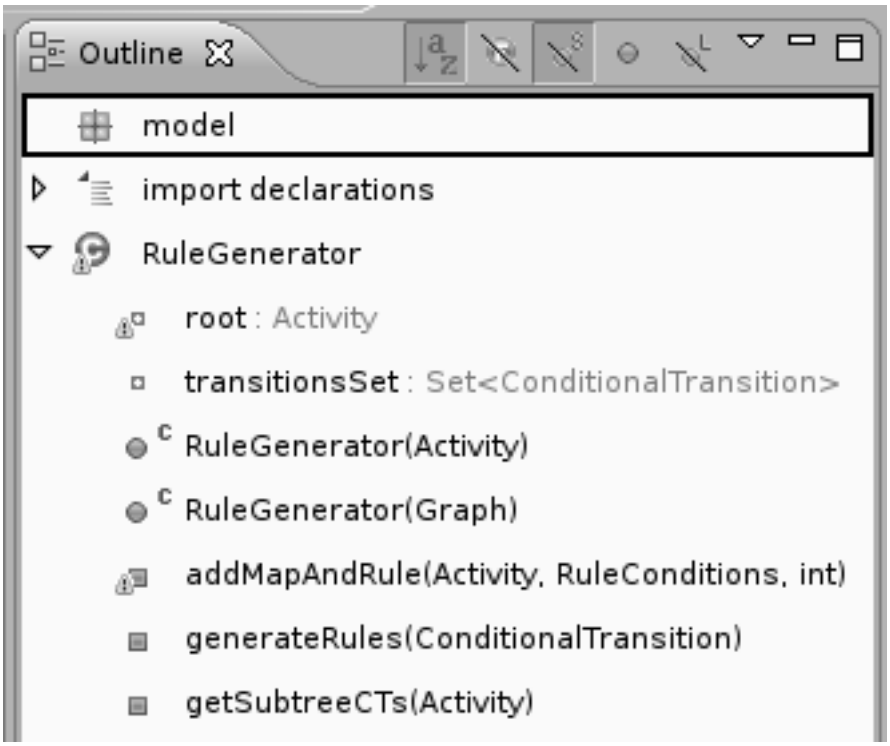
Πηγαίος Κώδικας 3.1: Το πρόγραμμα Hello World σε απλή Java

```
public class HelloWorld {
        public static void main(String[] args) {
        System.out.println("Hello World");
        }
}
```

Στο πλαίσιο της πλατφόρμας Eclipse, αυτό το πρότυπο μεταβάλλεται. Αντί να το σκεφτόμαστε ως ένα ανεξάρτητο πρόγραμμα, το αντιμετωπίζουμε ως μια επέκταση της πλατφόρμας.
Γενικά, υπάρχουν αρκετοί διαφορετικοί τρόποι με τους οποίους μπορεί να επεκτείνει κανείς το περιβάλλον χρήστη του πάγκου εργασίας. Ο πάγκος εργασίας είναι ένα παράθυρο που αποτελείται από διάφορα οπτικά μέρη (αντικείμενα). Αυτά τα μέρη χωρίζονται σε δύο βασικές κατηγορίες: τις όψεις (views) και τους επεξεργαστές (editors).

Οι επεξεργαστές επιτρέπουν στο χρήστη να επεξεργάζεται κάτι μέσα στον πάγκο εργασίας. Οι επεξεργαστές είναι "εγγραφο–κεντρικοί" και μοιάζουν σε πολλά με τους γνωστούς επεξεργαστές αρχείων (π.χ. αρχείων κειμένου). Τα έγγραφα ακολουθούν έναν κύκλο ζωής άνοιγμα – αποθήκευση – κλείσιμο. Οι επεξεργαστές είναι στενά δεμένοι και ενσωματωμένοι στον πάγκο εργασίας. Ο πάγκος εργασίας προσφέρει έναν έτοιμο editor για αρχεία απλού κειμένου. Παρέχει επίσης μια κοινή διαπροσωπεία με το χρήστη για τις συνηθισμένες εργασίες (π.χ. κουμπιά για άνοιγμα, αποθήκευση, αποθήκευση ως, κλπ.), αλλά ο τρόπος με τον οποίο γίνονται αυτές οι εργασίες υλοποιείται ξεχωριστά από κάθε editor.

Οι όψεις παρέχουν πληροφορίες σχετικά με κάποιο αντικείμενο με το οποίο εργάζεται ο χρήστης στον πάγκο εργασίας. Συχνά οι όψεις αλλάζουν τα περιεχόμενά τους όταν ο χρήστης επιλέγει διαφορετικά αντικείμενα στον πάγκο εργασίας. Πολλές φορές οι όψεις δρουν συμπληρωματικά με τους επεξεργαστές, παρέχοντας πληροφορίες σχετικά με το περιεχόμενο τους. Είναι δυνατό μία όψη να είναι αρκετά γενική ώστε να μπορεί να χρησιμοποιείται σε πολλές διαφορετικές περιστάσεις με διαφορετικούς τρόπους. Για παράδειγμα, η όψη ιδιοτήτων (Properties View) δείχνει πληροφορίες όπως πλήρες όνομα, κατάλογος, μέγεθος, ημερομηνία τελευταίας τροποποίησης, κτλ. όταν ο χρήστης επιλέξει κάποιο αρχείο μέσα στην όψη Navigator (βλ. σχήμα 3.2), αλλά μέσα σε έναν επεξεργαστή αρχείων XML μπορεί να δείχνει ιδιότητες όπως όνομα, τύπο, κτλ. του επιλεγμένου στοιχείο ή ιδιότητας XML. Ομοίως, η όψη περίληψης (Outline View, σχήμα 3.3), όταν ο χρήστης κάνει επεξεργασία πηγαίου κώδικα δείχνει μια σύνοψη του ενεργού αρχείου που αποτελείται από δηλώσεις, ονόματα και τύπους μεταβλητών και μεθόδων, η εμβέλεια των οποίων επισημαίνεται με διαφορετικά χρώματα/σχήματα, αλλά μπορεί να χρησιμοποιηθεί και αλλιώς.



Σχήμα 3.3: Όψη Outline για ένα αρχείο πηγαίου κώδικα Java

Στην εφαρμογή που αναπτύξαμε, χρησιμοποιήθηκαν και οι δύο παραπάνω τρόποι επέκτασης της πλατφόρμας. Βασικά, η εφαρμογή είναι ένας *γραφικός* επεξεργαστής, και τα έγγραφα τα οποία επεξεργάζεται είναι διαγράμματα όπως αυτά που περιγράφτηκαν στην ενότητα §2.3 (π.χ. σχήμα 2.15). Όμως προκαταβολικά μπορούμε να πούμε ότι χρησιμοποιήθηκαν και οι όψεις Properties, για εμφάνιση και επεξεργασία στοιχείων όπως π.χ. το όνομα της επιλεγμένης δραστηριότητας, Outline, για εμφάνιση της δενδρικής αναπαράστασης του διαγράμματος (σαν αυτή του σχήματος 2.16), όπως και μια άλλη όψη σμίκρυνσης.

Η διαδικασία επέκτασης της πλατφόρμας περιγράφεται στο υπόλοιπο αυτής της ενότητας. Επειδή η διαδικασία είναι πιο απλή για τις όψεις, παρά για τους επεξεργαστές, χρησιμοποιούμε το παράδειγμα μιας απλής όψης με ονομα Hello World.

Δημιουργία του project για το plug-in

Για να δημιουργήσουμε ένα plug-in για την πλατφόρμα Eclipse μπορούμε να χρησιμοποιήσουμε οποιοδήποτε Java IDE, αλλά το ίδιο το Eclipse SDK προσφέρει επιπλέον εργαλεία για την ανάπτυξη plug-ins. Συγκεκριμένα, υπάρχουν οι ευκολίες γεννήτορα κώδικα του Plug-in Development Environment (PDE) που μας δίνουν ένα template από το οποίο μπορούμε να ξεκινήσουμε. Το PDE αρχικοποιεί ένα project στο οποίο μπορούμε να γράψουμε κώδικα Java και δημιουργεί τα προκαθορισμένα αρχεία manifest του plug-in (τα οποία θα εξηγηθούν αναλυτικότερα παρακάτω) καθώς και μια κλάση η οποία υλοποιεί την όψη μας.

Η δημιουργία ενός νέου project για ένα plug-in είναι μια διαδικασία που μπορεί να γίνει μέσα στο Eclipse SDK με τη βοήθεια ενός wizard (βλ. σχήμα 3.4) . Η διαδικασία είναι απλή και δεν αξίζει να αναφερθεί πιο αναλυτικά εδώ. Το αποτέλεσμά της είναι ένα project δομημένο ώστε να περιέχει αρκετούς φακέλους, αρχεία και ένα πακέτο Java. Τα σημαντικά αρχεία σε αυτό το στάδιο είναι το plugin.xml και το MANIFEST.MF (αρχεία manifest) καθώς και ο κώδικας Java για το plug-in.

Σχήμα 3.4: Δημιουργία ενός νέου plug-in με τη βοήθεια του PDE

Η όψη Hello World

Για να μετατρέψουμε το παραπάνω πρόγραμμα Hello World σε plug-in του Eclipse μπορούμε να υλοποιήσουμε μια νέα όψη (view) η οποία να εμφανίζει στο χρήστη το μήνυμα "Hello World".

Το plug-in org.eclipse.ui.workbench ορίζει τις περισσότερες από τις δημόσιες διεπαφές (interfaces) που απαρτίζουν το API του πάγκου εργασίας. Αυτές οι διεπαφές βρίσκονται στο πακέτο org.eclipse.ui και στα υποπακέτα του. Πολλές από αυτές τις διεπαφές έχουν προκαθορισμένες κλάσεις υλοποίησης, τις οποίες μπορούμε να επεκτείνουμε για να παρέχουμε απλές τροποποιήσεις στο σύστημα. Στο παράδειγμα Hello World, θα επεκτείνουμε μια όψη του πάγκου εργασίας για να παρέχουμε μια ταμπέλα που λέει hello.

Η διαπροσωπεία που μας ενδιαφέρει είναι η IviewPart, η οποία ορίζει τις μεθόδους που πρέπει να

υλοποιούνται για να συνεισφέρουμε μια όψη στον πάγκο εργασίας. Η κλάση ViewPart παρέχει μια προκαθορισμένη υλοποίηση αυτής της διαπροσωπείας. Με λίγα λόγια, ένα αντικείμενο ViewPart είναι υπεύθυνο για τη δημιουργία των οπτικών αντικειμένων που χρειάζονται για να εμφανίζεται η όψη.

Η κλάση HelloWorldView

Ο κώδικας 3.2 περιέχει την υλοποίηση σε γλώσσα Java της όψης Hello World.

Πηγαίος Κώδικας 3.2: Η όψη Hello World

```
package com.example.helloworld;

import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.SWT;
import org.eclipse.ui.part.ViewPart;

public class HelloWorldView extends ViewPart {
    Label label;
    public HelloWorldView() {
    }
    public void createPartControl(Composite parent) {
        label = new Label(parent, SWT.WRAP);
        label.setText("Hello World");
    }
    public void setFocus() {
    }
}
```

Η όψη αυτή δημιουργεί τα οπτικά αντικείμενα που την αντιπροσωπεύουν μέσα στην createPartControl(). Σε αυτό το παράδειγμα, δημιουργήσαμε μια ετικέτα SWT [1] και θέσαμε το κείμενο “Hello World” μέσα σε αυτήν. Η παραπάνω κλάση είναι ίσως η απλούστερη όψη που θα μπορούσε κανείς να δημιουργήσει.

Αρχεία manifest

Όπως αναφέραμε, το PDE αναλαμβάνει και την αρχικοποίηση δύο αρχείων manifest που περιγράφουν κάθε plug-in που δημιουργεί.
Το plugin.xml (κώδικας 3.3) συμπεριλαμβάνει όλες τις πληροφορίες σχετικά με την επέκτασή μας και το πώς αυτή εκτελείται μέσα σε μια συσκευασία XML. Στη συγκεκριμένη περίπτωση δηλώνεται, μέσα στο στοιχείο extension, ότι το νέο plug-in επεκτείνει την πλατφόρμα στο σημείο επέκτασης

[1] Το SWT (Standard Widget Toolkit) είναι μια εργαλειοθήκη γραφικών η οποία μπορεί να χρησιμοποιηθεί με την πλατφόρμα Java. Αναπτύχθηκε αρχικά από την IBM και τώρα συντηρείται από το ίδρυμα Eclipse, όπως και το Eclipse IDE. Είναι μια εναλλακτική λύση των εργαλειοθηκών γραφικής διαπροσωπείας χρήστη AWT και Swing, τις οποίες παρέχει η Sun Microsystems μαζί με την πλατφόρμα Java.

Το SWT είναι γραμμένο σε γλώσσα Java. Για να εμφανίζει τα γραφικά στοιχεία, χρησιμοποιεί διαφορετικές, τοπικές (native) γραφικές βιβλιοθήκες για κάθε λειτουργικό σύστημα (π.χ. Motif ή GTK+ για λειτουργικά συστήματα της οικογένειας Unix, Win32 για Microsoft Windows, Carbon ή Cocoa για Mac OS), με αποτέλεσμα ”η εμφάνιση και αίσθηση” μιας γραφικής εφαρμογής που βασίζεται στο SWT να ταιριάζει με τις υπόλοιπες εφαρμογές του λειτουργικού συστήματος. Αυτό έρχεται σε αντίθεση με τις εφαρμογές γραμμένες σε AWT και Swing, οι οποίες έχουν μια ενιαία εμφάνιση σε όλα τα λειτουργικά συστήματα.

Η πλατφόρμα Eclipse και όλα τα plug-ins που είναι γραμμένα για αυτήν χρησιμοποιούν το SWT ως εργαλειοθήκη γραφικής διαπροσωπείας χρήστη.

org.eclipse.ui.views. Το αναγνωριστικό (id) της επέκτασης, δηλαδή το όνομα με το οποίο η πλατφόρμα γνωρίζει το plug-in είναι το example.HelloWorldView και η κλάση Java που υλοποιεί την επέκταση είναι η com.example.helloworld.HelloWorldView. Το plugin ορίζει επίσης και μια νέα κατηγορία όψεων, τη Hello Category.

Πηγαίος Κώδικας 3.3: plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin>
  <extension point="org.eclipse.ui.views">
    <category
        name="Hello Category"
        id="example.helloworld">
    </category>
    <view
        name="Hello View"
        icon="icons/sample.gif"
        category="com.example.helloworld"
        class="com.example.helloworld.HelloWorldView"
        id="example.HelloWorldView">
    </view>
  </extension>
</plugin>
```

Το άλλο αρχείο που δημιουργείται αυτομάτα είναι το OSGi manifest, το MANIFEST.MF (κώδικας 3.4). Το αρχείο αυτό περιγράφει πληροφορίες χαμηλότερου επιπέδου σχετικά με το πακετάρισμα του plug-in, χρησιμοποιώντας ορολογία της τεχνολογίας OSGi. Περιέχει πληροφορίες όπως το όνομα του plug-in (bundle), τα bundles τα οποία απαιτεί να υπάρχουν, την ελάχιστη έκδοση του eclipse, κλπ. [2]

Πηγαίος Κώδικας 3.4: MANIFEST.MF

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Hello World Plug-in
Bundle-SymbolicName: com.example.helloworld; singleton:=true
Bundle-Version: 1.0.0
```

[2] Η ανάπτυξη της τεχνολογίας OSGi ξεκίνησε το 1998 και προσπαθεί να λύσει το πρόβλημα του πώς να κατασκευάζονται εφαρμογές οι οποίες αποτελούνται από ανεξάρτητες ψηφίδες (components). Σήμερα η τεχνολογία προσφέρει ένα ώριμο σύστημα ψηφίδων το οποίο λειτουργεί σε ένα μεγάλο αριθμό διαφορετικών περιβαλλόντων. Είναι ένα σύστημα το οποίο χρησιμοποιείται για την κατασκευή ιδιαίτερα περίπλοκων εφαρμογών όπως το Eclipse IDE, εφαρμογών server (όπως οι GlassFish, IBM Websphere, Oracle/BEA Weblogic, Jonas, JBoss), πλαισίων εφαρμογών (application frameworks, όπως το Spring και το Guice), βιομηχανικών αυτοματισμών, τηλεφώνων και πολλών άλλων.

Η τεχνολογία OSGi είναι ένα σύνολο *προδιαγραφών* που ορίζουν ένα σύστημα δυναμικών ψηφίδων (components) για τη γλώσσα Java. Όπως προαναφέρθηκε, με αυτές τις προδιαγραφές γίνεται δυνατό ένα μοντέλο ανάπτυξης όπου οι εφαρμογές συντίθενται δυναμικά από πολλά διαφορετικές ψηφίδες. Οι προδιαγραφές OSGi επιτρέπουν στις ψηφίδες να κρύβουν τις υλοποιήσεις τους από τις άλλες ψηφίδες, ενώ επικοινωνούν μέσω *υπηρεσιών* (services), οι οποίες είναι αντικείμενα διαμοιρασμένα μεταξύ των ψηφίδων. Αυτό το απλό μοντέλο έχει σημαντικές επιπτώσεις για σχεδόν όλες τις πλευρές της διεργασίας ανάπτυξης του λογισμικού.

