ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Βιβλιοθήκη Συναθροιστικών Εκμαγείων και Εκμαγείων Γενικού Προγραμματισμού με την χρήση Μετασχηματισμών XSL

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΟΥ

Σωτήρη Κ. Νικολάου

Επιβλέπων : Γεώργιος Στασινόπουλος
Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2008

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Βιβλιοθήκη Συναθροιστικών Εκμαγείων και Εκμαγείων Γενικού Προγραμματισμού με την χρήση Μετασχηματισμών XSL

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σωτήρη Κ. Νικολάου

Επιβλέπων : Γεώργιος Στασινόπουλος
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 17η Ιουλίου 2008

……..	……..	……..
Γ. Στασινόπουλος	Μ. Θεολόγου	Β. Λούμος
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

……

Σωτήρης Νικολάου

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

Copyright © Σωτήρης Νικολάου, 2008

Με επιφύλαξη παντός δικαιώματος. All rights reserved.
Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ' ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνεται προς τον συγγραφέα
Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου

Περίληψη

Οι τεχνολογίες XML και XSLT έχουν επεκταθεί με ρυθμούς γεωμετρικής προόδου τα τελευταία χρόνια. Από τότε που εκδόθηκαν από το w3c (world wide web consortium) έχουν βρει τεράστια απήχηση κυρίως σε εφαρμογές διαδικτύου και σε εφαρμογές βάσεων δεδομένων. Ένας από τους κυριότερους λόγους που οι τεχνολογίες αυτές έχουν επεκταθεί σε μεγάλο βαθμό είναι η δυνατότητα επέκτασης τους και η δυνατότητα προσαρμογής εφαρμογών σε διάφορες υπολογιστικές πλατφόρμες. Σκοπός αυτής της διπλωματικής εργασίας είναι η αξιοποίηση των υπολογιστικών δυνατοτήτων της XSLT δημιουργώντας διαφορετικές λειτουργίες οι οποίες θα δρουν σε μια συλλογή από δεδομένα.

Προς την κατεύθυνση αυτή, δημιουργήσαμε μια συλλογή από συνεργαζόμενα εκμαγεία (templates) σε μια μορφή βιβλιοθήκης, η οποία μπορεί να εισαχθεί σε οποιαδήποτε εφαρμογή XSLT με πολύ απλό τρόπο. Όπως είναι γνωστό διαφορετικές δομές δεδομένων μπορούν να αναπαρασταθούν με αρχεία XML. Πολλές φορές όμως παρουσιάζεται η ανάγκη για την επεξεργασία μιας συλλογής δεδομένων για την δημιουργία αναφορών ή την εξαγωγή κάποιων συμπερασμάτων ή την απλή επεξεργασία αυτών των δεδομένων. Για την επεξεργασία αυτή των δεδομένων χρειάζεται η συγγραφή κώδικα, που μερικές φορές μπορεί να είναι μια επίπονη διαδικασία.

Υπάρχουν πολλά πλεονεκτήματα όταν εισάγουμε την βιβλιοθήκη μας σε μια εφαρμογή XSLT. Το κυριότερο είναι ότι μειώνεται σημαντικά ο χρόνος ανάπτυξης της εφαρμογής μας, γιατί όπως αναφέρθηκε και πιο πριν, η συγγραφή κώδικα μπορεί να είναι μια επίπονη και χρονοβόρα διαδικασία. Επίσης η οργάνωση του κώδικα της εφαρμογής μας απλουστεύεται σημαντικά καθώς το κομμάτι κώδικα που εκτελεί την αντίστοιχη λειτουργία της βιβλιοθήκης μας βρίσκεται σε μια μη ορατή διαφορετική τοποθεσία.

Σημαντικό είναι να αναφέρουμε ότι η βιβλιοθήκη παρέχει την δυνατότητα επεκτασιμότητας. Η δομή της βιβλιοθήκης είναι τέτοια που αποτελείται από μικρά ανεξάρτητα κομμάτια. Στα υπάρχοντα κομμάτια μπορούμε να προσθέσουμε και άλλα αυξάνοντας έτσι την λειτουργικότητα της και το πεδίο εφαρμογών στο οποίο μπορεί να χρησιμοποιηθεί αυτή.

Λέξεις κλειδιά: Μετασχηματισμοί XSL, XSLT, extensible stylesheet language, XML, extensible markup language, εκμαγεία, αρχεία XML, αναδρομή, συνάθροιση.

Abstract

XML and XSLT technologies rapidly expanded over the last few years. Since the time that w3c officially recommended them, a lot applications, mostly web and database applications have embedded their use. One of the major reasons that contributed in this significant growth of the XML and XSLT technologies is the possibility of expansion and the adaption of applications that are using these technologies across a variety of computational platforms. The goal of this diploma thesis is to take advantage of XSLT's computational capabilities by creating a series of different operations that can be applied to a collection of data.

Towards this direction, we have composed a collection of collaborative templates in a form of library that can be imported in an XSLT application in a simple way. It is widely known that different data structures can be represented and stored within XML files. Many times though, a need appears to process a data collection to create reports or reach conclusions or even just process and manipulate this data collection. However, this need to process our data, demands that code should be written, and most of the times this task can be a very demanding and time costly one.

Many advantages derive from importing our library in an XSLT application. The most significant of all is the reduction in application development time. As mentioned before, the writing of code can be a very demanding and time costly task. Also, the complexity of the code's structure of our application can be reduced a lot leading to a more simplified and comprehensive code since the part of the code that executes the respective operation of our library is placed in a location different from our application and is not visible to the developer.