Η τεχνολογία OSGi ακολουθεί ένα διαστρωματωμένο μοντέλο στη βάση του οποίου είναι κατ' αρχήν το λειτουργικό σύστημα και η εικονική μηχανή Java, ενώ πάνω σε αυτά βρίσκονται διαδοχικά το περιβάλλον εκτέλεσης, τα αρθρώματα (modules), ο κύκλος ζωής (life cycle), οι υπηρεσίες (services) και στο ανώτερο στρώμα τα bundles.

Τα *bundles* είναι οι ψηφίδες του OSGi οι οποίες κατασκευάζονται από τους δημιουργούς των τελικών εφαρμογών. Ο μηχανισμός του eclipse που υποστηρίζει τη δημιουργία των plug-ins είναι το OSGi. Από αυτή τη σκοπιά, ένα plug-in είναι το ίδιο πράγμα με ένα OSGi bundle, και το αρχείο MANIFEST.MF περιγράφει τη λειτουργία του.

Require-Bundle: org.eclipse.ui,
org.eclipse.ui.views

Και τα δύο αρχεία manifest μπορούμε να τα επεξεργαζόμαστε είτε απευθείας είτε μέσω των γραφικών ευκολιών που προσφέρει το PDE και συγκεκριμένα ο plug-in editor.

Εκτέλεση του plug-in

Μετά τη μεταγλώττιση της νέας όψης, υπάρχουν δύο διαφορετικοί πιθανοί τρόποι, με τους οποίους μπορεί να εκτελεστεί.

- Τοποθετούμε τα αρχεία manifest και το αρχείο jar στον κατάλογο eclipse/plugins. Όταν ο πάγκος εργασίας επανεκκινηθεί, θα βρει το νέο plug-in.
- Το εργαλείο PDE μπορεί να χρησιμοποιηθεί για να εκτελεστεί ένας άλλος πάγκος εργασίας μέσα από τον τρέχοντα πάγκο. Αυτή η μέθοδος χρησιμοποιείται κατά τη διάρκεια της ανάπτυξης του plug-in και είναι πιο βολική για τον προγραμματιστή γιατί μπορεί να δοκιμάζει τα νέα plug-ins που δημιουργεί άμεσα.

Όπως και αν επιλέξουμε να εκτελέσουμε το plug-in Hello World, η όψη θα μπορεί να εμφανιστεί από το μενού Window > Show View > Hello Category > Hello View του πάγκου εργασίας του eclipse (δεδομένου ότι το plug-in παρέχει μια επέκταση για το σημείο επέκτασης org.eclipse.views, όπως ορίστηκε στο plugin.xml).
Μέχρι τη στιγμή που θα επιλέξει ο χρήστης να εμφανίσει την όψη Hello View, ο κώδικας του plug-in δεν έχει εκτελεστεί. Η δήλωση μέσα στο plugin.xml είναι αρκετή για να γνωρίζει ο πάγκος εργασίας ότι υπάρχει μια όψη με αυτό το όνομα στην κατηγορία Hello Category καθώς και την κλάση η οποία υλοποιεί την όψη. Τη στιγμή που θα επιλέξει ο χρήστης να δείξει την όψη ο πάγκος εργασίας ενεργοποιεί το plug-in, αρχικοποιεί την κλάση HelloWorldView και τώρα ο κώδικας του plug-in εκτελείται.

3.1.3 Πλατφόρμα Rich Client (Rich Client Platform)

Γενικά

Ενώ η πλατφόρμα Eclipse είναι σχεδιασμένη να λειτουργεί ως μια πλατφόρμα ανοιχτών εργαλείων, η αρχιτεκτονική της είναι τέτοια που οι ψηφίδες της θα μπορούσαν να χρησιμοποιηθούν για να κατασκευαστεί σχεδόν οποιαδήποτε εφαρμογή πελάτης.
Εφαρμογές διαφορετικές από IDEs μπορούν να κατασκευαστούν με χρήση ενός υποσυνόλου της πλατφόρμας. Αυτές οι "πλούσιες" εφαρμογές εξακολουθούν να βασίζονται σε ένα μοντέλο δυναμικών plug-ins και η διαπροσωπεία χρήστη κατασκευάζεται με τις ίδιες εργαλειοθήκες και τα ίδια σημεία επέκτασης. Η διάταξη και η λειτουργικότητα του πάγκου εργασίας βρίσκονται υπό τον έλεγχο της ομάδας ανάπτυξης του plug-in.
Η πλατφόρμα Rich Client [3] (RCP) είναι το ελάχιστο σύνολο των plug-ins που χρειάζονται για να

[3] Ο όρος *rich client* πρωτοεμφανίστηκε στις αρχές τις δεκαετίας του 1990, με τον ενθουσιασμό δημιουργίας εφαρμογών πελάτη χρησιμοποιώντας γλώσσες όπως η Visual Basic και η Delphi. Η δραματική αύξηση σε αριθμό και δημοφιλία αυτών των εφαρμογών οφειλόταν σε ένα βαθμό στην επιθυμία για μια "πλούσια" εμπειρία από την πλευρά του χρήστη.

Οι rich clients υποστηρίζουν μια υψηλής ποιότητας εμπειρία του χρήστη για έναν συγκεκριμένο τομέα, παρέχοντας πλούσιες διεπαφές χρήστη καθώς και γρήγορη επεξεργασία στον τοπικό υπολογιστή. Οι πλούσιες διεπαφές χρήστη παρέχουν λειτουργίες επιφάνειας εργασίας όπως drag-and-drop, clipboard, πλοήγηση και παραμετροποίηση. Όταν επιτυγχάνει το σκοπό της, μια εφαρμογή Rich Client είναι σχεδόν διαφανής και επιτρέπει στο χρήστη να εστιάζει στη δουλειά και όχι στο σύστημα. Ο όρος rich client δημιουργήθηκε για να διαφοροποιήσει τέτοιες εφαρμογές πελάτες από τις εφαρμογές τερματικών, ή *simple clients*, τις οποίες αντικατέστησαν.

Ο όρος έρχεται σε αντίθεση και με τον όρο *thin clients*, δηλαδή τις δικτυακές και ιστοκεντρικές εφαρμογές, οι οποίες εμφανίστηκαν λίγο αργότερα. Μια εφαρμογή thin client επικοινωνεί με μια εφαρμογή server η οποία συνήθως εκτελείται σε έναν άλλο υπολογιστή, και οι δύο εφαρμογές επικοινωνούν μέσω ενός δικτύου. Το κύριο κομμάτι της επεξεργασίας γίνεται στην εφαρμογή server, το κόστος μειώνεται, αλλά η εφαρμογή client δεν παρέχει την ίδια ταχύτητα ούτε τις ίδιες προχωρημένες λειτουργίες που θα περίμεναν οι χρήστες.

κατασκευαστεί μια εφαρμογή με διαπροσωπεία χρήστη βασισμένη στην πλατφόρμα. Μια εφαρμογή RCP απαιτεί κατ' αρχήν μόνο δύο plug-ins, τα org.eclipse.ui και org.eclipse.core.runtime, μαζί με τις εξαρτήσεις τους.
Παρόλα αυτά, μια εφαρμογή rich client μπορεί να χρησιμοποιεί επιπλέον οποιαδήποτε άλλη ψηφίδα που χρειάζεται για τη λειτουργικότητά της, και μπορεί να απαιτεί οποιοδήποτε άλλο plug-in εκτός από τα απολύτως απαραίτητα. Παραδείγματα είναι το σύστημα βοήθειας προς το χρήστη και ο διαχειριστής αναβαθμίσεων.

Ιστορία του Eclipse ως πλατφόρμα RCP

Το έργο Eclipse δεν ξεκίνησε με το σκοπό να δημιουργήσει μια πλατφόρμα RCP. Ο αρχικός στόχος ήταν να δημιουργηθεί μια πλατφόρμα για να ενσωματώνονται σε αυτήν εργαλεία ανάπτυξης εφαρμογών, δηλαδή ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE). Το Eclipse σαν RCP ξεκίνησε κατά τη διάρκεια της έκδοσης 2.1, όταν κάποιοι παρατήρησαν ότι τα IDEs που βασίζονταν πάνω στο Eclipse είχαν καλή λειτουργικότητα, εμφάνιση και επιδόσεις, και ότι το ίδιο πλαίσιο που θα μπορούσε να χρησιμοποιηθεί για να χτιστούν εφαρμογές για πιο γενική χρήση.
Η παρατήρηση ήταν ορθή, αλλά η υλοποίηση της ιδέας είχε πολλές πρακτικές δυσκολίες. Οι κυριότερες ήταν όλες οι υποθέσεις που είχαν γίνει σχετικά με το Eclipse ως μια πλατφόρμα δημιουργίας εργαλείων και η συνακόλουθη έλλειψη δυνατότητας αλλαγής κάποιων στοιχείων της εμφάνισης και της λειτουργικότητας της πλατφόρμας.
Η έκδοση 3.0 ήταν ένας σημαντικός σταθμός για το Eclipse ως μια πλατφόρμα RCP. Όλες οι αλληλεξαρτήσεις που σχετίζονταν με τη λειτουργικότητα ως IDE εξαλείφθηκαν και στα διάφορα μέρη της διαπροσωπείας με το χρήστη προστέθηκαν νέες δυνατότητες ρυθμίσεων. Εισήχθη το περιβάλλον χρόνου εκτέλεσης το βασισμένο στο OSGi και έτσι τέθηκαν οι βάσεις για δυναμική εγκατάσταση, διαγραφή και αναβάθμιση των plug-ins. Αυτές οι δύο μεγάλες αλλαγές σήμαιναν μια μεγάλη αναδιάρθρωση της πλατφόρμας.

Μετατροπή ενός απλού plug-in σε εφαρμογή RCP

Το εργαλείο PDE μπορεί εύκολα να χρησιμοποιηθεί για να δημιουργηθεί και μια εφαρμογή RCP, αφού περιέχει μια τέτοια επιλογή στον wizard για δημιουργία νέου plug-in. Οι αλλαγές που χρειάζεται να γίνουν είναι μικρές: στο αρχείο plugin.xml προστίθεται μια επέκταση στο σημείο επέκτασης org.eclipse.core.runtime.applications (κώδικας 3.5).

Πηγαίος Κώδικας 3.5: plugin.xml για εφαρμογή RCP

```
<plugin>
  <extension
      id="application"
      point="org.eclipse.core.runtime.applications">
    <application>
      <run
          class="merged.Application">
      </run>
    </application>
  </extension>
  <!-- ... -->
</plugin>
```

Μέσα στο στοιχείο run ορίζεται η κλάση η οποία αναλαμβάνει να ελέγχει όλες τις πλευρές της εκτέλεσης της εφαρμογής, δηλαδή την εκκίνηση, τον τερματισμό και πιθανώς την επανεκκίνησή της. Πρόκειται για μια απλή κλάση που υλοποιεί τη διαπροσωπεία IApplication, δηλαδή τις μεθόδους start() και stop() και μπορεί να μοιάζει με τον κώδικα 3.6.

Πηγαίος Κώδικας 3.6: Application.java

```
import org.eclipse.equinox.app.IApplication;
import org.eclipse.equinox.app.IApplicationContext;
import org.eclipse.swt.widgets.Display;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.PlatformUI;

public class Application implements IApplication {
    public Object start(IApplicationContext context)
        throws Exception {
        Display display = PlatformUI.createDisplay();
        try {
            int returnCode = PlatformUI.createAndRunWorkbench
        (display, new ApplicationWorkbenchAdvisor());
            if (returnCode == PlatformUI.RETURN_RESTART)
                return IApplication.EXIT_RESTART;
            else
                return IApplication.EXIT_OK;
        } finally {
            display.dispose();
        }
    }

    public void stop() { // ...
}
```

Η κλάση Application δημιουργεί και εκτελεί τον πάγκο εργασίας. Τώρα πια το αρχικό plug-in έχει μετατραπεί σε μια πλήρη εφαρμογή που μπορεί να στέκεται μόνη της.
Στην περίπτωση του Simple Sequencing Editor, ο λόγος που οδήγησε στην απόφαση να στηθεί ως εφαρμογή RCP και όχι ως ένα απλό plug-in για την πλατφόρμα είναι κυρίως το γεγονός ότι η εφαρμογή είναι σχετικά μικρή και δε χρειάζεται όλες τις λειτουργίες που παρέχει η μια συνηθισμένη εγκατάσταση του eclipse. Με την αρχιτεκτονική RCP η γραφική διαπροσωπεία του χρήστη απλουστεύεται, καθώς παρουσιάζονται στο χρήστη μόνο όσες λειτουργίες σχετίζονται με την ανάπτυξη εγγράφων του Simple Sequencing. Ταυτόχρονα, επειδή ο αριθμός των plug-ins–εξαρτήσεων που είναι απαραίτητα για τη λειτουργία της εφαρμογής είναι κατά πολύ περιορισμένος σε σχέση με τον αριθμό των plug-ins που περιλαμβάνονται σε μια τυπική εγκατάσταση του eclipse, μειώνεται δραστικά (περίπου κατά πέντε φορές) ο απαιτούμενος για την εφαρμογή χώρος στο δίσκο.

Προϊόντα Eclipse

Ένα προϊόν eclipse είναι μια επέκταση στο σημείο επέκτασης της πλατφόρμας που λέγεται org.eclipse.core.runtime.products. Η μετατροπή μιας εφαρμογής RCP σε ένα προϊόν eclipse, παρουσιάζει κάποια πλεονεκτήματα για έναν προγραμματιστή.
Όταν διανέμεται η εφαρμογή στους τελικούς χρήστες, πρέπει έχουν τοποθετηθεί σε ένα πακέτο όλα τα plug-ins-εξαρτήσεις που χρειάζονται για να εκτελεστεί αυτή, και να διανέμονται και αυτά μαζί με τον αντικειμενικό κώδικα του "κυρίως" plug-in. Σε πρώτο επίπεδο, η πληροφορία για τις άμεσες εξαρτήσεις του plug-in υπάρχει στο αρχείο MANIFEST.MF, αλλά καθένα από τα plug-ins-εξαρτήσεις έχει πιθανόν και δικές του εξαρτήσεις που απαιτεί το ίδιο για να λειτουργήσει, με αποτέλεσμα τη δημιουργία ενός *δέντρου εξαρτήσεων*. Για ένα προϊόν eclipse, ο υπολογισμός όλων αυτών των εξαρτήσεων μπορεί να γίνεται αυτόματα από το PDE, και στη συνέχεια υποστηρίζεται και η εξαγωγή της εφαρμογής σε μορφή έτοιμη να εκτελεστεί για διάφορα λειτουργικά συστήματα (linux, macosx,

solaris, win32).
Επίσης, χρησιμοποιώντας ένα προϊόν μπορεί κανείς να ορίσει και να παραμετροποιήσει την εμφάνιση του Eclipse για μια συγκεκριμένη εφαρμογή.
Στον κώδικα 3.7 φαίνεται ότι με αυτόν τον τρόπο μπορεί για παράδειγμα να οριστεί μια σύντομη περιγραφή (aboutInfo) με εικόνα και μικρό κείμενο της εφαρμογής. Υπάρχουν και άλλα στοιχεία εμφάνισης που ρυθμίζονται εδώ, όπως τα εικονίδια που θα εμφανίζονται στα διάφορα λειτουργικά συστήματα, ή η εμφάνιση μιας αρχικής οθόνης (splash) την ώρα που εκκινείται η εφαρμογή, τα οποία έχουν παραλειφθεί για συντομία. Τέλος, φαίνεται η λίστα όλων των plug-ins που απαιτούνται για τη λειτουργία της εφαρμογής.
Η δημιουργία και η επεξεργασία του αρχείου .product είναι απλές διαδικασίες και γίνονται μέσα από το περιβάλλον ανάπτυξης PDE.

Πηγαίος Κώδικας 3.7: RCPSS.product

```
<?xml version="1.0" encoding="UTF-8"?>
<?pde version="3.4"?>
<product name="RCP Simple Sequencing Editor"
         id="merged.product"
         application="merged.application"
         useFeatures="false">
  <aboutInfo>
    <image path="/merged/images/about.png"/>
    <text>
      RCP Simple Sequencing Editor v0.1.0
      A Graphical Editor that creates Simple Sequencing Documents
      by Christos Gitsis, Andreas Papasalouros
      (c) Copyright authors
    </text>
  </aboutInfo>
  <configIni use="default">
  </configIni>
  <launcherArgs>
    <programArgs>-console -noExit</programArgs>
    <vmArgsMac>-XstartOnFirstThread
      -Dorg.eclipse.swt.internal.carbon.smallFonts
    </vmArgsMac>
  </launcherArgs>

  <!-- ... -->

  <plugins>
    <plugin id="com.ibm.icu"/>
    <plugin id="javax.servlet"/>
    <plugin id="org.apache.ant"/>
    <plugin id="org.eclipse.core.commands"/>
    <plugin id="org.eclipse.core.filesystem"/>
    <plugin id="org.eclipse.core.filesystem.linux.x86_64"
        fragment="true"/>
    <!-- ... σύνολο 59 plug-ins-->
  </plugins>
</product>
```

3.2 Το πλαίσιο Graphical Editing Framework

Το πλαίσιο Graphical Editing Framework (GEF) παρέχει το υπόβαθρο ώστε μια ομάδα ανάπτυξης λογισμικού να μπορεί να δημιουργήσει έναν πλούσιο, διαδραστικό γραφικό επεξεργαστή (editor) από ένα ήδη υπάρχον μοντέλο. Αρχιτεκτονικά, το GEF αποτελείται από δύο plug-ins:

Draw2d (org.eclipse.draw2d) Μια ελαφριά εργαλειοθήκη γραφικών για σχεδίαση και διάταξη οπτικών αντικειμένων πάνω σε έναν καμβά SWT.

GEF (org.eclipse.gef) ένα διαδραστικό πλαίσιο αρχιτεκτονικής MVC, χτισμένο πάνω στο Draw2d.

Ο όρος "GEF" είναι πιθανό να αναφέρεται σε ολόκληρη την ψηφίδα (2 plug-ins) ή μόνο στο plug-in org. eclipse .gef.

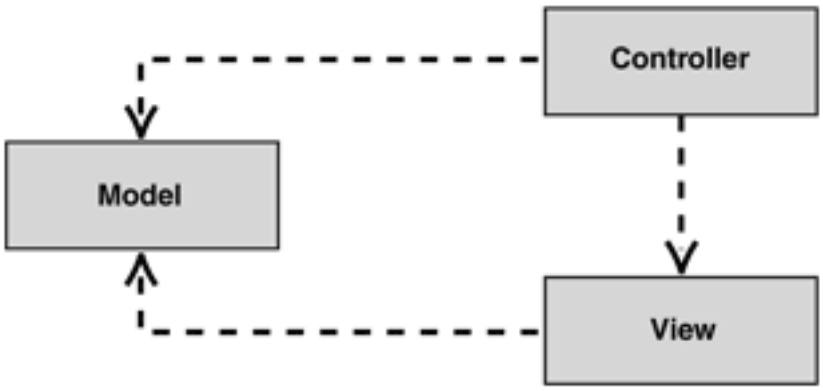
3.2.1 Αρχιτεκτονική MVC

Το αρχιτεκτονικό μόρφημα Model-View-Controller (MVC) [10, 11] διαμερίζει μια διαδραστική εφαρμογή σε τρεις συνιστώσες. Το μοντέλο (model) περιέχει τη βασική λειτουργικότητα και τα δεδομένα. Οι όψεις (views) και οι ελεγκτές (controllers) από κοινού συνιστούν τη διαπροσωπεία χρήστη. Ένας μηχανισμός αλλαγής-ενημέρωσης εξασφαλίζει τη συνέπεια μεταξύ της διαπροσωπείας χρήστη και του μοντέλου. Αναλυτικότερα:

Το μοντέλο είναι η ειδική για το πεδίο με το οποίο ασχολείται η εφαρμογή αναπαράσταση της πληροφορίας, μαζί με τους κανόνες που καθορίζουν τους τρόπους πρόσβασης σε αυτήν και τους τρόπους μεταβολής της. Συνήθως το μοντέλο είναι προσέγγιση μιας διεργασίας από τον πραγματικό κόσμο.

Η όψη μετατρέπει το μοντέλο σε μια μορφή κατάλληλη για διάδραση, συνήθως σε ένα στοιχείο διαπροσωπείας χρήστη. Η όψη έχει την ευθύνη να διατηρεί σε κάθε στιγμή μια συνεπή αναπαράσταση του μοντέλου, όταν αυτό μεταβάλλεται. Είναι δυνατό να υπάρχουν πολλαπλές όψεις για το ίδιο μοντέλο, οι οποίες να εξυπηρετούν διαφορετικούς σκοπούς.

Ο ελεγκτής είναι η συνιστώσα που αντιδρά σε συμβάντα (συνήθως ενέργειες του χρήστη) πάνω στην όψη, τα επεξεργάζεται και τα μεταφράζει σε ενέργειες που θα εκτελεστούν από το μοντέλο.



Σχήμα 3.5: Αρχιτεκτονικό μόρφημα MVC

Είναι σημαντικό να σημειωθεί ότι τόσο ο ελεγκτής όσο και η όψη έχουν εξάρτηση από το μοντέλο. Αντίθετα, το μοντέλο δεν εξαρτάται από καμία από τις άλλες δύο συνιστώσες, και αυτό είναι ένα από τα σημαντικότερα πλεονεκτήματα της αποσύζευξης των συνιστωσών. Ο διαχωρισμός αυτός επιτρέπει την κατασκευή και τον έλεγχο του μοντέλου ανεξάρτητα από την οπτική αναπαράστασή του.

3.2.2 Draw2d

Γενικά

Το Draw2d είναι μια "ελαφριά" εργαλειοθήκη γραφικών ψηφίδων, οι οποίες ονομάζονται *σχήματα* (figures). Ο όρος ελαφριά σημαίνει ότι ένα σχήμα είναι απλώς ένα αντικείμενο Java, χωρίς κανέναν πόρο δεσμευμένο στο λειτουργικό σύστημα που να αντιστοιχεί σε αυτό.
Τα σχήματα μπορούν να συντεθούν μέσω μιας σχέσης παιδιού-γονέα. Κάθε σχήμα έχει ορθογώνια *όρια* (bounds), μέσα στα οποία ζωγραφίζονται το ίδιο και τα παιδιά του. Ένας διαχειριστής διάταξης μπορεί να χρησιμοποιηθεί για να τοποθετήσει τα παιδιά ανάλογα με τον αύξοντα αριθμό τους ή/και με τον περιορισμό τους.
Ένα ελαφρύ σύστημα (LightweightSystem) συσχετίζει μια σύνθεση σχημάτων με έναν καμβά SWT. Το ελαφρύ σύστημα τοποθετεί ακροατές (listeners) για τα περισσότερα συμβάντα SWT, και προωθεί τα περισσότερα από αυτά σε έναν αποστολέα συμβάντων (EventDispatcher).
Αντίθετα, τα συμβάντα που σχετίζονται με τη σχεδίαση, προωθούνται στον διαχειριστή ανανεώσεων (UpdateManager) ο οποίος συντονίζει τη σχεδίαση και τη διάταξη των σχημάτων. Τα σχήματα μπορούν να ανανεωθούν με τρόπους που επηρεάζουν την εμφάνιση ή το μέγεθός τους, και ο διαχειριστής ανανεώσεων εξασφαλίζει ότι μόνο μια επαναδιάταξη συμβαίνει, ακολουθούμενη από επανασχεδίαση μόνο της περιοχής που ανανεώθηκε.
Κάθε τύπος διαγράμματος, εγγράφου ή σχεδίου μπορεί να κατασκευαστεί εύκολα και να ανανεώνεται αποδοτικά με συνδυασμένη χρήση των ήδη παρεχόμενων από το Draw2d υλοποιήσεων των σχημάτων και των διατάξεων, καθώς επίσης και με τη χρήση ειδικών νέων σχημάτων ή διατάξεων όποτε χρειάζεται.

Ζωγραφική

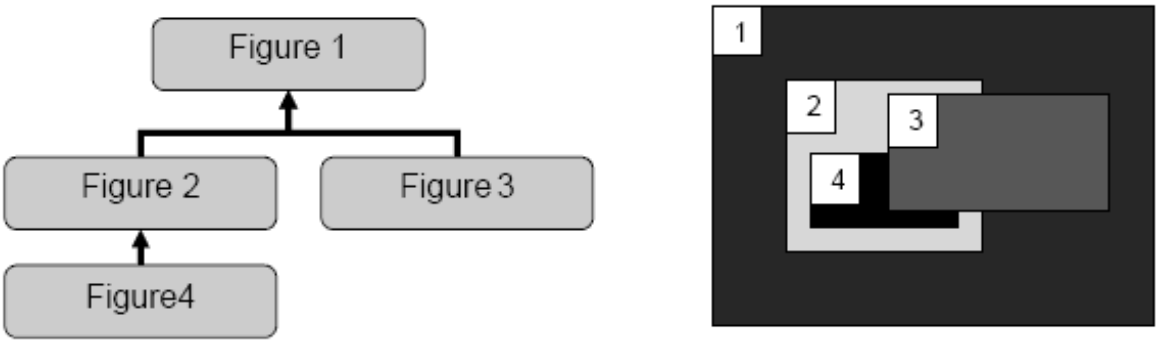
Συνοπτικά, η διαδικασία με την οποία ένα σχήμα ζωγραφίζεται είναι η εξής:

Figure#paint() Αυτή η μέθοδος ξεκινά τη διαδικασία σχεδίασης, καθορίζοντας κάποιες ιδιότητες όπως η γραμματοσειρά και το χρώμα σχεδίασης του υπόβαθρου και του πρώτου πλάνου, και στη συνέχεια καλούνται οι εξής μέθοδοι:

- **Figure#paintFigure()** Το σχήμα ζωγραφίζει τον εαυτό του. Μια απλή μέθοδος ζωγραφικής θα ήταν να γεμίζει την περιοχή μέσα στα όριά του με το χρώμα του υποβάθρου του.
- **Figure#paintClientArea()** Η περιοχή πελάτη είναι αυτή όπου πρόκειται (και επιτρέπεται) να εμφανίζονται τα παιδιά. Είναι πιθανό να γίνονται σε αυτήν τη μεθοδο μεταβολές στα γραφικά οι οποίες επηρεάζουν μόνο τα παιδιά, όπως αλλαγές που αφορούν το σύστημα συντεταγμένων.
  - **Figure#paintChildren()** Μετά την ετοιμασία της περιοχής πελάτη, ζωγραφίζονται τα παιδιά.
- **Figure#paintBorder()** Τέλος, το σχήμα ζωγραφίζει διακοσμήσεις που πιθανόν να χρειάζεται να εμφανίζονται πάνω από τα παιδιά, και, αν έχει ρυθμιστεί έτσι, ζωγραφίζει και τα σύνορά του.

Το αποτέλεσμα αυτής της διαδικασίας είναι μια σύνθεση σχημάτων η οποία είναι δομημένη ως ένα δέντρο και ζωγραφίζεται με σειρά preorder και depth-first, με εξαίρεση τα σύνορα, που ζωγραφίζονται με σειρά postorder (σχήμα 3.6[4]).

[4] Σχήματα αυτής και της επόμενης ενότητας: Copyright ©IBM Corporation and others 2007, άδεια Eclipse Public Licence v1.0, έγγραφο [12]



Σχήμα 3.6: Ένα δέντρο σχημάτων και η γραφική αναπαράστασή του αν κάθε σχήμα ζωγραφιστεί ως πλήρες ορθογώνιο παραλληλόγραμμο

Συστήματα συντεταγμένων

Το Draw2d χρησιμοποιεί δύο διαφορετικά συστήματα συντεταγμένων. Το προκαθορισμένο σύστημα συντεταγμένων ονομάζεται *απόλυτο* και είναι πολύ απλό αφού είναι το ίδιο για όλα τα σχήματα. Παράλληλα, υπάρχει και η δυνατότητα να χρησιμοποιείται *σχετικό* (ή *τοπικό*) σύστημα συντεταγμένων, στο οποίο τα όρια κάθε παιδιού θεωρούνται σχετικά με την περιοχή πελάτη (δηλαδή τα όρια) του γονέα τους.
Για περιστάσεις όπου χρειάζεται να βρεθεί ένα σχήμα για μια θέση του δείκτη του ποντικιού, το Draw2d παρέχει τις κατάλληλες μεθόδους όπως την findFigureAt(Point p).

Συνδέσεις

Τέλος, το Draw2d χρησιμοποιεί ειδικά σχήματα για να εμφανίζει γραμμές (*συνδέσεις*) μεταξύ δύο σημείων. Τα σχήματα αυτά υλοποιούν τη διαπροσωπεία Connection. Συνήθως οι συνδέσεις πρέπει να εμφανίζονται σε ένα στρώμα πάνω από τα άλλα σχήματα.
Το αρχικό και το τελικό σημείο της γραμμής ορίζονται με χρήση σημείων άγκυρας (ConnectionAnchor), και τα υπόλοιπα σημεία της γραμμής σχεδιάζονται από έναν δρομολογητή (ConnectionRouter). Σε αντίθεση με τα άλλα σχήματα, τα όρια των συνδέσεων δεν καθορίζονται από τους γονείς τους ή από άλλες εξωτερικές οντότητες, αλλά υπολογίζονται από τον δρομολογητή. Ο απλούστερος δρομολογητής ενώνει το αρχικό και το τελικό σημείο με ένα ευθύγραμμο τμήμα, ενώ ένας άλλος δρομολογητής μπορεί να επιλέγει διαφορετικό τρόπο, λόγου χάρη να ορίζει τεθλασμένες γραμμές, ή οτιδήποτε άλλο. Ως σχήματα, οι συνδέσεις μπορούν να έχουν και αυτές παιδιά. Τα παιδιά αυτά συνήθως είναι διακοσμήσεις, όπως για παράδειγμα ένα βέλος που δείχνει την κατεύθυνση της σύνδεσης ή μια ετικέτα κειμένου που την περιγράφει.

3.2.3 GEF

Γενικά

Το Draw2d εστιάζει στην αποδοτική διάταξη και σχεδίαση σχημάτων. Το plug-in GEF προσθέτει δυνατότητες επεξεργασίας πάνω στο Draw2d και ο σκοπός του είναι να:

1. Διευκολύνει την εμφάνιση οποιουδήποτε μοντέλου με γραφικό τρόπο χρησιμοποιώντας σχήματα του Draw2d.

2. Υποστηρίζει διαδραστικότητα μέσα από το ποντίκι, το πληκτρολόγιο ή τον πάγκο εργασίας.

3. Παρέχει έτοιμες κάποιες συχνά χρησιμοποιούμενες ψηφίδες που σχετίζονται με τα παραπάνω.

Το σχήμα 3.7 δείχνει μια όψη του GEF, το οποίο μπορεί να οριστεί περίπου ως η περιοχή που βρίσκεται στη μέση. Είναι αυτό που παρέχει τη σύνδεση ανάμεσα στο μοντέλο και στην όψη μιας εφαρμογής.

Επίσης παρέχει χειριστές, όπως τα εργαλεία (tools) και οι ενέργειες (actions), οι οποίοι μετατρέπουν συμβάντα (events) σε αιτήματα (requests). Τα αιτήματα και οι εντολές (commands) χρησιμοποιούνται για την ενθυλάκωση των διαδράσεων και των αποτελεσμάτων τους πάνω στο μοντέλο, αντίστοιχα.
Μια εφαρμογή βασισμένη στο GEF χρησιμοποιεί αρχιτεκτονική MVC. Ο τρόπος με τον οποίο οι ρόλοι της αρχιτεκτονικής MVC εφαρμόζονται εδώ είναι ο εξής: Ο ορισμός του *μοντέλου* δεν αλλάζει, είναι τα δεδομένα που αποθηκεύονται, και αυτό προκύπτει από το πεδίο της εφαρμογής. Οι *εντολές* είναι ο τρόπος με τον οποίο το μοντέλο μεταβάλλεται, κατά έναν τρόπο που να μπορεί να γίνεται αναίρεση και επανάληψη των αλλαγών από το χρήστη. Η *όψη* είναι οτιδήποτε ορατό στο χρήστη, και τα στοιχεία της είναι είτε σχήματα του Draw2d (Figures), είτε στοιχεία ενός δέντρου (TreeItems). Μπορεί μια εφαρμογή να χρησιμοποιεί όψεις και των δύο ειδών ταυτόχρονα. Ο *ελεγκτής* στο GEF ονομάζεται EditPart και δεν είναι ένας, αλλά υπάρχει συνήθως ένας ελεγκτής για κάθε στοιχείο του μοντέλου που έχει οπτική αναπαράσταση. Οι ελεγκτές είναι ο σύνδεσμος ανάμεσα στο μοντέλο και την όψη, και είναι υπεύθυνοι και για την επεξεργασία του μοντέλου. Τα EditParts περιέχουν βοηθητικά στοιχεία που ονομάζονται *EditPolicies*, τα οποία χειρίζονται το μεγαλύτερο μέρος της επεξεργασίας. Τα EditParts εμφανίζουν τις όψεις τους μέσα σε στοιχεία *EditPartViewer*. Υπάρχουν δύο τύποι τέτοιων στοιχείων, οι γραφικοί viewers, που εμφανίζουν Figures, και οι δενδρικοί viewers, που εμφανίζουν TreeItems.