It is important to mention that our library offers the capability of expansion. The structure of the library is composed from small modules, each one of them implementing a different operation. In the existing modules we can easily add new ones, expanding it and increasing its functionality and its spectrum of applications that it can be used in.

Keywords: XSL transformations, XSLT, extensible stylesheet language, XML, extensible markup language, templates, XML files, recursion, aggregation.

Ευχαριστίες

Η διπλωματική εργασία αυτή πραγματοποιήθηκε στην Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, υπό την επίβλεψη του Καθηγητή Γεώργιου Στασινόπουλου.

Καταρχήν θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Γεώργιο Στασινόπουλο, για την εποπτεία του κατά την εκπόνηση της εργασίας μου, αλλά και για τη συμβολή του στη διαμόρφωσή μου ως μηχανικού από τις διδασκαλίες του. Για ένα φοιτητή είναι πολύ σημαντικό να συναντήσει καλούς διδασκάλους στην ακαδημαική του πορεία. Γιατί οι διδακαλοί του θα ενεργήσουν καταλυτικά τόσο στην ακαδημαική του πορέια αλλά και την προσωπική του σταδιοδρομία. Θα τον βοηθήσουν να χτίσει τα θεμέλια της μελλοντικής του πορείας και μια γερή προσωπικότητα.

Τέλος, θα ήθελα να ευχαριστήσω το οικογενειακό και φιλικό μου περιβάλλον και κυρίως την πολύ καλή μου φίλη Μαρία, η υποστήριξη των οποίων με βοήθησε τόσο στην εκπόνηση της εργασίας όσο και στην ολοκλήρωση των προπτυχιακών μου σπουδών.

Πίνακας Περιεχομένων

1. Περιγραφή της XML και της XSLT 15
   α. XML (Extensible Markup Language) 15
   β. XSLT (Extensible Stylesheet Language Transformations) 19
   γ. Εκτέλεση ενός Stylesheet 21
2. Βασικές έννοιες και στοιχεία 25
   α. Παράμετροι (parameters) και μεταβλητές (variables): 25
   γ. Μαθηματικές λειτουργίες με αναδρομή 32
3. Γενικός Προγραμματισμός(Generic Programming) 37
   α. Γενικά 37
      α.1 Επέκταση περιεχομένων καθολικών μεταβλητών 37
   β. Συνάθροιση (Aggregation) 39
      β1. Αλγόριθμος κλήσης αναδρομικών templates 46
4. Η Βιβλιοθήκη Συνάθροισης 49
   α. Γενικά 49
   β. Δομή της Βιβλιοθήκης 49
   γ. Συναθροιστικές Λειτουργίες 51
      γ.1 Named Templates 51
      γ.2 Generic Template 53
   δ. Εισαγωγή βιβλιοθήκης σε εφαρμογές XSLT 54
4. Άλλα θέματα 57
5. Παραρτήματα 63
   α. Παράρτημα Α – Κώδικας Βιβλιοθήκης 63
      α.1 Κώδικας Βιβλιοθήκης – Μέρος Α 64
      α.2 Κώδικας Βιβλιοθήκης – Μέρος Β (generic templates) 70
   β. Παράρτημα Β – Κώδικας Τελικών Παραδειγμάτων 73
6. Βιβλιογραφία 77

1. Περιγραφή της XML και της XSLT

α. XML (Extensible Markup Language)

Η γλώσσα XML είναι μια απλή περιγραφική (μετα)γλώσσα δεδομένων γενικού σκοπού η οποία ορίστηκε από το World Wide Web Consortium (W3C). Ανήκει στην κατηγορία των επεκτάσιμων γλωσσών (Extensible Languages) γιατί επιτρέπει στον χρήστη να ορίσει τα δικά του στοιχεία (elements) δίνοντας έτσι νόημα στα δεδομένα. Ο σημαντικότερος σκοπός της XML είναι ο διαμοιρασμός και αναπαράσταση δομημένης πληροφορίας ανάμεσα σε διαφορετικά υπολογιστικά συστήματα διαμέσου του διαδικτύου(internet). Η XML είναι ένα σύνολο από κανόνες και οδηγίες για την περιγραφή της δομημένης πληροφορίας σε απλό κείμενο σε αντίθεση με το δυαδικό τρόπο αναπαράστασης. Παρόλα αυτά, η XML ξεφεύγει από τις τεχνικές προδιαγραφές της. Από το 1998, όταν και προτυποποιήθηκε είναι η κινητήρια δύναμη πίσω από μεγάλο αριθμό προτύπων που διαμορφώνουν μια θεμελιώδη αλλαγή στην τεχνολογία λογισμικού.

Η XML επηρεάζει τον τρόπο με τον οποίο αναπτύσσονται οι διάφορες εφαρμογές λογισμικού και τον τρόπο με τον οποίο αντιλαμβανόμαστε τα κατανεμημένα συστήματα. Γύρω από την XML υπάρχει μια οικογένεια προτύπων, τεχνολογιών και πρωτοκόλλων, η οποία έδωσε την δυνατότητα της ανταλλαγής πληροφορίας στον παγκόσμιο ιστό (World Wide Web) και της ανάπτυξης της τηλεπικοινωνιακής υποδομής.