EditParts

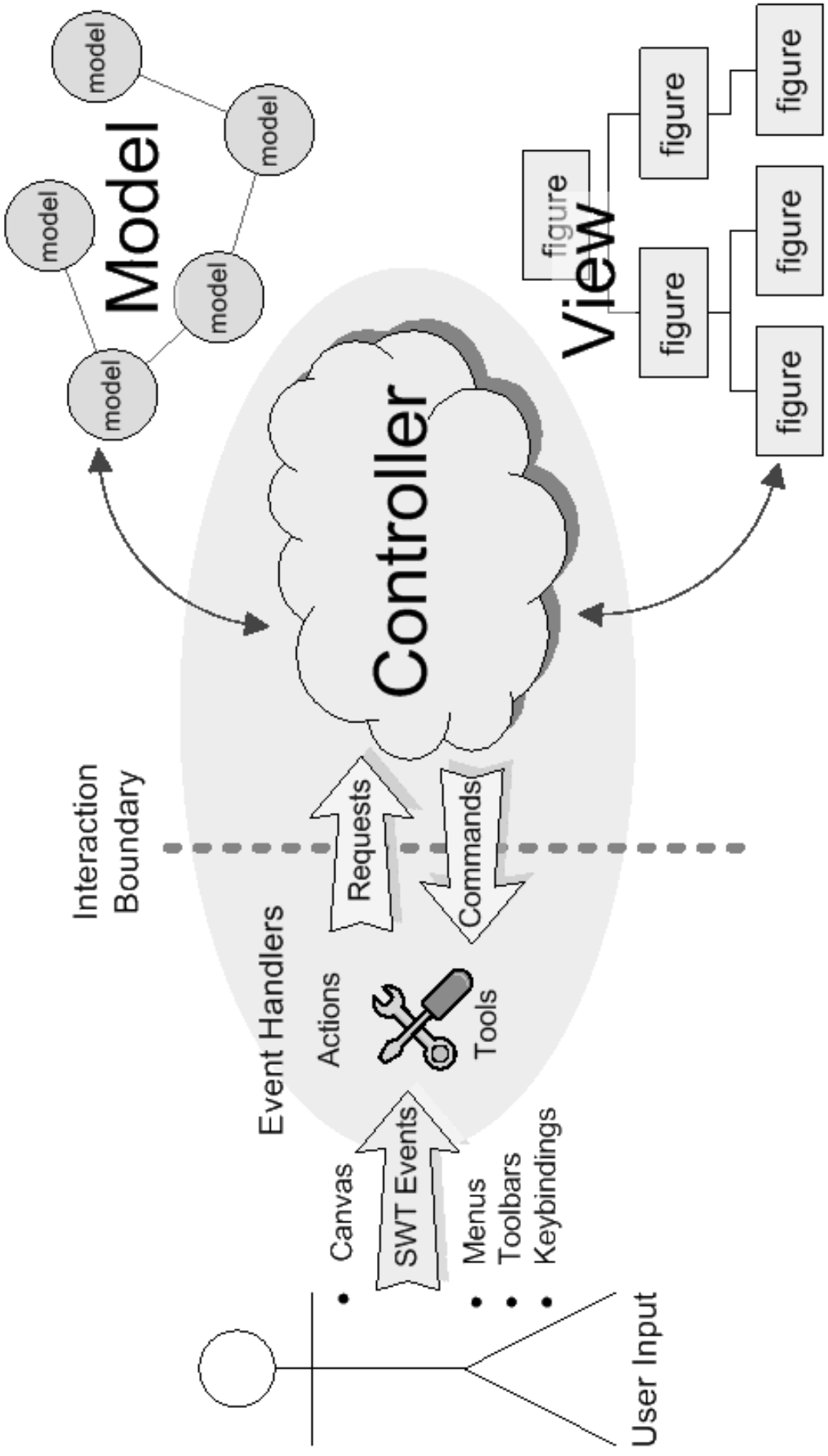
Τα EditParts συσχετίζουν το μοντέλο με την όψη τους, αλλά σχηματίζουν και τα ίδια μια δενδρική δομή. Κάθε editpart έχει παιδιά και συνήθως υπάρχει μια αντίστοιχη σχέση στο μοντέλο. Η σχέση γονέα – παιδιού των editparts επεκτείνεται και στα σχήματά τους, όπου το σχήμα του γονέα περιέχει τα σχήματα των παιδιών. Το αποτέλεσμα είναι τρεις διαφορετικές ιεραρχικές δομές δεδομένων, οι οποίες είναι περίπου παράλληλες η μία με την άλλη (σχήμα 3.8).
Η εξαίρεση σε αυτήν την απλή δενδρική δομή είναι οι συνδέσεις (Connections), οι οποίες αντιπροσωπεύουν ένα σύνδεσμο ή μια συσχέτιση μεταξύ δύο αντικειμένων.
Υπάρχουν δύο διαφορετικές υλοποιήσεις των editparts που παρέχει το GEF. Τα γραφικά editparts, που για όψη τους έχουν ένα σχήμα (figure) του Draw2d, υποστηρίζουν και τη δημιουργία συνδέσεων, οι οποίες είναι και αυτές γραφικά editparts. Τα δενδρικά editparts έχουν για όψη τους ένα αντικείμενο SWT TreeItem.
Οι ευθύνες ενός editpart είναι οι εξής:

- Η δημιουργία και η διατήρηση μιας όψης (είτε figure είτε treeitem)
- Η δημιουργία και η διατήρηση των editparts – παιδιών του
- Η δημιουργία και η διατήρηση των editparts – συνδέσεων
- Η υποστήριξη της επεξεργασίας του μοντέλου

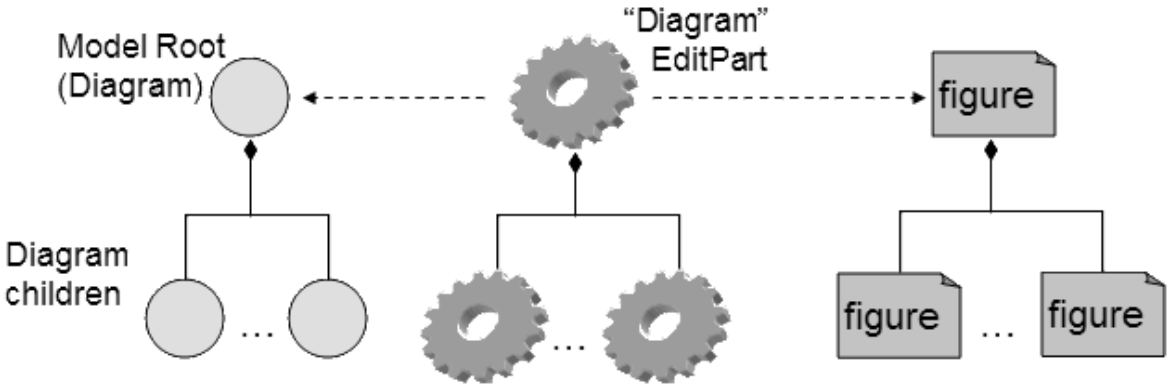
Η διατήρηση της όψης και των editparts προϋποθέτει ότι το editpart θα ενημερώνεται όταν γίνονται αλλαγές στο μοντέλο. Συνήθως αυτό γίνεται με την προσθήκη από το editpart ενός κατάλληλου listener (“ακροατή”) πάνω στο αντικείμενο του μοντέλου με το οποίο συσχετίζεται. Όταν λάβει μια ειδοποίηση ότι έγινε κάποια αλλαγή, ανανεώνει την όψη ή τη δομή του, ανάλογα με το είδος της αλλαγής.

Δημιουργία γραφικής όψης του μοντέλου

Αν έχουμε ένα μοντέλο και μερικά σχημάτα (figures), το επόμενο βήμα είναι να τα εμφανίσουμε στο χρήστη. Οι περισσότερες εφαρμογές χρησιμοποιούν ένα αντικείμενο της κλάσης ScrollingGraphicalViewer, που παρέχει το GEF για αυτό το σκοπό. Η κλάση αυτή είναι μια υλοποίηση ενός viewer που χρησιμοποιεί έναν καμβά FigureCanvas του Draw2d. Το επόμενο βήμα είναι να ορίσουμε το *ριζικό* editpart που θα χρησιμοποιήσουμε, το οποίο συνήθως είναι ένα ScalableRootEditPart και παρέχει κάποιες τυπικές βοηθητικές λειτουργίες, για παράδειγμα δυνατότητα zoom in / zoom out.



Σχήμα 3.7: Η αρχιτεκτονική MVC όπως υλοποιείται από το GEF



Σχήμα 3.8: Δενδρική δομή των EditParts

Στη συνέχεια ορίζονται τα editparts – παιδιά. Τα editparts αυτά είναι υποκλάσεις της βασικής κλάσης AbstractGraphicalEditPart από την οποία κληρονομούν τις απαραίτητες ιδιότητες και συμπεριφορά, και την οποία επεκτείνουν κατάλληλα, υπερσκελίζοντας απαραίτητα τις ακόλουθες μεθόδους:

createFigure() Η μέθοδος που δημιουργεί την όψη (ή αλλιώς σχήμα) ενός editpart.

refreshVisuals() Η μέθοδος αυτή αντανακλά τις ιδιότητες του μοντέλου πάνω στην όψη. Η μέθοδος αυτή καλείται κατά τη δημιουργία του editpart, αλλά και όταν γίνεται κάποια αλλαγή στο μοντέλο, αν χρειάζεται να ανανεωθεί η όψη για να είναι συνεπής με τη νέα κατάσταση του μοντέλου.

getModelChildren() Η μέθοδος αυτή επιστρέφει μια λίστα με τα παιδιά του αντίστοιχου αντικειμένου στο μοντέλο. Χρειάζεται ώστε να μπορεί το edipart να δημιουργεί και τα παιδιά του.

getModelSourceConnections(), getModelTargetConnections() Οι δύο αυτές μέθοδοι πρέπει να υπερσκελίζονται από ένα editpart, αν αυτό έχει τη δυνατότητα να είναι αρχή ή τέλος μιας σύνδεσης. Επιστρέφουν τα αντικείμενα του μοντέλου που αντιπροσωπεύουν τις συνδέσεις.

Τα editparts – συνδέσεις είναι ειδικά editparts που συνδέουν οποιαδήποτε δύο editparts μέσα σε ένα διάγραμμα. Ένα editpart ονομάζεται *κόμβος* (NodeEditPart) αν μπορεί να είναι αρχή ή τέλος μιας σύνδεσης. Τα σχήματα των editparts – συνδέσεων είναι ειδικά σχήματα (συνδέσεις του Draw 2d) αλλά κατά τα άλλα οι συνδέσεις μοιάζουν με τα άλλα editparts στο ότι μπορεί να έχουν και αυτές παιδιά και ιδιότητες που πρέπει να αντανακλώνται στην όψη τους.

Επεξεργασία

Η επεξεργασία είναι συνήθως η πιο περίπλοκη εργασία που κάνει ένα editpart και συμπεριλαμβάνει την πραγματοποίηση αλλαγών στο μοντέλο και την εμφάνιση γραφικής ανατροφοδότησης την ώρα που ο χρήστης αλληλεπιδρά με την όψη.
Η πηγή μιας αλληλεπίδρασης αναπαριστάνεται με ένα *αίτημα* (request), το οποίο δημιουργείται από ένα *εργαλείο* όπως εξηγείται παρακάτω. Το πρώτο βήμα της επεξεργασίας είναι να αποφασίσει το σύστημα ποια editparts επηρεάζονται. Αυτά συνήθως είναι τα επιλεγμένα από το χρήστη editparts ή το editpart που βρίσκεται στην τρέχουσα θέση του δείκτη ποντικιού, το οποίο μπορεί να βρεθεί με τη μέθοδο getTargetEditPart (Request) του EditPartViewer. Η *εντολή* (Command) είναι αυτή που τελικά μεταβάλλει το μοντέλο. Τα editparts είναι κατασκευασμένα ώστε να επιστρέφουν μια εντολή για ένα αίτημα που τους δίνεται. Αν ένα editpart δεν επιστρέφει εντολή για ένα αίτημα, αυτό σημαίνει ότι δεν επιτρέπεται αυτό που ζητήθηκε (για παράδειγμα, μπορεί να απαγορεύεται η διαγραφή κάποιου συγκεκριμένου στοιχείου) και η γραφική διαπροσωπεία δείχνει μια κατάλληλη ένδειξη στο χρήστη.
Τα editparts δε χειρίζονται απευθείας την επεξεργασία, αλλά την αναθέτουν σε *πολιτικές επεξεργασίας* (EditPolicies). Κάθε EditPolicy εστιάζει σε μια εργασία ή σε μια ομάδα συναφών εργασιών και

υλοποιεί μεθόδους getCommand(Request) για να επιστρέφει την κατάλληλη εντολή για ένα δοθέν αίτημα. Κάθε EditPart εγκαθιστά τις EditPolicies που χρειάζεται κατά την ώρα της δημιουργίας του. Οι εντολές χρησιμοποιούνται αντί της απευθείας επεξεργασίας του μοντέλου με σκοπό να ενθυλακώνουν και να συνδυάζουν τις αλλαγές πάνω στο μοντέλο της εφαρμογής. Κάθε εφαρμογή έχει μια μοναδική στοίβα εντολών, και κάθε φορά που εκτελείται μια εντολή, αυτή προστίθεται στη στοίβα. Κάθε εντολή υλοποιεί τις μεθόδους undo() και redo() ώστε ο χρήστης να μπορεί να κάνει αναίρεση και επανάληψη των τελευταίων αλλαγών που έκανε.

Εργαλεία και παλέττα εργαλείων

Τα *εργαλεία* αναλαμβάνουν το χειρισμό των συμβάντων. Μπορεί μια εφαρμογή να έχει πολλαπλά εργαλεία, τα οποία εμφανίζονται συνήθως σε μια *παλέττα* δίπλα στον καμβά, αλλά μόνο ένα από αυτά είναι ενεργό κάθε στιγμή. Η παλέττα χρησιμοποιείται για να επιλέγει ο χρήστης το ενεργό εργαλείο. Τα εργαλεία επιτελούν λειτουργίες όπως:

- Ζητούν από τα editparts να δείξουν ανατροφοδότηση σχετικά με το τρέχον αίτημα
- Ζητούν και παίρνουν εντολές από τα editparts
- Εκτελούν μια εντολή στη στοίβα εντολών
- Ανανεώνουν κατάλληλα το δείκτη του ποντικιού

Το προεπιλεγμένο εργαλείο είναι συνήθως το εργαλείο επιλογής (Selection Tool).

Τύποι διαδράσεων

Ο χρήστης μιας εφαρμογής βασισμένης στο GEF μπορεί να αλληλεπιδρά με αυτήν με διάφορους τρόπους. Με τον όρο αλληλεπίδραση εννοούμε οτιδήποτε επηρεάζει το μοντέλο ή την κατάσταση της γραφικής διαπροσωπείας. Πολλές αλληλεπιδράσεις είναι γραφικές, αλλά μερικές δεν είναι, παραδείγματα:

- Πρόκληση κάποιας *ενέργειας* (Action) (συνήθως από μια γραμμή μενού, μια γραμμή εργαλείων, ή ένα μενού δεξιού κλικ)
- Πάτημα του ποντικιού πάνω σε κάτι
- Σύρσιμο κάποιου αντικειμένου με το ποντίκι
- Άφημα κάποιου αντικειμένου που σύρθηκε (λειτουργία Drag-N-Drop)
- Πάτημα κάποιων πλήκτρων

Υπάρχουν κάποιες βασικές λειτουργίες που μπορεί να εκτελεί ο χρήστης πάνω στο μοντέλο, οι οποίες είναι κοινές για πολλές εφαρμογές και γι αυτό το GEF παρέχει έτοιμες ευκολίες για την υλοποίησή τους. Για παράδειγμα, αν έχουμε έναν γραφικό editor που επεξεργάζεται ψηφιακά λογικά κυκλώματα, μερικές βασικές λειτουργίες θα ήταν η δημιουργία μιας νέας λογικής πύλης, η δημιουργία μιας σύνδεσης μεταξύ δύο πυλών, η μετακίνηση μιας πύλης στο χώρο, η αλλαχή των διαστάσεων ενός οπτικού αντικειμένου, η διαγραφή μιας πύλης κ.ο.κ.
Εξετάζοντας λίγο πιο αναλυτικά το παράδειγμα της διαγραφής ενός στοιχείου, έστω ότι ο χρήστης έχει επιλέξει, με χρήση του εργαλείου επιλογής, μια λογική πύλη (ή μπορεί και περισσότερες). Για να τη διαγράψει από το διάγραμμα θα πρέπει να κάνει μια *ενέργεια διαγραφής* (DeleteAction), κάτι που μπορεί να γίνει είτε από τη γραμμή μενού, είτε από το μενού δεξιού κλικ, είτε από το πληκτρολόγιο με το πλήκτρο Del ή και κάπως αλλιώς. Όπως και αν γίνει η ενέργεια, το αποτέλεσμα είναι να σταλεί ένα *αίτημα διαγραφής* με προκαθορισμένο όνομα REQ_DELETE στο editpart της επιλεγμένης πύλης. Όλα τα editparts πρέπει να έχουν μια editpolicy που είτε υποστηρίζει είτε απαγορεύει τη διαγραφή του αντίστοιχου αντικειμένου. Μια ορθή και πλήρης υλοποίηση της εντολής διαγραφής θα πρέπει να λαμβάνει υπόψιν της αν το αντικείμενο που πρόκειται να διαγραφεί έχει συνδέσεις ή αν τα παιδιά του έχουν συνδέσεις, και να φροντίζει να διαγράφονται και αυτές τις συνδέσεις.

3.3 Απαιτήσεις από την εφαρμογή

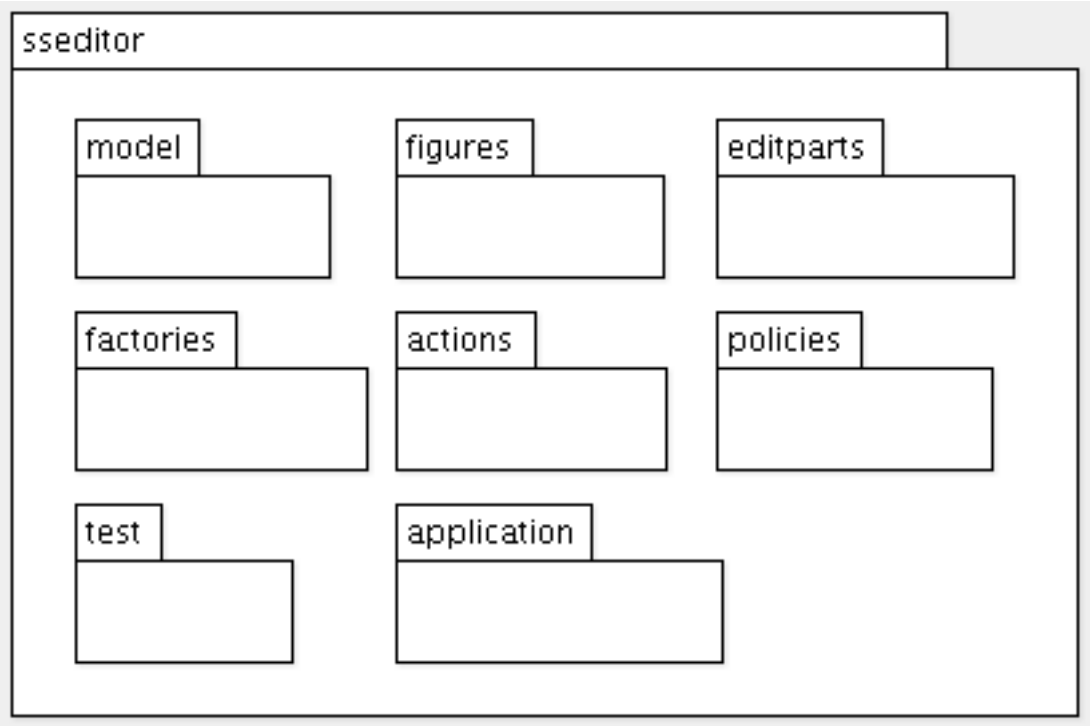
Οι απαιτήσεις που έχουμε από την εφαρμογή μπορούν να συνοψιστούν όπως φαίνεται στο διάγραμμα χρήσεων του σχήματος 3.9[5].
Όπως έχει προαναφερθεί, η εφαρμογή θα κάνει *γραφική επεξεργασία* διαγραμμάτων καταστάσεων σαν αυτά που περιγράψαμε στην ενότητα §2.3. Με τον όρο γραφική επεξεργασία εννοούμε ότι ο χρήστης πρέπει να μπορεί να δημιουργεί με γραφικό τρόπο εκπαιδευτικές δραστηριότητες και υπό συνθήκη μεταβάσεις που τις συνδέουν μεταξύ τους, και να ορίζει αν θέλει ένα ή περισσότερα objectives για κάθε δραστηριότητα. Επίσης θα πρέπει να μπορεί να αλλάζει τις ιδιότητες αυτών των στοιχείων, για παράδειγμα να τα μετονομάζει, να τα διαγράφει, κλπ. Είναι επιθυμητό κατά τη διάρκεια της γραφικής επεξεργασίας ο χρήστης να έχει δυνατότητα αναίρεσης ή επανάληψης της τελευταίας αλλαγής (ή των προηγούμενων αλλαγών) που έκανε στο διάγραμμα.
Η περίπτωση χρήσης "Αποθήκευση εργασίας" σημαίνει ότι ο χρήστης θα μπορεί να αποθηκεύει τα έργα που έχει δημιουργήσει σε μια ενδιάμεση μορφή (αρχείο) και.
Τέλος, αφού ολοκληρώνεται η γραφική επεξεργασία του διαγράμματος, ο χρήστης θα μπορεί να το εξάγει σε περιγραφή Simple Sequencing, δηλαδή με μια εντολή η εφαρμογή θα εκτελεί τον αλγόριθμο (1) μετασχηματισμού που περιγράφτηκε στην ενότητα §2.3. Η τελική μορφή πρέπει να είναι μια ιεραρχία καταλόγων

Σχήμα 3.9: Διάγραμμα χρήσεων της εφαρμογής

3.4 Οργάνωση του πηγαίου κώδικα

Όπως ήδη αναφέρθηκε, οι εφαρμογές που βασίζονται στο πλαίσιο Graphical Editing Framework είναι σχεδιασμένες με βάση την αρχιτεκτονική Model–View–Controller. Το ίδιο ισχύει και για την εφαρμογή αυτής της εργασίας. Ο κώδικας της εφαρμογής, γραμμένος στη γλώσσα Java, είναι οργανωμένος στα πακέτα που φαίνονται στο σχήμα 3.10.
Όπως προαναφέρθηκε, στην αρχιτεκτονική MVC (§3.2.1), το μοντέλο δεν εξαρτάται από την όψη και τον ελεγκτή και επομένως η ανάπτυξή του μπορεί να γίνει ανεξάρτητα από τις άλλες δύο συνιστώσες. Στη συγκεκριμένη περίπτωση, ο κώδικας του μοντέλου γράφτηκε πριν από τις άλλες συνιστώσες που σχετίζονται με το πλαίσιο GEF (όψεις και ελεγκτές) και τον κώδικα που αφορά την ίδια την εφαρμογή και τις επεκτάσεις του eclipse.

[5] Τα UML διαγράμματα αυτής και της επόμενης ενότητας κατασκευάστηκαν με το εργαλείο ArgoUML, http://argouml.tigris.org

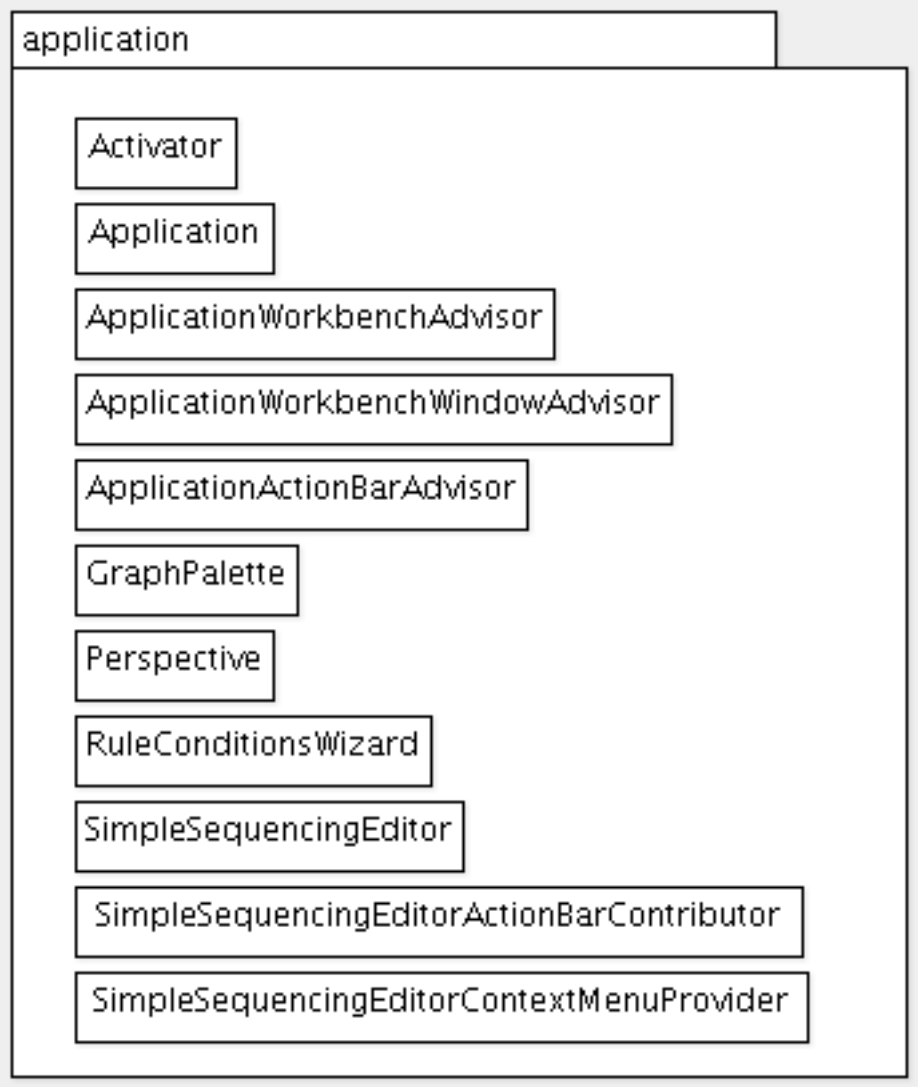


Σχήμα 3.10: Οργάνωση του πηγαίου κώδικα σε πακέτα

Το πακέτο application περιέχει τις κλάσεις που αφορούν τη γενική λειτουργία της εφαρμογής ως plug-in της πλατφόρμας eclipse, όπως για παράδειγμα η αρχικοποίηση της εφαρμογής, η εμφάνιση των διαφόρων τμημάτων της γραφικής διαπροσωπείας, τα μενού, οι μπάρες εργαλείων, οι λειτουργίες αποθήκευσης/ανοίγματος αρχείων δεδομένων, κλπ.
Το πακέτο model περιέχει τις κλάσεις του μοντέλου, οι οποίες προκύπτουν σχεδόν αυτόματα από το μοντέλο δεδομένων της προδιαγραφής Simple Sequencing (ενότητα §2.2). Το πακέτο model.commands τυπικά δεν είναι μέρος του μοντέλου, αλλά συνδέεται στενά με αυτό και περιέχει τις εντολές που μεταβάλλουν το μοντέλο (π.χ. δημιουργία και διαγραφή ενός Activity/Objective/Conditional Transition).
Το πακέτο figures περιέχει τις περιγραφές των σχημάτων Draw2d που αναπαριστάνουν τις κλάσεις Activity και Objective του μοντέλου.
Το πακέτο editparts περιέχει τους ελεγκτές για αυτές τις κλάσεις του μοντέλου που έχουν μια γραφική αναπαράσταση και τις οποίες μπορεί να επεξεργάζεται ο χρήστης απευθείας (δεν ανήκουν όλες οι κλάσεις του μοντέλου σε αυτήν την κατηγορία). Πιο συγκεκριμένα, ο χρήστης μπορεί να επεξεργάζεται το διάγραμμα, τις δραστηριότητες, τα Objectives και τις συνδέσεις μεταξύ των δραστηριοτήτων. Οι κλάσεις του πακέτου editparts.tree είναι παρόμοιες και αφορούν τη δενδρική αναπαράσταση του διαγράμματος. Το πακέτο policies περιέχει τις πολιτικές που αναλαμβάνουν τη δημιουργία των εντολών για τα αιτήματα που δέχονται τα editparts.
Το πακέτο factories περιέχει τις κλάσεις που αναλαμβάνουν τη δημουργία στιγμιοτύπων των κλάσεων του μοντέλου κατά τη διάρκεια της επεξεργασίας από το χρήστη.
Τέλος, το πακέτο test περιέχει κλάσεις που χρησιμοποιούνται για τον έλεγχο ορθότητας κατά την ανάπτυξη της εφαρμογής και δεν επηρεάζει την τελική εφαρμογή που βλέπει ο χρήστης.

3.5 Η εφαρμογή από τη σκοπιά του χρήστη

Ο RCP Simple Sequencing Editor είναι ένας γραφικός επεξεργαστής. Με αυτόν ο χρήστης μπορεί να δημιουργήσει διαγράμματα καταστάσεων όπως αυτά που περιγράφτηκαν στην ενότητα §2.3, τα οποία να περιγράφουν τη δομή ενός πακέτου Simple Sequencing. Η οθόνη που βλέπει ο χρήστης μοιάζει με



Σχήμα 3.11: Το πακέτο application

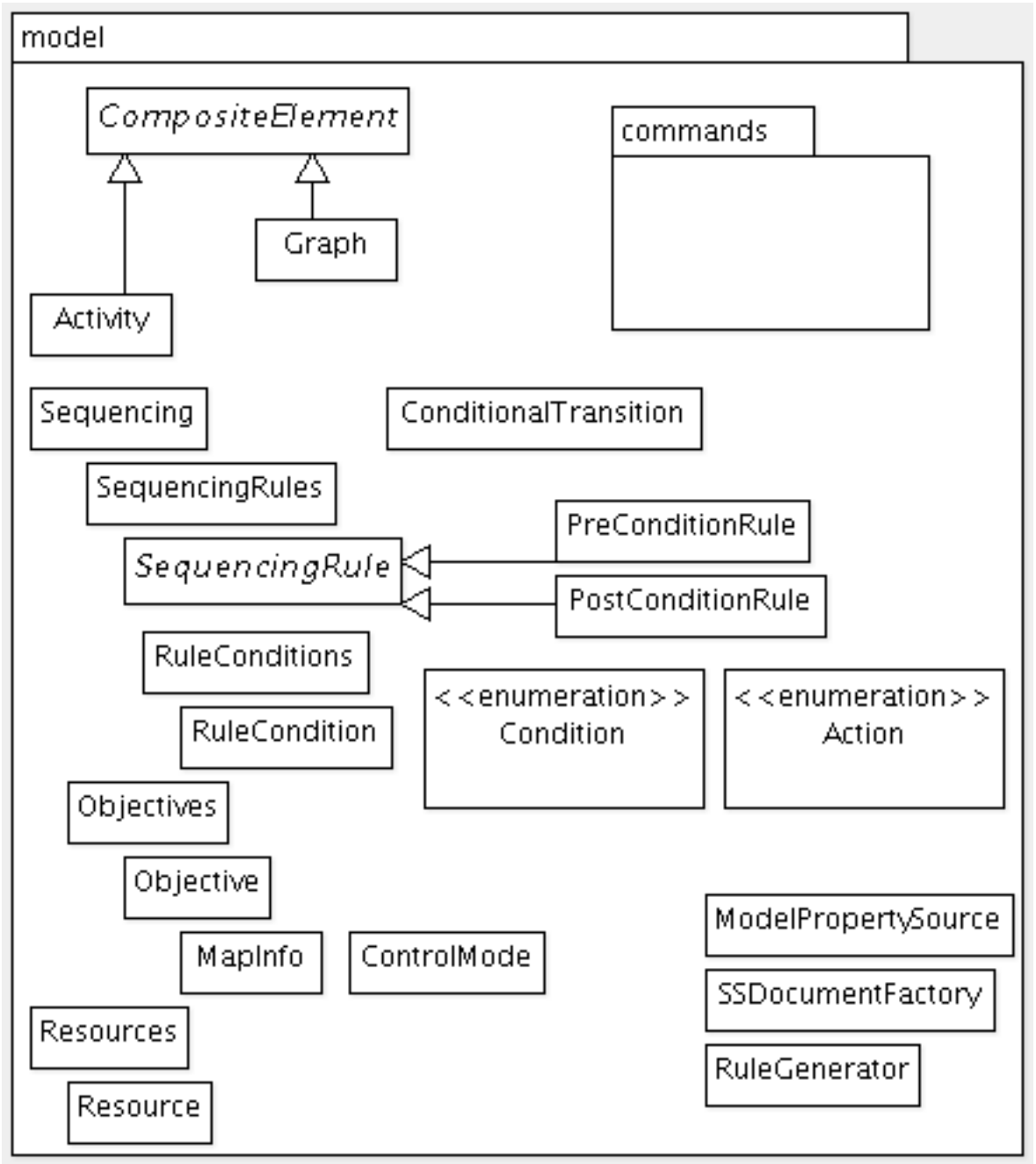
αυτή του σχήματος 3.19.

Δημιουργία/Αποθήκευση/Άνοιγμα αρχείων

Ο editor δημιουργεί και επεξεργάζεται αρχεία με κατάληξη .par. Κάθε νέο αρχείο που δημιουργείται πρέπει να ανήκει σε ένα Project. Έτσι, την πρώτη φορά πρέπει να δημιουργηθεί ένα καινούριο Project (File -> New -> Project). Στη συνέχεια μπορεί ο χρήστης να δημιουργήσει ένα νέο αρχείο .par μέσα σε αυτό το Project.

Γραφική επεξεργασία

Αφού δημιουργήσει το αρχείο, ο χρήστης μπορεί να αρχίσει τη γραφική επεξεργασία. Χρησιμοποιώντας την παλέτα εργαλείων που βρίσκεται στο πλάι, μπορούν να δημιουργήθούν γραφικά αντικείμενα, όπως μερικά Activities (εργαλείο Activity) ή μερικά Objectives (Εργαλείο Objective) για αυτά τα Activities ή μερικά Conditional Transitions (Εργαλείο Conditional Transition) που να συνδέουν δύο Activities μεταξύ τους.