Η XML αντλεί την περισσότερη από την δύναμη της σε συνδυασμό με το διαδίκτυο (web). Το διαδίκτυο παρέχει μια συλλογή από πρωτόκολλα για μεταφορά δεδομένων και η XML αναπαριστά τον τρόπο με τον οποίο ορίζονται αυτά τα δεδομένα. Η άμεση επίδραση της XML είναι στον τρόπο με τον οποίο παρατηρεί κάποιος μια επιχείρηση. Αντίθετα με το στενά πλεγμένο δίκτυο από servers, η επιχείρηση τώρα μοιάζει να καλύπτει όχι μόνο το παραδοσιακό δίκτυο μιας επιχείρησης αλλά σχεδόν όλο το διαδίκτυο (web).

Η XML είναι απλή. Με τεχνικούς όρους, είναι μια γλώσσα για την δημιουργία άλλων γλωσσών η οποία βασίζεται στην εισαγωγή ετικετών(tags) για

να περιγράψει δεδομένα. Η XML είναι ένας συνδυασμός από ετικέτες και περιεχόμενο στο οποίο οι ετικέτες δίνουν νόημα. Το πιο κάτω παράδειγμα δείχνει μια απλή XML περιγραφή δεδομένων ενός φοιτητή.

```
<student>
        <name>Σωτήρης Νικολάου</name>
        <phones>
                <home>2107488990</home>
                <mobile>6932248159</mobile>
        </phones>
        <address>Χλόης 40 Ζωγράφου, Αθήνα</address>
        <entry-year>2003</entry-year>
        <email>soterisnicolaou@yahoo.com</email>
</student>
```

Στο πιο πάνω παράδειγμα τα δεδομένα περικλείονται σε ετικέτες. Για παράδειγμα το όνομα (name) ορίζεται στο στοιχείο (element) <name> . Το κάθε στοιχείο έχει μια ετικέτα έναρξης και μια ετικέτα τέλους, στο παράδειγμα η ετικέτα έναρξης είναι η <name> και η ετικέτα τέλους η </name>. Η αναπαράσταση δεδομένων με στοιχεία δεν είναι ο μοναδικός τρόπος αναπαράστασης. Τα δεδομένα μπορούν να αναπαρασταθούν επίσης με ιδιότητες (attributes). Στο παράδειγμα που ακολουθεί κάποια από τα στοιχεία του αρχικού παραδείγματος(name, home, mobile) έχουν μετατραπεί σε ιδιότητες.

```
<student name="Σωτήρης Νικολάου">
        <phones home="2107488990" mobile="6932248159"/>
        <address>Χλόης 40 Ζωγράφου, Αθήνα</address>
        <entry-year>2003</entry-year>
        <email>soterisnicolaou@yahoo.com</email>
</student>
```

Η σημασία και χρήση των ιδιοτήτων είναι επιλογή του χρήστη και της εφαρμογής που επεξεργάζεται το XML. Και στα δυο παραδείγματα τα δεδομένα είναι τα ίδια, αυτό που διαφέρει όμως είναι η μορφή. Μερικά σημαντικά σημεία για τον ορισμό δεδομένων είναι τα εξής:

- Η XML επιτρέπει σε δεδομένα να αποθηκευτούν είτε σε στοιχεία (elements) είτε σε ιδιότητες (attributes).
- Τα στοιχεία και οι ιδιότητες μπορούν να ονομαστούν με τέτοιο τρόπο ώστε να δίνουν νόημα στα δεδομένα.
- Ετικέτες έναρξης και ετικέτες τέλους ορίζουν στοιχεία τα οποία είναι η βάση για την XML δενδρική δομή αναπαράστασης αρχείων.

- Τα στοιχεία μπορούν να περιέχουν δεδομένα κειμένου ή άλλα στοιχεία (υποστοιχεία).

Η XML έχει αντίκτυπο σε ένα μεγάλο φάσμα από εφαρμογές. Παρακάτω συνοψίζονται μερικοί από τους παράγοντες που έχουν συμβάλει στην αφομοίωση της από ένα μεγάλο σύνολο οργανισμών και ατόμων:

- Τα αρχεία XML είναι αναγνώσιμα από τον άνθρωπο. Η XML σχεδιάστηκε ως κείμενο, ώστε στην χειρότερη περίπτωση κάποιος να μπορεί να το διαβάσει με μια εφαρμογή επεξεργασίας κειμένου και να αντιληφθεί τα δεδομένα του. Αυτό δεν μπορεί να το κάνει με δυαδικές μορφές αναπαράστασης δεδομένων.
- Ευρεία υποστήριξη από την βιομηχανία πληροφορικής. Υπάρχει μεγάλος αριθμός εργαλείων και εφαρμογών τα οποία παρέχονται μέσα από εφαρμογές πλοήγησης (web browsers), συστήματα βάσεων δεδομένων, λειτουργικά συστήματα τα οποία κάνουν ευκολότερη και φθηνότερη για επιχειρήσεις την εισαγωγή/εξαγωγή (import/export) δεδομένων.
- Μερικά από τα ευρέως χρησιμοποιούμενα σχεσιακά συστήματα βάσεων δεδομένων (RDBMS) έχουν την ενσωματωμένη δυνατότητα να διαβάζουν και να δημιουργούν δεδομένα XML.
- Μια μεγάλη οικογένεια από τεχνολογίες που υποστηρίζουν την XML είναι διαθέσιμη για την μεταγλώττιση και μετασχηματισμό XML δεδομένων για προβολή σε ιστοσελίδες και δημιουργία αναφορών.