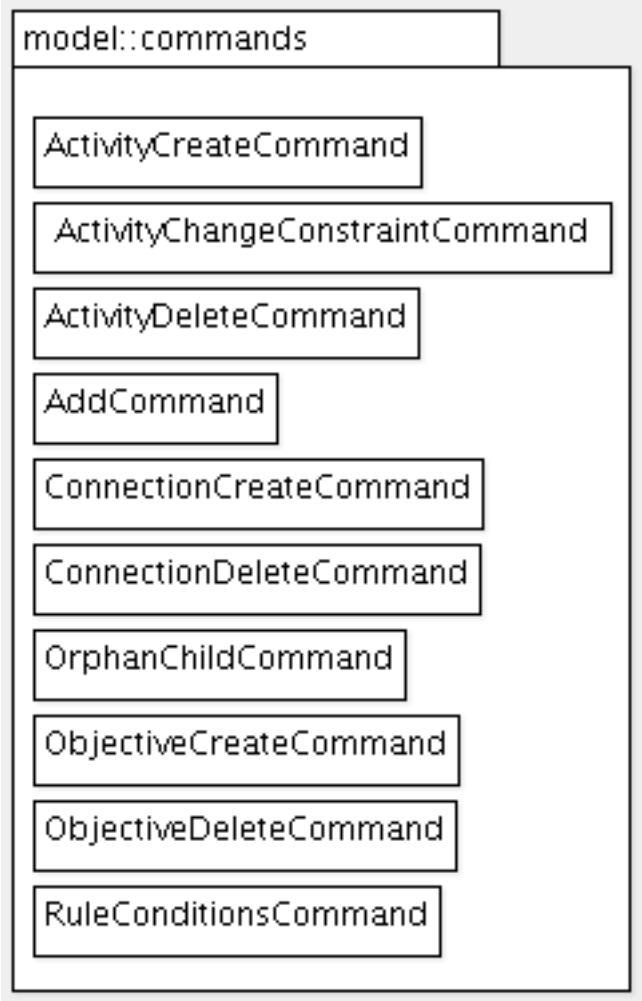
Σχήμα 3.12: Το πακέτο model

Όψη Properties

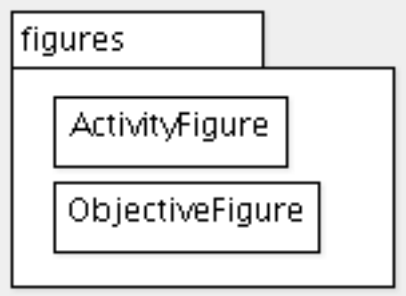
Η όψη αυτή δείχνει κάποιες ιδιότητες του επιλεγμένου γραφικού αντικειμένου, για παράδειγμα τον τύπο του (Activity, Objective κτλ.) ή το όνομά του. Μερικές από τις ιδιότητες μπορούν να μεταβληθούν απευθείας από το χρήστη, π.χ. μπορεί να αλλάξει η ονομασία ενός Objective, ενώ άλλες παρέχουν μόνο πληροφορίες και δεν μπορούν να αλλάξουν.

Όψη Outline

Η όψη Outline δείχνει το γράφο που έχει δημιουργήσει έως τώρα ο χρήστης σε μια δενδρική μορφή.



Σχήμα 3.13: Το πακέτο model.commands



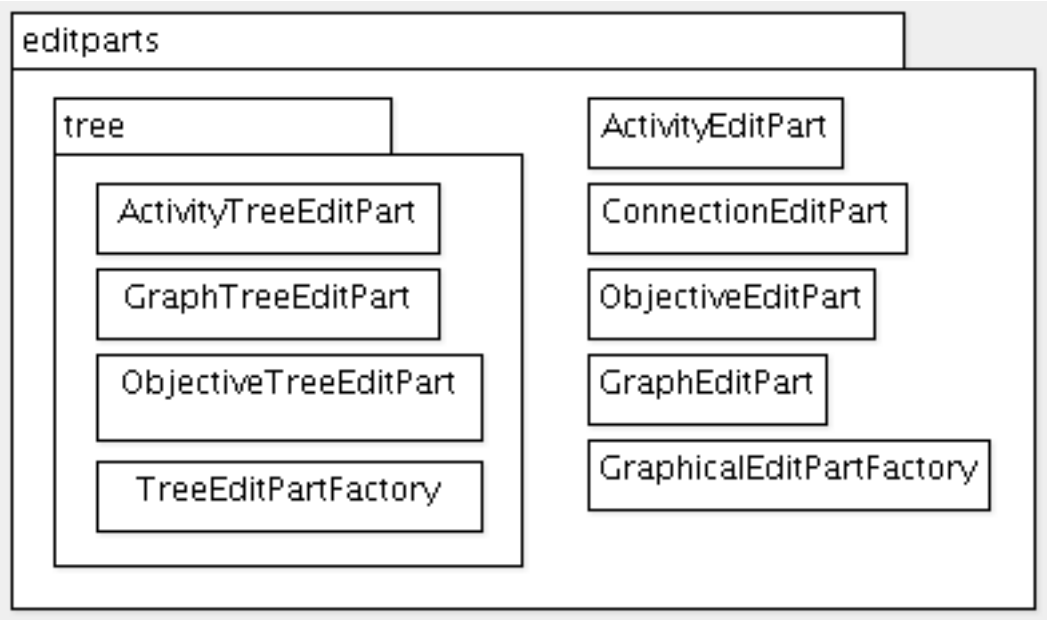
Σχήμα 3.14: Το πακέτο figures

Zoom Manager

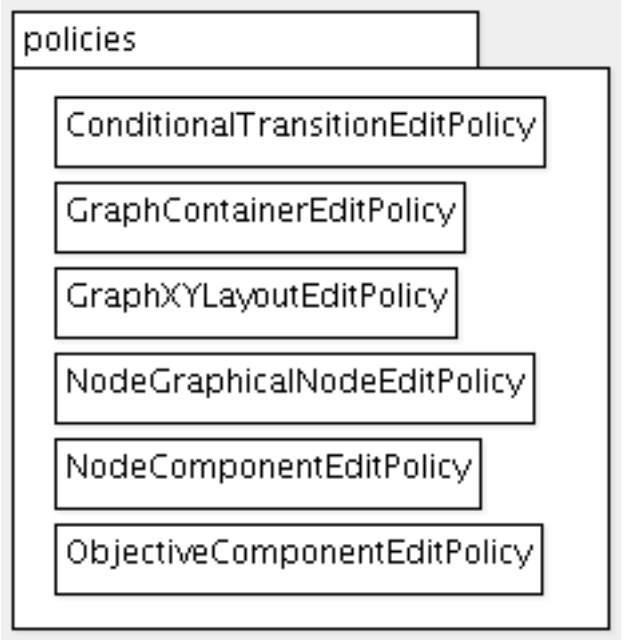
Ο editor παρέχει λειτουργίες Zoom που είναι χρήσιμες οταν το διάγραμμα καταστάσεων που επεξεργάζεαι ο χρήστης είναι αρκετά μεγάλο. Η κλίμακα μεγέθυνσης/σμίκρυνσης μπορεί να επιλέγεται από την μπάρα εργαλείων ή με χρήση της ροδέλας του ποντικιού πάνω στην επιφάνεια επεξεργασίας.

Ενέργειες/Εντολές

Με κλικ επάνω σε ένα γραφικό αντικείμενο, ή με χρήση των εργαλείων Select και Marquee της παλέτας, μπορουν να να επιλέγούν ένα ή περισσότερα γραφικά αντικείμενα. Στη συνέχεια μπορούν να



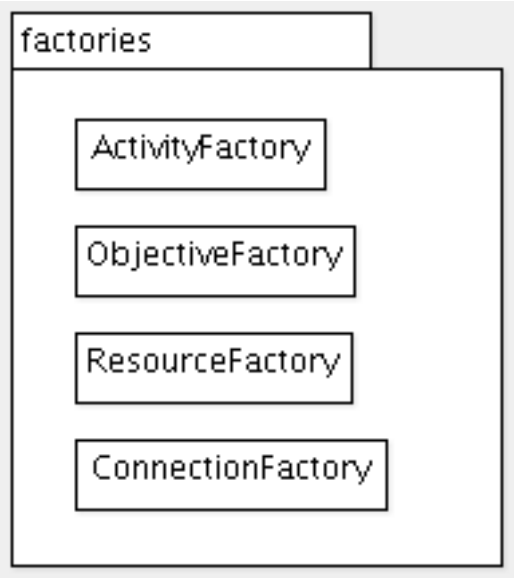
Σχήμα 3.15: Το πακέτο editparts



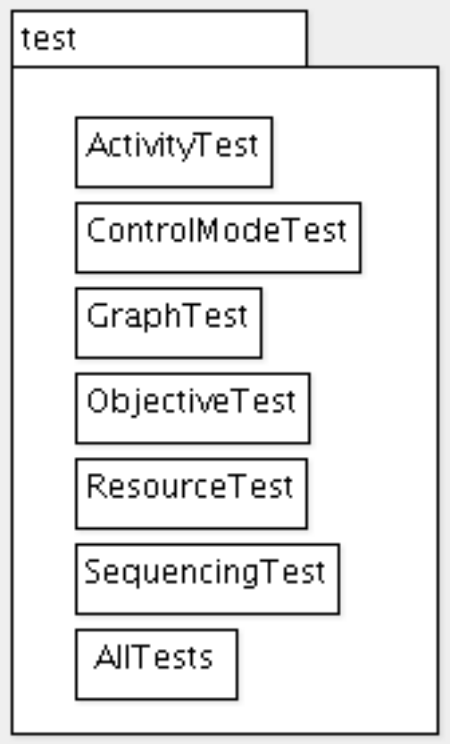
Σχήμα 3.16: Το πακέτο policies

εκτελέστούν διάφορες λειτουργίες με αυτά τα αντικείμενα:

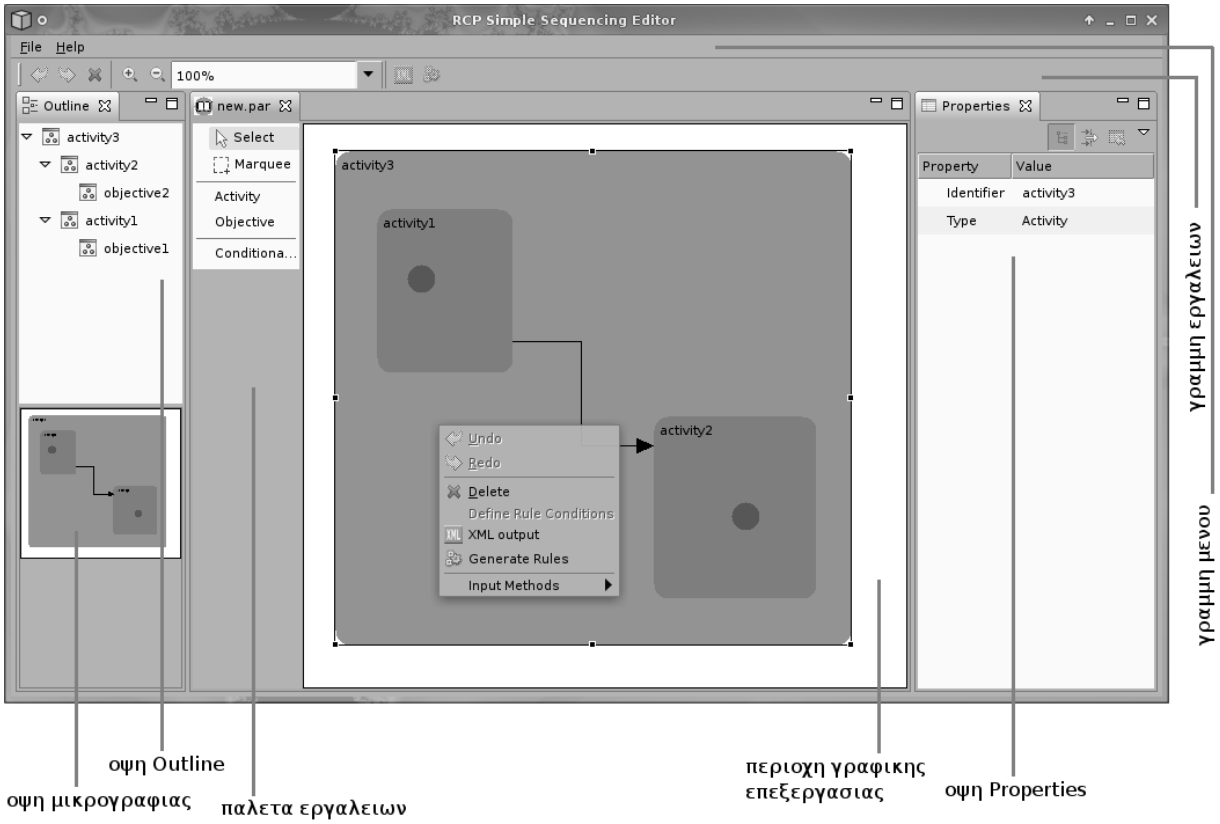
- Τα γραφικά αντικείμενα μπορούν να μετακινηθούν στην οθόνη και να αλλάξουν διαστάσεις (με τη μέθοδο drag 'n drop). Ειδικά τα Activities μπορούν να μετακινηθούν έτσι ώστε να αλλάξουν γονέα. Αυτή η ενέργεια αλλάζει και τη δομή του γράφου.
- Μπορεί να γίνει διαγραφή ενός αντικειμένου με μια ενέργεια delete.
- Μπορούν να οριστούν Rule Conditions για μια επιλεγμένη Conditional Transition με την ενέργεια Define Rule Conditions. Σε αυτήν την περίπτωση εμφανίζεται ένας wizard που βοηθά το χρήστη να επιλέξει τις συνθήκες οι οποίες πρέπει να ικανοποιούνται για να ενεργοποιηθεί αυτή η μετάβαση.



Σχήμα 3.17: Το πακέτο factories



Σχήμα 3.18: Το πακέτο test



Σχήμα 3.19: Το γραφικό περιβάλλον της εφαρμογής

- Όταν έχει ολοκληρωθεί η επεξεργασία, μπορεί να δημιουργηθεί το τελικό έγγραφο imsmanifest.xml, που περιγράφει το πακέτο Simple Sequencing, με την εντολή Generate Rules.

Όλες οι ενέργειες που επηρεάζουν το μοντέλο που επεξεργάζεται ο χρήστης μπορούν να ανακληθούν ή να επαναληφθούν με τις ενέργειες Undo/Redo. Κάθε ενέργεια είναι δυνατό να εκτελεστεί με περισσότερους από έναν τρόπους, π.χ. η ενέργεια Delete (διαγραφή γραφικού αντικειμένου) μπορεί να κληθεί για το επιλεγμένο αντικείμενο είτε από το πληκτρολόγιο (πλήκτρο Del) είτε από το μενού δεξί κλίκ είτε από τη μπάρα εργαλείων.

3.6 Λειτουργικός δυναμικός έλεγχος

3.6.1 Έλεγχος του μοντέλου – Το πλαίσιο ελέγχου JUnit

Το JUnit είναι ένα πλαίσιο που χρησιμεύει στην πραγματοποίηση *ελέγχου μονάδας* για τη γλώσσα προγραμματισμού Java. Ο χρήστης – προγραμματιστής δημιουργεί δοκιμές (τεστ) μονάδων, οι οποίες ορίζουν τις απαιτήσεις του κώδικα, πολλές φορές πριν τη συγγραφή του ίδιου του κώδικα (μια τεχνική που είναι γνωστή ως ανάπτυξη οδηγούμενη από τον έλεγχο, test–driven development [6]). Οι δοκιμές

[6] Η τεχνική test–driven development είναι μια τεχνική ανάπτυξης λογισμικού που χρησιμοποιεί μικρούς κύκλους (επαναλήψεις) ανάπτυξης βασισμένους σε περιπτώσεις δοκιμής που έχουν γραφτεί πριν από τον κώδικα. Σε κάθε κύκλο παράγεται ο κώδικας που απαιτείται για να περάσει η εφαρμογή τα τεστ αυτού του κύκλου με επιτυχία. Μια βασική ιδέα της τεχνικής είναι ότι η συγγραφή των τεστ πριν από τον κώδικα διευκολύνει τη γρήγορη πραγματοποίηση αλλαγών στον κώδικα με βάση τα αποτελέσματα των τεστ.

Ο κύκλος ανάπτυξης αποτελείται από τα εξής βήματα: Προσθήκη μιας δοκιμής (test), εκτέλεση όλων των δοκιμών για να δούμε αν η καινούρια δοκιμή αποτυγχάνει, συγγραφή κώδικα για να επιτύχει η νέα δοκιμή, εκτέλεση όλων των δοκιμών για να δούμε αν επιτυγχάνουν όλες, και τέλος πραγματοποίηση όποιων άλλων αλλαγών χρειάζονται στον κώδικα.

Μετά την ολοκλήρωση κάθε κύκλου ξεκινά ένας νέος με μια νέα δοκιμή. Μετά από κάθε αλλαγή, είναι πολύ εύκολη η εκτέλεση όλων των δοκιμών για να εντοπίζουμε γρήγορα πιθανά λάθη που προκλήθηκαν από την τελευταία αλλαγή και

περιέχουν *υποθέσεις* (assertions) που μπορεί να είναι αληθείς ή ψευδείς. Η γρήγορη εκτέλεση των δοκιμών εξασφαλίζει τη σωστή συμπεριφορά των μονάδων ενώ ο κώδικας εξελίσσεται και τροποποιείται. Ο ρόλος ενός πλαισίου όπως είναι το JUnit είναι να διευκολύνει τον προγραμματιστή στη δημιουργία και στην αυτόματη εκτέλεση πολλών περιπτώσεων δοκιμής.
Το JUnit χρησιμοποιήθηκε για το δυναμικό έλεγχο των κλάσεων που αποτελούν το μοντέλο της εφαρμογής (πακέτο model). Όλες οι κλάσεις που ανήκουν στο πακέτο test είναι περιπτώσεις δοκιμής του JUnit. Στο παράδειγμα του πηγαίου κώδικα 3.8 φαίνεται ένα τμήμα της κλάσης test.ActivityTest, η οποία δοκιμάζει τις μεθόδους της κλάσης Activity.

Πηγαίος Κώδικας 3.8: ActivityTest.java

```
package test;

import model.Activity;
import junit.framework.TestCase;
import junit.framework.Assert;
// ...

public class ActivityTest extends TestCase {
    private Activity r;
    private Activity i;
    // ...

    protected void setUp() {
        r = new Activity("root", null);
        i = new Activity("i", r);
        ii = new Activity("ii", r);
        //...
    }

    public void testGetFather() { //...
    }

    //..

    public void testGetLeftSiblings() {
        Assert.assertTrue(r.getLeftSiblings().isEmpty());
        Assert.assertTrue(i.getLeftSiblings().isEmpty());
        Activity expected [] = {i};
        Assert.assertEquals(Arrays.asList(expected),
                            ii.getLeftSiblings());
        Assert.assertTrue(iia.getLeftSiblings().isEmpty());
        expected [0] = iia;
        Assert.assertEquals(Arrays.asList(expected),
                            iib.getLeftSiblings());
        Activity expected2 [] = {iia, iib};
        Assert.assertEquals(Arrays.asList(expected2),
                            iic.getLeftSiblings());
    }

    public void testNearestCommonAncestor() {
```

προκαλούν την αποτυχία μιας δοκιμής.

```
        Assert.assertEquals(r, r.nearestCommonAncestor(ii));
        Assert.assertEquals(r, ii.nearestCommonAncestor(r));
        Assert.assertEquals(r, i.nearestCommonAncestor(ii));
        Assert.assertEquals(r, ii.nearestCommonAncestor(i));
        Assert.assertEquals(r, i.nearestCommonAncestor(iib));
        Assert.assertEquals(r, iib2.nearestCommonAncestor(i));
        Assert.assertEquals(r, i.nearestCommonAncestor(iic1));
        Assert.assertEquals(r, iic1.nearestCommonAncestor(i));
        Assert.assertEquals(ii, iia.nearestCommonAncestor(iic2));
        Assert.assertEquals(ii, iic2.nearestCommonAncestor(iia));
        Assert.assertEquals(iia, iia1.nearestCommonAncestor(iia2));
        Assert.assertEquals(iia, iia2.nearestCommonAncestor(iia1));
        Assert.assertEquals(i, i.nearestCommonAncestor(ib));
        Assert.assertEquals(iia, iia2.nearestCommonAncestor(iia1));
        Assert.assertEquals(i, ib.nearestCommonAncestor(i));
    }
}
```

Στη μέθοδο setUp() γίνεται αρχικοποίηση του δέντρου των δραστηριοτήτων που φαίνεται στο παράδειγμα του σχήματος 2.16, δηλαδή δημιουργείται ο ριζικός κόμβος r, τα παιδιά του i και ii, τα παιδιά αυτών, κ.ο.κ. Κάθε μία από τις υπόλοιπες μεθόδους δοκιμάζει την αντίστοιχη μέθοδο της κλάσης Activity. Για παράδειγμα, στον αλγόριθμο μετασχηματισμού του διαγράμματος καταστάσεων σε περιγραφή Simple Sequencing (§2.3.2) χρειάζεται να ξέρουμε ποιά είναι τα αριστερά αδέλφια ενός κόμβου και η μέθοδος getLeftSiblings() επιστρέφει μια λίστα με αυτό το αποτέλεσμα. Η μέθοδος testGetLeftSiblings(), που παρουσιάζεται εδώ, χρησιμοποιείται για να συγκρίνει τα αναμενόμενα αποτελέσματα που μας είναι ήδη γνωστά (π.χ. τα αριστερά αδέλφια της II.C είναι οι II.A και II.B) με τα αποτελέσματα της εκτέλεσης της μεθόδου. Ομοίως, η επόμενη μέθοδος δοκιμάζει τη μέθοδο που βρίσκει τον κοντινότερο κοινό πρόγονο δύο κόμβων.

Κατά την εκτέλεση των δοκιμών, αν τα δύο αποτελέσματα συμπίπτουν, η αντίστοιχη υπόθεση assertEquals() ή assertTrue() επαληθεύεται. Για να επιτύχει η συγκεκριμένη δοκιμή πρέπει να επαληθευτούν όλες οι υποθέσεις που βρίσκονται μέσα στην κλάση ActivityTest. Αν όμως κάποια από τις υποθέσεις δείξει λάθος αποτέλεσμα, τότε πρέπει να γίνουν αλλαγές στον κώδικα και επανεκτέλεση των δοκιμών.

Κεφάλαιο 4

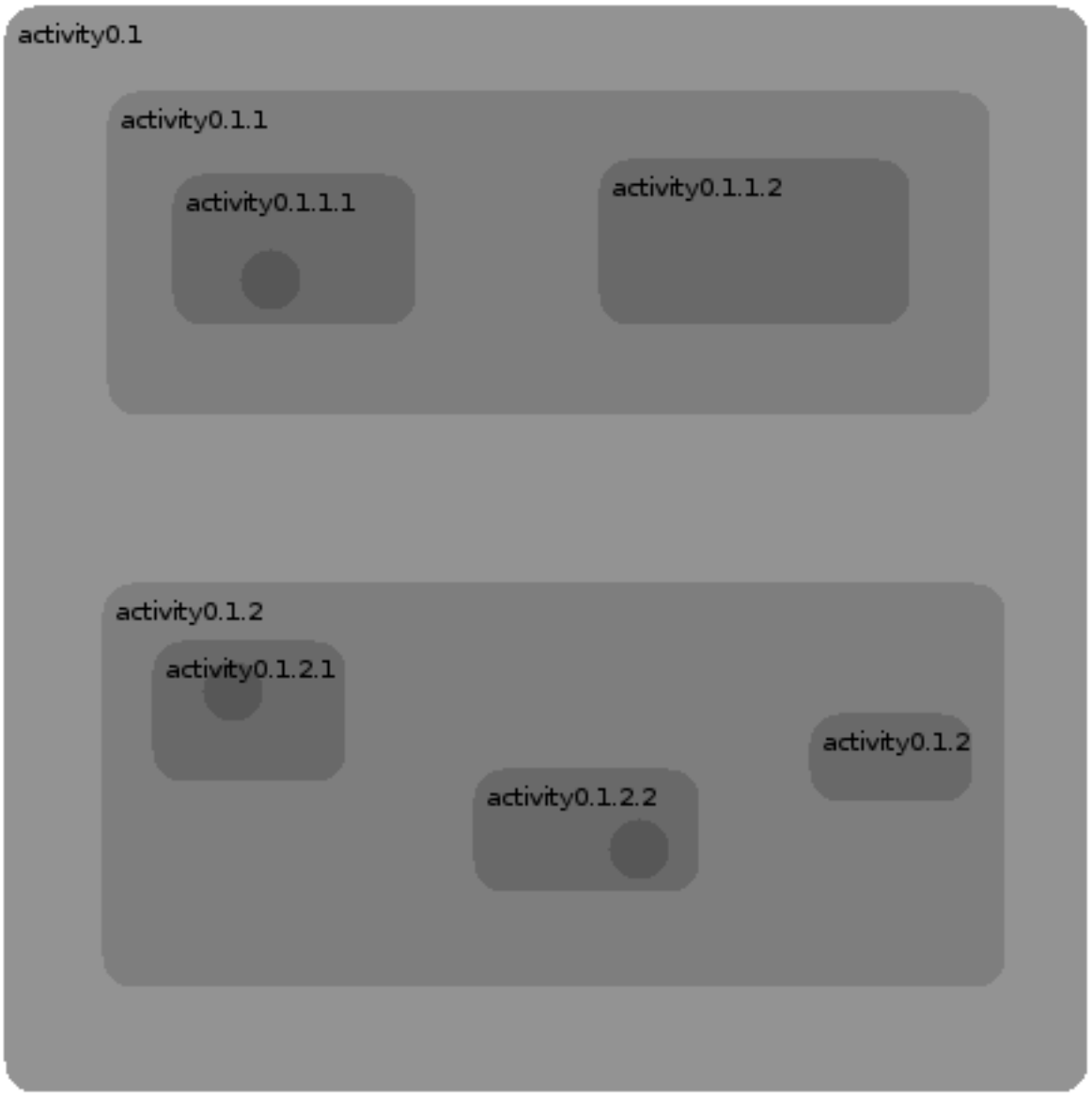
Αξιολόγηση

4.1 Επικύρωση του παραγόμενου εγγράφου Simple Sequencing

Οι προδιαγραφές Content Packaging και Simple Sequencing του οργανισμού IMS Global Learning Consortium έχουν, όπως περιγράφηκε στο κεφάλαιο 2, ως γλώσσα υλοποίησης την XML.
Οι προδιαγραφές ορίζουν ότι για να είναι έγκυρο ένα έγγραφο IMS Manifest, πρέπει η δομή του να υπακούει στους κανόνες που ορίζονται σε κάποια *σχήματα* XML (βλ. παράρτημα A□): Συγκεκριμένα, η προδιαγραφή Content Packaging ορίζει το σχήμα imscp_v1p1.xsd και η προδιαγραφή Simple Sequencing το imsss_v1p0.xsd (τα οποία περιέχουν και αναφορές σε άλλα υπο-σχήματα για κάθε πλευρά της περιγραφής, π.χ. imsss_v1p0objective.xsd, imsss_v1p0seqrule.xsd, συνολικά 11 αρχεία XSD). Τα αρχεία αυτά διανέμονται μαζί με το υπόλοιπο περιεχόμενο και περιλαμβάνονται μέσα στο πακέτο (Content Package).
Για να διαπιστώσουμε την ορθή λειτουργία της εφαρμογής, το πρώτο βήμα είναι να επιβεβαιώσουμε ότι τα αποτελέσματα που παράγει είναι έγκυρες περιγραφές Content Packaging και Simple Sequencing. Πρέπει δηλαδή να κάνουμε συντακτική ανάλυση της παραγόμενης XML με ένα κατάλληλο εργαλείο. Το εργαλείο που χρησιμοποιήθηκε ήταν το εργαλείο συντακτικής ανάλυσης XML Xerces Java Parser (http://xerces.apache.org/xerces-j).
Οι δοκιμές γίνονται με την εξής διαδικασία: Δημιουργούμε ένα διάγραμμα καταστάσεων με τον γραφικό επεξεργαστή του εργαλείου, εκτελούμε την κατάλληλη ενέργεια (εκτέλεση του αλγόριθμου μετασχηματισμού σε περιγραφή Simple Sequencing), παίρνουμε το παραγόμενο IMS Manifest και το ελέγχουμε με το συντακτικό αναλυτή. Η διαδικασία αυτή επαναλαμβάνεται για αρκετά διαγράμματα.
Ένα πολύ απλό διάγραμμα σχεδιασμένο μέσα στο γραφικό επεξεργαστή φαίνεται στο σχήμα 4.1 και ο κώδικας 4.1 περιέχει το παραγόμενο αρχείο manifest για αυτό το διάγραμμα. Το αρχείο imsmanifest.xml περνά με επιτυχία τον έλεγχο από τον Xerces parser.

Πηγαίος Κώδικας 4.1: Παραγόμενο imsmanifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
        xmlns:imsss="http://www.imsglobal.org/xsd/imsss"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1.xsd
        imscp_v1p1p4.xsd http://www.imsglobal.org/xsd/ imsss_v1p0.xsd"
        identifier="SSedit_generated_document">
  <organizations>
    <organization identifier="Course">
      <title>activity0</title>
      <item identifier="activity0.1" identifierref="RESOURCE1">
        <item identifier="activity0.1.1" identifierref="RESOURCE3">
          <item identifier="activity0.1.1.1" identifierref="RESOURCE5">
            <imsss:sequencing>
              <imsss:objectives>
```



Σχήμα 4.1: Παράδειγμα διαγράμματος μέσα στο γραφικό επεξεργαστή

```
            <imsss:primaryObjective objectiveID="objective6" />
          </imsss:objectives>
        </imsss:sequencing>
      </item>
      <item identifier="activity0.1.1.2" identifierref="RESOURCE6" />
    </item>
    <item identifier="activity0.1.2" identifierref="RESOURCE4">
      <item identifier="activity0.1.2.1" identifierref="RESOURCE7" />
      <item identifier="activity0.1.2.2" identifierref="RESOURCE8" />
      <item identifier="activity0.1.2.3" identifierref="RESOURCE9" />
    </item>
  </item>
 </organization>
</organizations>
<resources>
```