Επιπλέον των παραγόντων και πλεονεκτημάτων που αναφέρθηκαν πιο πάνω, σημαντικό ρόλο στην επιτυχία και διάδοση της XML έχουν αυτά που δεν ορίζει (design by omission). Τρια σχεδιαστικά στοιχεία της XML τα οποία παραλείφθηκαν συμβάλουν στην επιτυχία της XML:

- Δεν ορίζεται τρόπος προβολής. Αντίθετα με την HTML, η XML δεν βγάζει συμπεράσματα για τον τρόπο με τον οποίο πρέπει να παρασταθούν οι διάφορες ετικέτες σε ένα πρόγραμμα περιήγησης. Την δυνατότητα αυτή παρέχουν βοηθητικές τεχνολογίες όπως τα stylesheets.
- Δεν υπάρχει ενσωματωμένη τυποποίηση δεδομένων. Τα DTDs και XML Schema παρέχουν την υποστήριξη για τον ορισμό δομής και τύπο δεδομένων τα οποία σχετίζονται με ένα XML αρχείο.

- Δεν υπάρχει καθορισμένος τρόπος μεταφοράς. Το πρότυπο XML δεν ορίζει πως θα μεταφερθούν τα XML δεδομένα στο διαδίκτυο. Το γεγονός αυτό ώθησε στην καινοτομία για την μεταφορά XML δεδομένων μέσω HTTP, FTP ή SMTP

Με άλλα λόγια οι παραλήψεις αυτές ηθελημένα περιορίζουν τις δυνατότητες και προσδοκίες της XML για να μεγιστοποιήσουν την αλληλεπίδραση με άλλες τεχνολογίες.

β. XSLT (Extensible Stylesheet Language Transformations)

Η γλώσσα XSLT είναι μια απλή γλώσσα η οποία επιτρέπει τον μετασχηματισμό (transform) αρχείων XML σε νέα αρχεία XML, HTML, XHTML ή και απλά αρχεία κειμένου. Η XPath (XML Path Language) είναι μια συνοδευτική τεχνολογία της XSLT η οποία βοηθάει στην αναγνώριση και στην εύρεση κόμβων (nodes) σε αρχεία XML. Υπάρχουν αρκετοί τρόποι να χρησιμοποιήσει κανείς την XSLT μερικοί από τους οποίους είναι:

- Ο μετασχηματισμός ενός αρχείου XML σε αρχείο HTML για να προβληθεί σε ένα πρόγραμμα περιήγησης web (Θα δοθούν παραδείγματα σε επόμενο κεφάλαιο).
- Η εξόρυξη κειμένου από ένα αρχείο XML για χρήση σε μη-XML εφαρμογές ή περιβάλλοντα.
- Η δημιουργία για παράδειγμα ενός ελληνικού αρχείου συλλέγοντας όλα τα κείμενα τα οποία είναι γραμμένα σε ελληνική γλώσσα σε ένα πολυγλωσσικό XML αρχείο.

Οι τρόποι που αναφέρθηκαν πιο πάνω είναι ενδεικτικοί. Υπάρχουν πολλοί άλλοι τρόποι με τους οποίους θα μπορούσε κανείς να χρησιμοποιήσει την XSLT.

Κάθε αρχείο XSL είναι και αυτό γραμμένο συμβατά με την σύνταξη της XML και περιλαμβάνει ιδιαίτερες ετικέτες (xsl tags) οι οποίες ορίζουν και τις επιπλέον λειτουργικότητες. Επιπλέον η XSL κάνει την υπόθεση ότι επεξεργαζόμαστε το αρχείο XML σαν δέντρο για να παράγουμε στην έξοδο απλό κείμενο, HTML, XML ή ακόμα και PDF.

Παρακάτω δίνονται κάποια απλά παραδείγματα τα οποία δίνουν μια εισαγωγική εικόνα για το πώς μπορεί να χρησιμοποιηθεί η XML σε συνδυασμό με την XSL. Το stylesheet που ακολουθεί επεξεργάζεται το αρχικό παράδειγμα (κώδικας 2.1) που δώσαμε πιο πριν.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="text"/>
	<xsl:template match="/">
		<xsl:apply-templates select="student"/>
	</xsl:template>
	<xsl:template match="student">
		<xsl:text>Name: </xsl:text><xsl:value-of select="name"/>
		<xsl:text>&#10;Phones: </xsl:text>
		<xsl:apply-templates select="phones"/>
		<xsl:text>&#10;Address: </xsl:text><xsl:value-of select="address"/>
		<xsl:text>&#10;Entry year in NTUA: </xsl:text><xsl:value-of select="entry-
year"/>
		<xsl:text>&#10;Email: </xsl:text><xsl:value-of select="email"/>
	</xsl:template>
	<xsl:template match="phones">
		<xsl:text>&#10;    Home: </xsl:text><xsl:value-of select="home"/>
		<xsl:text>&#10;    Mobile: </xsl:text><xsl:value-of select="mobile"/>
	</xsl:template>
</xsl:stylesheet>
```

Όταν εφαρμόσουμε αυτό το xsl στον κώδικα 2.1 που είναι σε ένα αρχείο XML θα πάρουμε στην έξοδο το ακόλουθο:

```
Name: Σωτήρης Νικολάου
Phones:
   Home: 2107488990
   Mobile: 6932248159
Address: Χλόης 40 Ζωγράφου, Αθήνα
Entry year in NTUA: 2003
Email: soterisnicolaou@yahoo.com
```

Στο συγκεκριμένο παράδειγμα η έξοδος είναι ένα κείμενο. Θα μπορούσε να είναι ένα νέο αρχείο XML ή HTML. Αυτό καθορίζεται στο στοιχείο <xsl:output method="text"/>. Στο παράδειγμα αυτό η έξοδος καθορίζεται ως *text*. Μπορούμε να παρατηρήσουμε ότι όλος ο κώδικας περικλείεται στο στοιχείο <xsl:stylesheet> το οποίο ανοίγει στην πρώτη γραμμή και κλείνει στην τελευταία με την ετικέτα </xsl:stylesheet>. Μέσα σε αυτό το στοιχείο ορίζονται τα διάφορα εκμαγεία (templates) και στο παράδειγμα ορίζονται τρία templates. Κάθε template έχει μια ιδιότητα match με την οποία δηλώνεται επί ποιού υποδέντρου του αρχείου XML θα εφαρμοστεί το template αυτό. Μέσα στο template μπορούν να παρατίθενται οποιαδήποτε στοιχεία XML (ανεξαρτήτως του αρχικού XML το οποίο κάλεσε το συγκεκριμένο XSL. Εκτός από αυτά υπάρχουν και στοιχεία/εντολές του κατ' εξοχήν XSL (τα εισαγόμενα με το namespace xsl:) τα οποία αναφέρονται και συλλέγουν στοιχεία από το αρχικό XML όπως για παράδειγμα το <xsl:value-of select="name"/> το οποίο ανατρέχει στο στοιχείο *name* και γράφει στην έξοδο τα περιεχόμενα του.

γ. Εκτέλεση ενός Stylesheet

Πριν να εκτελεστεί ένα stylesheet πρέπει πρώτα να διαβαστεί από τον XSLT processor. Δεν έχει σημασία το πώς ο processor θα αποθηκεύσει τις πληροφορίες στην μνήμη. Στο stylesheet του παραδείγματος περιέχονται τέσσερα στοιχεία (elements): ένα <xsl:output> και τρία </xsl:template> τα οποία ορίζουν πως τα διάφορα τμήματα του αρχείου XML θα μετασχηματιστούν. Όταν ο XSLT processor διαβάσει το stylesheet πρέπει να διαβάσει μετά και το αρχείο το οποίο ορίστηκε για να μετασχηματίσει. Ο XSLT processor φτιάχνει μια δενδρική δομή για το πηγαίο αρχείο XML και αυτή την δενδρική δομή πρέπει να έχει ο προγραμματιστής του stylesheet όταν προγραμματίζει. Όταν διαβαστούν τα δυο αρχεία (το XSL και το XML) αρχίζει η επεξεργασία τους όπως παρακάτω:

- Υπάρχουν κόμβοι για επεξεργασία? Οι κόμβοι για επεξεργασία αντιπροσωπεύονται από το λεγόμενο *context*. Αρχικά το context είναι η ρίζα του XML αρχείου αλλά αλλάζει καθώς προχωράει η επεξεργασία.
- Όσο υπάρχουν κόμβοι στο context κάνε τα παρακάτω:
- Πάρε τον επόμενο κόμβο από το context. Υπάρχουν <xsl:template> τα οποία να ταιριάζουν με τον κόμβο αυτό?
- Αν ένα ή περισσότερα <xsl:templates> ταιριάζουν ο XSLT processor επιλέγει το καταλληλότερο (το καταλληλότερο είναι το πιο συγκεκριμένο) και το επεξεργάζεται. Αν δεν υπάρχουν <xsl:template> τα οποία ταιριάζουν τότε ο XSLT processor εφαρμόζει κάποια ενσωματωμένα templates (built-in templates).

Το παραπάνω μοντέλο είναι ένα αναδρομικό μοντέλο. Επεξεργαζόμαστε τον τρέχων κόμβο επιλέγοντας το κατάλληλο <xsl:template> το οποίο με την σειρά του μπορεί να καλέσει άλλα <xsl:template> και τα οποία με την σειρά τους μπορούν να καλέσουν άλλα <xsl:template> κ.ο.κ.

Εφαρμόζοντας τους παραπάνω απλούς κανόνες στο δικό μας παράδειγμα μπορούμε να διακρίνουμε τα ακόλουθα βήματα, τα οποία μπορούν να βοηθήσουν στην αντίληψη της διαδικασίας με την οποία εκτελείται ένα stylesheet:

1. Το XSLT stylesheet αναλύεται και μετατρέπεται σε δεντρική δομή.

2. Το XML αρχείο επίσης αναλύεται και μετατρέπεται σε δεντρική δομή. Δεν έχει ιδιαίτερη σημασία το πώς μοιάζει αυτό το δέντρο και ποιος είναι ο τρόπος λειτουργίας του
3. Ο XSLT processor αφού αναλύσει τα δυο αρχεία βρίσκεται στην ρίζα (root) του αρχείου XML. Αυτό είναι το προκαθορισμένο context.
4. Στο παράδειγμα μας υπάρχει ένα <xsl:template> το οποίο ταιριάζει με το context.