```
    <resource identifier="RESOURCE3" type="webcontent"
        href="RESOURCE3.html" />
    <resource identifier="RESOURCE9" type="webcontent"
        href="RESOURCE9.html" />
    <resource identifier="RESOURCE8" type="webcontent"
        href="RESOURCE8.html" />
    <resource identifier="RESOURCE5" type="webcontent"
        href="RESOURCE5.html" />
    <resource identifier="RESOURCE7" type="webcontent"
        href="RESOURCE7.html" />
    <resource identifier="RESOURCE4" type="webcontent"
        href="RESOURCE4.html" />
    <resource identifier="RESOURCE1" type="webcontent"
        href="RESOURCE1.html" />
    <resource identifier="RESOURCE6" type="webcontent"
        href="RESOURCE6.html" />
  </resources>
</manifest>
```

4.2 Δοκιμές του παραγόμενου εγγράφου Simple Sequencing σε ένα περιβάλλον χρόνου εκτέλεσης

Το επόμενο βήμα είναι η δοκιμή σε κάποιο σύστημα χρόνου εκτέλεσης για να διαπιστώσουμε, εκτός από την εγκυρότητα, ότι η εφαρμογή μπορεί πράγματι να παράγει την επιθυμητή συμπεριφορά σε κάποιο μαθητή που αλληλεπιδρά με το εκπαιδευτικό περιεχόμενο που δημιουργήθηκε.
Μερικά συστήματα στα οποία μπορούμε να δοκιμάσουμε το αποτέλεσμα της εκτέλεσης της εφαρμογής μας είναι το SCORM[1] Sample Run-Time Environment (http://www.adlnet.gov) το RELOAD SCORM Player (www.reload.ac.uk) ή το σύστημα LMS είναι που αναπτύχθηκε στα πλαίσια της εργασίας [1].

4.3 Σχετικά έργα

Το έργο RELOAD[2] είναι ένα έργο που εστιάζει στην ανάπτυξη εργαλείων τα οποία βασίζονται στις προδιαγραφές που αφορούν τη διαλειτουργικότητα αναμεσα στις νέες τεχνολογίες εκμάθησης. Στα πλαίσια του έργου έχουν αναπτυχθεί αρκετά διαφορετικά εργαλεία λογισμικού που αφορούν τη συγγραφή (από το δημιουργό) ή την παράδοση (στο μαθητή) του εκπαιδευτικού περιεχομένου. Τα εργαλεία αυτά είναι ο RELOAD Editor, ο Learning Design[3] Editor, ο SCORM 1.2 Player, και ο Learning Design Player.
Ο RELOAD Editor έχει δύο εκδόσεις, τις Classic και Eclipse-based. Η δεύτερη είναι η νεότερη και βασίζεται στην πλατφόρμα Eclipse Rich Client Platform, (βλ. §3.1.3), όπως και η δική μας εφαρμογή. Πρόκειται για μια εφαρμογή συγγραφής εκπαιδευτικού περιεχομένου η οποία υποστηρίζει τις

[1] Η προδιαγραφή SCORM (Sharable Content Object Reference Model) είναι μια είναι μια συλλογή προδιαγραφών σχετικών με το δικτυακό e-learning, της πρωτοβουλίας Advanced Distributed Learning (ADL) που προέρχεται από το υπουργείο Άμυνας της κυβέρνησης των ΗΠΑ. Ορίζει τον τρόπο επικοινωνίας μεταξύ του εκπαιδευτικού περιεχομένου και ενός συστήματος – εξυπηρετητή που ονομάζεται περιβάλλον χρόνου εκτέλεσης (συνήθως μια λειτουργία του Learning Management System). Η προδιαγραφή SCORM ορίζει επίσης τον τρόπο με τον οποίο το εκπαιδευτικό περιεχόμενο μπορεί να πακετάρεται μέσα σε ενα αρχείο ZIP για εύκολη μεταφορά [13].

[2] Reusable eLearning Object Authoring and Delivery, [14].

[3] Η προδιαγραφή IMS Learning Design είναι μια προδιαγραφή του IMS Global Learning Consortium για μια μεταγλώσσα που επιτρέπει τη μοντελοποίηση των διεργασιών εκμάθησης [15].

εξής προδιαγραφές: IMS Metadata, IEEE LOM, IMS Content Packaging και επίσης SCORM 1.2 και SCORM 2004 (η οποία ενσωματώνει μεταξύ άλλων προδιαγραφών και την IMS Simple Sequencing). Η εφαρμογή αυτή προσφέρεται για σύγκριση με τη δική μας, επειδή εξυπηρετεί παρόμοιες ανάγκες. Ο RELOAD Editor επιτρέπει στο χρήστη (μεταξύ άλλων) τη δημιουργία ενός πακέτου ADL SCORM 2004. Το πακέτο αυτό μπορεί να περιέχει πληροφορίες αλληλουχίας οργανωμένες σύμφωνα με την προδιαγραφή Simple Sequencing. Ο χρήστης βλέπει το πακέτο (Content Package) που επεξεργάζεται σαν ένα δέντρο οργανωμένο ιεραρχικά, και μπορεί να προσθέτει αντικείμενα sequencing (για παράδειγμα ένα νέο Objective, έναν νέο κανόνα, κτλ.) αν ενεργοποιήσει μια κατάλληλη φόρμα. Η φόρμα αυτή περιγράφει το στοιχείο sequencing αναλυτικά με όρους της προδιαγραφής Simple Sequencing και έτσι ο χρήστης έχει τον απόλυτο έλεγχο όλων των μεταβλητών κάθε στοιχείου. Ουσιαστικά είναι ένας γραφικός τρόπος επεξεργασίας του στοιχείου που διευκολύνει το χρήστη ώστε να μη χρειάζεται να γράφει κώδικα XML, αλλά εξακολουθεί να απαιτεί από αυτόν να γνωρίζει τις προδιαγραφές του IMS και τη σημασιολογία τους, τουλάχιστον εκείνων των στοιχείων που θέλει να χρησιμοποιήσει. Επιπλέον, αν ο χρήστης θέλει να δημιουργήσει μια πιο περίπλοκη συμπεριφορά αλληλουχίας, όπως η υπό συνθήκη μετάβαση (ενότητα §2.3) πρέπει να σκεφτεί σε ποιες δραστηριότητες του δέντρου θα προσθέσει κανόνες, τι είδους κανόνες θα είναι αυτοί (PreCondition, PostCondition, ExitCondition;) να δημιουργήσει στοιχεία MapInfo, κ.λπ., με λίγα λόγια ουσιαστικά να εκτελέσει με το χέρι τον αλγόριθμο μετασχηματισμού (1).

Από τα παραπάνω είναι φανερό ότι ο RELOAD Editor απαιτεί από το χρήστη του βαθιά γνώση των προδιαγραφών και επομένως, όσο αφορά τη διάκριση μεταξύ των εργαλείων συγγραφής εκπαιδευτικού περιεχομένου που έγινε στην ενότητα §2.3, είναι ένα εργαλείο χαμηλού επιπέδου συγκρινόμενο με τη δική μας εφαρμογή, η οποία προσπαθεί να αποκρύπτει την πολυπλοκότητα των προδιαγραφών από το χρήστη. Από την άλλη μεριά, βέβαια, ο RELOAD Editor είναι ένα ώριμο εργαλείο που έχει αναπτυχθεί για πολλά χρόνια από μεγάλο αριθμό ανθρώπων στα πανεπιστήμια του Bolton και του Strathclyde και υλοποιεί πλήρως την προδιαγραφή Simple Sequencing καθώς και άλλες προδιαγραφές, σε αντίθεση με τη δική μας εφαρμογή που υλοποιεί μόνο την IMS Content Packaging και ένα υποσύνολο της IMS Simple Sequencing.

4.4 Ιδέες για μελλοντική εργασία

Για να είναι πλήρης η αξιολόγηση της προσέγγισης και της εφαρμογής μας θα έπρεπε να γίνουν συστηματικές δοκιμές της εφαρμογής από χρήστες – δημιουργούς εκπαιδευτικού περιεχομένου. Στη συνέχεια οι χρήστες αυτοί μπορεί να κληθούν να απαντήσουν σε κάποιες ερωτήσεις ώστε να εξαχθούν συμπεράσματα για τη χρηστικότητα του προγράμματος. Τέτοια συμπεράσματα θα μπορούσαν να χρησιμοποιηθούν για να βελτιωθεί το εργαλείο σε όποια κατεύθυνση πιθανόν υποδείξουν οι χρήστες. Αυτή η πλήρης αξιολόγηση είναι πιθανό αντικείμενο μελλοντικής εργασίας.

Πάντως, σχετικά με τη μελλοντική εξέλιξη και βελτίωση της εφαρμογής, μια κατεύθυνση θα μπορούσε να είναι η πλήρης υλοποίηση της προδιαγραφής Simple Sequencing δηλαδή η δυνατότητα επεξεργασίας από το χρήστη στοιχείων limit conditions, rollup rules, randomization Controls και delivery Controls, τα οποία αυτή τη στιγμή δεν υποστηρίζονται. Η υλοποίηση αυτών των τμημάτων της προδιαγραφής δεν είναι δύσκολη από τεχνική άποψη, αλλά το ζήτημα είναι να βρεθούν τρόποι ώστε να μη θυσιαστεί η ευχρηστία του εργαλείου με την προσθήκη των νέων δυνατοτήτων.

Μια δεύτερη ιδέα, η οποία αφορά τη διευκόλυνση του χρήστη της εφαρμογής, είναι η σύνδεσή της με κάποιον εξωτερικό ή ενσωματωμένο editor μέσα στον οποίο θα γίνεται η επεξεργασία των πόρων (resources), για παράδειγμα ενός επεξεργαστή HTML, αφού οι σελίδες HTML είναι πολύ συνηθισμένος τύπος αρχείων πόρων. Με αυτόν τον τρόπο θα μπορεί ο δημιουργός του εκπαιδευτικού περιεχομένου να χρησιμοποιεί μόνο μια εφαρμογή καθ' όλη τη διάρκεια της συγγραφής, ενώ τώρα απαιτείται ξεχωριστό εργαλείο για τη συγγραφή των πόρων. Η σύνδεση αυτή δεν προβλέπεται να είναι δύσκολη, λόγω της επεκτασιμότητας που παρέχει η αρχιτεκτονική των plug-ins του eclipse και των πολλών διαθέσιμων επεξεργαστών HTML.

Τέλος, ίσως θα ήταν σκόπιμη η δυνατότητα ενσωμάτωσης στο παραγόμενο από την εφαρμογή πακέτο

και στοιχείων από άλλες προδιαγραφές σχετιζόμενες με το e-learning, όπως η IMS Metadata, IMS Question & Test Interoperability κ.λπ.

Παράρτημα Α

Η γλώσσα XML

Η γλώσσα XML (eXtensible Markup Language) είναι μια προδιαγραφή γενικής χρήσης για τη δημιουργία *γλωσσών σήμανσης* (markup). Χαρακτηρίζεται *επεκτάσιμη* επειδή επιτρέπει στο χρήστη να ορίζει τα στοιχεία της σήμανσης. Ο σκοπός της XML είναι να βοηθά τα συστήματα πληροφοριών να αποθηκεύουν και να διαμοιράζονται δομημένα δεδομένα.
Η XML είναι μια πρόταση του World Wide Web Consortium. Είναι μια ανοιχτή προδιαγραφή, η οποία ορίζει κανόνες λεκτικής και συντακτικής ανάλυσης των εγγράφων.
Μια και η XML είναι η γλώσσα υλοποίησης των προδιαγραφών IMS Content Packaging και IMS Simple Sequencing, αλλά και του μηχανισμού επέκτασης της πλατφόρμας eclipse, παρουσιάζεται εδώ μια σύντομη περιγραφή της γλώσσας. Αναλυτικότερη περιγραφή υπάρχει στο [16].
Ένα έγγραφο XML έχει δύο επίπεδα ορθότητας:

Ορθή μορφή Ένα ορθά διαμορφωμένο έγγραφο υπακούει στους συντακτικούς κανόνες της XML (συντακτική ορθότητα). Για παράδειγμα, αν υπάρχει κάπου μια ετικέττα εκκίνησης (< >) χωρίς αντίστοιχή ετικέττα τερματισμού (</>), το έγγραφο δεν έχει τη σωστή μορφή και δεν είναι έγγραφο XML.

Εγκυρότητα Ένα έγκυρο έγγραφο υπακούει επιπλέον σε κάποιους σημασιολογικούς κανόνες (σημασιολογική ορθότητα), είτε ορισμένους από το χρήστη, είτε μέσα σε ένα *σχήμα* XML, ή σε έναν *ορισμό τύπου εγγράφου*. Για παράδειγμα, αν ένα έγγραφο περιέχει ένα μη ορισμένο στοιχείο, τότε το έγγραφο δεν είναι έγκυρο.

Ορθή μορφή

Είναι δυνατό σε κάποιες περιπτώσεις να μη γίνεται κανένας έλεγχος εγκυρότητας, οπότε η γλώσσα XML χρησιμοποιείται ως ένα γενικό πλαίσιο για την αποθήκευση οποιασδήποτε ποσότητας κειμένου ή άλλων δεδομένων, των οποίων η δομή μπορεί να αναπαρασταθεί με ένα δέντρο.
Η βασική σύνταξη ενός στοιχείου (element) XML είναι η εξής:
<όνομαΣτοιχείου όνομαΙδιότητας=“τιμή ιδιότητας”>
Περιεχόμενο στοιχείου
</όνομαΣτοιχείου>
Τα δύο στιγμιότυπα του ονόματος του στοιχείου αναφέρονται σαν *ετικέττα εκκίνησης* (start-tag) και *ετικέττα τερματισμού* (end-tag) αντίστοιχα. Το περιεχόμενο του στοιχείου είναι κάποιο κείμενο, που μπορεί να συμπεριλαμβάνει και άλλα στοιχεία XML, σχηματίζοντας έτσι μια *δενδρική* δομή δεδομένων. Κάθε στοιχείο μπορεί να έχει πολλές ιδιότητες. Μπορούν να τοποθετηθούν οπουδήποτε μέσα στη δενδρική δομή σχόλια, που ξεκινούν με <!-- και τελειώνουν σε -->. Οι όροι γονέας, παιδί και αδέλφια χρησιμοποιούνται για να περιγράψουν τις σχέσεις μεταξύ των στοιχείων.
Στο παράδειγμα εγγράφου XML του κώδικα Α.1, η πρώτη γραμμή είναι η δήλωση XML, που ορίζει την έκδοση της γλώσσας (1.0) και την κωδικοποίηση χαρακτήρων που χρησιμοποιείται (ISO-8859-1). Η επόμενη γραμμή (<note>) περιγράφει το *ριζικό στοιχείο* (root element) του εγγράφου, και ουσιαστικά λέει ότι αυτό το έγγραφο είναι ένα σημείωμα. Το ριζικό στοιχείο έχει και μια ιδιότητα time που το περιγράφει, με τιμή 9.30. Οι επόμενες τέσσερις γραμμές περιέχουν τέσσερα *στοιχεία παιδιά* του

ριζικού στοιχείου (to, from, heading, body). Η τελευταία γραμμή (</note>) σηματοδοτεί το τέλος του ριζικού στοιχείου. Θα μπορούσε κάποιο παιδί, εκτός από κείμενο, να έχει και δικά του παιδιά.

Πηγαίος Κώδικας Α.1: Παράδειγμα εγγράφου XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note time="9.30">
  <to>Chris</to>
  <from>Alex</from>
  <heading>Greeting</heading>
  <body>Good morning</body>
</note>
```

Μπορούμε να υποθέσουμε, χωρίς να έχουμε άλλα στοιχεία, ότι το παραπάνω έγγραφο περιγράφει ένα σημείωμα από τον Alex στον Chris. Γενικά, τα έγγραφα της XML έχουν τη χρήσιμη ιδιότητα να περιέχουν αρκετές πληροφορίες ώστε να περιγράφουν τον εαυτό τους.

Εγκυρότητα

Ένα έγκυρο έγγραφο XML είναι ένα "ορθά διαμορφωμένο" έγγραφο XML, σύμφωνα με αυτά που περιγράφτηκαν νωρίτερα, το οποίο επιπλέον υπακούει στους κανόνες ενός ορισμού Document Type Definition (DTD). Στο παράδειγμα Α.2 φαίνεται πώς, με χρήση της δήλωσης DOCTYPE, έχουμε αναφορά σε ένα εξωτερικό αρχείο DTD, το περιεχόμενο του οποίου φαίνεται στον κώδικα Α.3.

Πηγαίος Κώδικας Α.2: Έγγραφο XML με αναφορά σε αρχείο DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note time="9.30">
<!-- .. -->
```

Ο σκοπός του αρχείου DTD είναι να ορίσει τη δομή ενός εγγράφου XML, και αυτό γίνεται με μια λίστα από επιτρεπτά στοιχεία.

Πηγαίος Κώδικας Α.3: Αρχείο DTD

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

Όπου PCDATA (parsed character data) είναι κείμενο το οποίο υφίσταται συντακτική ανάλυση από τον XML parser (υπάρχει και η δυνατότητα κάποια τμήματα κειμένου, τα οποία ονομάζονται CDATA, να εξαιρούνται από τη συντακτική ανάλυση).
Υπάρχει επίσης ένας εναλλακτικός τρόπος περιγραφής της δομής ενός εγγράφου XML, που ονομάζεται XML Schema, και βασίζεται στην ίδια τη γλώσσα XML (παράδειγμα Α.4). Η μέθοδος αυτή, που είναι γνωστή και ως XSD (XML Schema Definition), θεωρείται διάδοχος του DTD επειδή είναι πιο ισχυρή στην περιγραφή γλωσσών XML.

Πηγαίος Κώδικας Α□.4: Αρχείο XSD

```
<xs:element name="note">

  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

</xs:element>
```

Βιβλιογραφία

[1] Μιχάλης Παπαδόπουλος, Διπλωματική εργασία: *Σχεδιασμός και Υλοποίηση ενός Συστήματος Βασισμένου σε Πρότυπα για τη Δυναμική Παρουσίαση Μαθησιακού Περιεχομένου*, Εθνικό Μετσόβιο Πολυτεχνείο, Ιούλιος 2005.

[2] IMS Global Learning Consortium, *IMS Content Packaging Information Model*, έκδοση 1.1.4, Οκτώβριος 2004.

[3] IMS Global Learning Consortium, *IMS Content Packaging Best Practice and Implementation Guide*, έκδοση 1.1.4, Οκτώβριος 2004.

[4] IMS Global Learning Consortium, *IMS Simple Sequencing Information and Behavior Model*, έκδοση 1.0, Μάρτιος 2003.

[5] IMS Global Learning Consortium, *IMS Simple Sequencing Best Practice and Implementation Guide* , έκδοση 1.0, Μάρτιος 2003.

[6] IMS Global Learning Consortium, *IMS Simple Sequencing XML Binding*, έκδοση 1.0, Μάρτιος 2003.

[7] Ανδρέας Παπασαλούρος, *Utilizing High-level Authoring of Simple Sequencing Descriptions*, Ιανουάριος 2009.

[8] The Eclipse Foundation, *Platform Plug-in Developer Guide*, http://help.eclipse.org, Φεβρουάριος 2009.

[9] Jeff McAffer and Jean-Michel Lemieux, *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java™ Applications*. Addison Wesley, 2005.

[10] Εμμανουήλ Στ. Σκορδαλάκης, *Λογισμική Μηχανική*, Εθνικό Μετσόβιο Πολυτεχνείο, 2005.

[11] Microsoft Developer Network, *Microsoft patterns & practices Developer Center*, http://msdn.microsoft.com/en-us/library, Φεβρουάριος 2009.

[12] The Eclipse Foundation, *GEF and Draw2d Plug-in Developer Guide*, http://help.eclipse.org, Φεβρουάριος 2009.

[13] Advanced Distributed Learning Initiative, *ADL Guidelines for Creating Reusable Content with SCORM 2004*, http://www.adlnet.gov, Μάρτιος 2009.

[14] RELOAD Project, *Reusable eLearning Object Authoring & Delivery*, http://www.reload.ac.uk, Απρίλιος 2009.

[15] IMS Global Learning Consortium, *IMS Learning Design Best Practice and Implementation Guide* , έκδοση 1.0, Ιανουάριος 2003.

[16] W3schools.com, *XML*, http://www.w3schools.com/xml, Μάρτιος 2009.