```
<xsl:template match="/">
        <xsl:apply-templates select="student"/>
</xsl:template>
```

   Μια απλή κάθετος γραμμή (/) είναι μια *XPath expression* η οποία σημαίνει η ρίζα του αρχείου του XML.
5. Τώρα η επεξεργασία αρχίζει μέσα στο <xsl:template>. Η μόνη εντολή είναι η <xsl:apply-templates> στους στα στοιχεία που ταιριάζουν με το *"student"* στον τρέχον context – δηλαδή τη ρίζα του αρχείου. Το τρέχον context μέσα στο template ορίζεται από την ιδιότητα match του <xsl:template> στοιχείου. Δηλαδή, ο XSLT processor ψάχνει για "student" στοιχεία μέσα στη ρίζα του αρχείου.
   Επειδή υπάρχει ένα "student" στοιχείο μέσα στην ρίζα του αρχείου, ο XSLT processor πρέπει να το επεξεργαστεί. (Αν περισσότερα από ένα στοιχεία ταυτίζονται με το τρέχον context ο XSLT processor πρέπει να τα επεξεργαστεί με την σειρά με την οποία αυτά εμφανίζονται). Ο XSLT processor τώρα εφαρμόζει το <xsl:template> για το στοιχείο "student" δηλαδή το δεύτερο template του παραδείγματος:

```
<xsl:template match="student">
        <xsl:text>Name: </xsl:text><xsl:value-of select="name"/>
        <xsl:text>&#10;Phones: </xsl:text>
        <xsl:apply-templates select="phones"/>
        <xsl:text>&#10;Address: </xsl:text><xsl:value-of select="address"/>
        <xsl:text>&#10;Entry year in NTUA: </xsl:text><xsl:value-of select="entry-year"/>
        <xsl:text>&#10;Email: </xsl:text><xsl:value-of select="email"/>
</xsl:template>
```

   Μέσα σε αυτό το template το τρέχον context είναι πια το */student*. Υπάρχει μια σειρά από εντολές οι οποίες εφαρμόζονται στα υποστοιχεία του στοιχείου "student". Για παράδειγμα η 1η εντολή <xsl:text>Name: </xsl:text> στέλνει στην ροή εξόδου (output stream) το κείμενο "Name: ". Η επόμενη

εντολή <xsl:value-of select="name"/> στέλνει επίσης στην ροή εξόδου την τιμή των στοιχείων "name" που βρίσκονται στο σύνολο των κόμβων που ορίζονται από το context */student*. Εδώ υπάρχει ένα στοιχείο "name" οπότε η συνολική έξοδος της εντολής αυτής θα είναι: 'Name: Σωτήρης Νικολάου'

6. Η εντολή στην τρίτη γραμμή <xsl:apply-templates select="phones"/> εφαρμόζει το τρίτο template του παραδείγματος στους κόμβους του τρέχον context /student που ταιριάζουν με "phones".

```
<xsl:template match="phones">
        <xsl:text>&#10;    Home: </xsl:text><xsl:value-of select="home"/>
        <xsl:text>&#10;    Mobile: </xsl:text><xsl:value-of select="mobile"/>
</xsl:template>
```

   Επειδή υπάρχει ένα στοιχείο "phones" σαν υποστοιχείο του "student" ο XSLT processor πρέπει να το εκτελέσει. Όταν μπει πια στο template αυτό το context αλλάζει από /student σε /student/phones και εκτελούνται οι εντολές που ορίζονται από αυτό το template.
7. Όταν εκτελεστούν οι εντολές του 3ου template ο XSLT processor επιστρέφει στο 2ο template και στο σημείο όπου κλήθηκε το 3ο template και συνεχίζει την εκτέλεση έχοντας πάλι σαν context το /student.
8. Όταν τελειώσει και η εκτέλεση των εντολών του 2ου template επιστέφει ο XSLT processor στο template για το root στοιχείο. Στο σημείο αυτό έχουν επεξεργασθεί όλα τα "student" στοιχεία και η εκτέλεση του αρχικού template τελειώνει.
9. Στο σημείο αυτό δεν υπάρχουν άλλα στοιχεία στο τρέχον context /, οπότε και η εκτέλεση του stylesheet τελειώνει.

Το παραπάνω παράδειγμα είναι ένα μικρό δείγμα του τι μπορεί κανείς να κάνει με την XML και την XSLT. Στην συνέχεια θα δοθούν εκτενέστερα παραδείγματα τα οποία θα δώσουν μια πιο ολοκληρωμένη εικόνα των δυνατοτήτων της.

2. Βασικές έννοιες και στοιχεία

Στο κεφάλαιο αυτό παρουσιάζονται κάποια από τα βασικά στοιχεία τα οποία είναι απαραίτητα στην συνέχεια. Μερικά από αυτά είναι οι παράμετροι, οι μεταβλητές, η συνάρτηση *document()* και άλλα

α. Παράμετροι (parameters) και μεταβλητές (variables):

Η XSLT προσφέρει αρκετούς τρόπους για να δεσμεύσει κάποιος μια τιμή σε ένα όνομα ώστε να μπορεί μελλοντικά να αναφέρεται αυτή η τιμή με το όνομα της σε αρκετά σημεία σε ένα stylesheet. Μια μεταβλητή ορίζεται με το στοιχείο *<xsl:variable>* δεσμεύει ένα όνομα σε μια αμετάβλητη τιμή όταν αρχικοποιείται. Αντίθετα, μια παράμετρος ορίζεται με το στοιχείο *<xsl:param>* και δεσμεύει ένα όνομα σε μια default τιμή αλλά η τιμή της μπορεί να αλλάξει. Μπορούμε δηλαδή να ορίσουμε μια default τιμή για μια μεταβλητή αλλά να την αλλάξουμε όταν περάσουμε την παράμετρο αυτή σε ένα stylesheet ή σε ένα template. Το στοιχείο *<xsl:with-param>* επιτρέπει την εφαρμογή ή κλήση (apply ή call) templates από ένα template με μια νέα τιμή για μια ή περισσότερες παραμέτρους σαν να καλούσαμε μια μέθοδο ή συνάρτηση με ορίσματα.

Οι μεταβλητές στην XSLT έχουν περιορισμένες δυνατότητες. Στις γλώσσες προγραμματισμού οι μεταβλητές μπορούν να αναθέτονται πολλές φορές σε ένα πρόγραμμα, γεγονός που δεν ισχύει στην XSLT. Γενικά, οι μεταβλητές ορίζονται μια φορά και μετά αναφέρονται στο stylesheet μετά όσες φορές χρειάζεται. Στις παραμέτρους αντίστοιχα η αρχική τιμή τους αλλάζει αν το θελήσουμε όταν περνούμε την παράμετρο σε ένα stylesheet ή ένα template. Με αυτό τον τρόπο χρησιμοποίησης των μεταβλητών της XSLT, μοιάζουν περισσότερο με constants σε γλώσσες προγραμματισμού.

Τα στοιχεία *<xsl:param>* και *<xsl:variable>* μπορούν να χρησιμοποιηθούν σε όλο το stylesheet είτε σαν καθολικά στοιχεία ή και τοπικά μέσα σε templates. Αν μια μεταβλητή είναι καθολική τότε η εμβέλεια της είναι όλο το stylesheet ενώ αν είναι τοπική τότε η εμβέλεια της περιορίζεται στο template στο οποίο αυτή

ορίζεται. Το στοιχείο <*xsl:with-param*> είναι δυνατό να εμφανίζεται μόνο ως υποστοιχείο του στοιχείου <*xsl:apply-templates*> ή του στοιχείου <*xsl:call-template*>.

Για να γίνει κατανοητή η χρήση των παραμέτρων και μεταβλητών θα δοθούν παρακάτω δυο παραδείγματα ένα για παραμέτρους και ένα για μεταβλητές.

Παράδειγμα παραμέτρων

Έστω έχουμε το παρακάτω αρχείο XML στο οποίο θέλουμε να προσθέσουμε μια ιδιότητα attribute στο στοιχείο "lab" με το όνομα "lab-grade" το οποίο θα αντιπροσωπεύει τον μέσο όρο των τριών εργαστηρίων "lab1", "lab2" και "lab3".

```
<!—- course.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<courses>
	<course name="Fysiki 1">
		<semester>1</semester>
		<labs>
			<lab1>8</lab1>
			<lab2>10</lab2>
			<lab3>7</lab3>
		</labs>
		<grade>7</grade>
		<extra-assignment done="yes"/>
	</course>
</courses>
```

Στο stylesheet θα αντιγράψουμε τα στοιχεία του αρχικού XML με το στοιχείο <*xsl:copy*> και θα χρησιμοποιήσουμε ένα template το οποίο θα δέχεται τρεις παραμέτρους, μια για κάθε βαθμό.

Στην συνέχεια το template αυτό, θα φτιάχνει ένα νέο attribute με το στοιχείο <*xsl:attribute*> και θα δίνει σαν τιμή στο attribute αυτό τον μέσο όρο των τριών παραμέτρων.

```
<!—- add-attribute-to-course.xsl -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="xml" indent="yes"/>

	<xsl:template match="/courses">
		<xsl:copy>
			<xsl:apply-templates select="course"/>
		</xsl:copy>
	</xsl:template>
```

```
<!—- add-attribute-to-course (συνέχεια).xsl -->

        <xsl:template match="course">
                <xsl:copy>
                        <xsl:apply-templates select="labs|node()|@*">
                                <xsl:with-param name="grade1" select="labs/lab1"/>
                                <xsl:with-param name="grade2" select="labs/lab2"/>
                                <xsl:with-param name="grade3" select="labs/lab3"/>
                        </xsl:apply-templates>
                </xsl:copy>
        </xsl:template>

        <xsl:template match="node()|@*">
                <xsl:copy>
                        <xsl:apply-templates select="node()|@*"/>
                </xsl:copy>
        </xsl:template>

        <xsl:template match="labs">
                <xsl:param name="grade1"/>
                <xsl:param name="grade2"/>
                <xsl:param name="grade3"/>
                <xsl:copy>
                        <xsl:attribute name="lab-grade">
                                <xsl:value-of select="($grade1 +$grade2 + $grade3) div 3"/>
                        </xsl:attribute>
                        <xsl:apply-templates select="node()|@*"/>
                </xsl:copy>
        </xsl:template>

</xsl:stylesheet>
```

Εφαρμόζοντας το template αυτό στο πιο πάνω αρχείο XML θα πάρουμε στην έξοδο ένα νέο αρχείο XML το οποίο θα έχει στο στοιχείο “labs” μια επιπλέον ιδιότητα “lab-grade” με τον μέσο όρο των τριών εργαστηρίων:

```
<?xml version="1.0"?>
<courses>
        <course name="Fysiki 1">
                <semester>1</semester>
                <labs lab-grade="8.33">
                        <lab1>8</lab1>
                        <lab2>10</lab2>
                        <lab3>7</lab3>
                </labs>
                <grade>7</grade>
                <extra-assignment done="yes"/>
        </course>
</courses>
```

Παρατηρώντας το πιο πάνω stylesheet, βλέπουμε ότι ο σκοπός του είναι να αντιγράψει το αρχικό αρχείο XML σε ένα νέο αρχείο με κάποιες δομικές αλλαγές. Για τις αλλαγές αυτές φροντίζει το template το οποίο ταιριάζει στο context “labs”. Όταν καλείται αυτό το template, πρέπει να περαστούν σε αυτό τρεις παράμετροι. Αυτό επιτυγχάνεται με το στοιχείο <xsl:with-param> το οποίο αντιστοιχεί την κάθε παράμετρο που αντιστοιχεί στην ιδιότητα *name* με μια τιμή η οποία ορίζεται με την ιδιότητα *select.* Εδώ, αντιστοιχούνται στους παραμέτρους *grade1, grade2* και *grade3* τα στοιχεία “lab1”, “lab2” και “lab3” και στη συνέχεια προκύπτει ο μέσος όρος ο οποίος τοποθετείται σαν μια νέα ιδιότητα στο στοιχείο labs.

Παράδειγμα μεταβλητών

Οι μεταβλητές μπορούν να είναι καθολικές, δηλαδή να είναι υποστοιχεία(παιδιά) του stylesheet ή να είναι τοπικές και να ορίζονται σε κάποιο template. Στο πιο κάτω παράδειγμα ορίζεται μια καθολική μεταβλητή *bonus* με τιμή η οποία όπως αναφέρθηκε και πιο πριν δεν μπορεί να αλλάξει και έχει τιμή '1.5'. Η μεταβλητή αυτή αντιστοιχεί στον επιπλέον βαθμό που θα έχει ένας φοιτητής στο μάθημα εάν παραδώσει την προαιρετική εργασία και προστίθεται στον τελικό βαθμό.

```
<!-- add-bonus-to-course.xsl -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

	<xsl:output method="xml" indent="yes"/>
	<xsl:variable name="bonus" select="1.5"/>

	<xsl:template match="/courses">
		<xsl:copy>
			<xsl:apply-templates select="course"/>
		</xsl:copy>
	</xsl:template>

	<xsl:template match="course">
		<xsl:copy>
			<xsl:attribute name="name">
				<xsl:value-of select="@name"/>
			</xsl:attribute>
			<xsl:copy-of select="semester"/>
			<xsl:choose>
				<xsl:when test="extra-assignment/@done = 'yes'">
					<xsl:if test="grade &gt; 8.5">
						<grade>10</grade>
					</xsl:if>
					<xsl:if test="grade &lt; 8.5">
					<grade><xsl:value-of select="$bonus+grade"/></grade>
					</xsl:if>
				</xsl:when>
				<xsl:otherwise>
					<xsl:copy-of select="grade"/>
				</xsl:otherwise>
			</xsl:choose>
		</xsl:copy>
	</xsl:template>
</xsl:stylesheet>
```

Το πιο πάνω stylesheet έχει πάλι στόχο να αντιγράψει ένα αρχείο XML σε ένα νέο το οποίο θα περιέχει τον βαθμό σε ένα μάθημα. Προτού όμως γίνει η αντιγραφή του στοιχείου "grade" γίνεται έλεγχος εάν ο φοιτητής παράδωσε την προαιρετική εργασία με το στοιχείο <xsl:when test="extra-assignment/@done = 'yes'"> και αν έχει παραδώσει ο βαθμός του φοιτητή αυξάνεται κατά την τιμή που ορίζει η μεταβλητή *bonus* (στην προκειμένη κατά 1.5 μονάδα) όμως ο συνολικός βαθμός δεν μπορεί να ξεπεράσει το δέκα. Η χρήση της μεταβλητής αυτής στο συγκεκριμένο παράδειγμα είναι αρκετά απλή, αλλά όπως θα φανεί και στην συνέχεια μπορεί να χρησιμοποιηθεί σε πολύ πολυπλοκότερες περιπτώσεις

β. Η συνάρτηση document()

Μια από τις σημαντικότερες δυνατότητες της XSLT είναι η συνάρτηση *document().* Η συνάρτηση *document()* επιτρέπει την χρήση ενός τμήματος ενός αρχείου XML το οποίο περιγράφεται με μια *XPath expression* σαν ένα *URI.* Με άλλα λόγια, μπορούμε να προσπελάσουμε στοιχεία ενός αρχείου XML διαφορετικού από αυτό στο οποίο επιδρά ο XSLT μετασχηματισμός, να χρησιμοποιήσουμε κάποια τμήματα του σαν URLs ή ονόματα αρχείων (filenames), να τα ανοίξουμε και να τα αναλύσουμε (parse) και να εκτελέσουμε λειτουργίες που ορίζονται στο stylesheet σε ένα συνδυασμό από αρχεία XML.

Η *document()* είναι χρήσιμη όταν θέλουμε να ορίσουμε όψεις πολλών αρχείων XML ή όταν θέλουμε να δούμε συνοπτικά κάποια στοιχεία τα οποία είναι αποθηκευμένα σε διαφορετικά αρχεία. Στο παρακάτω παράδειγμα θα κάνουμε την υπόθεση ότι έχουμε μια συλλογή από αρχεία XML το οποίο το κάθε ένα αντιστοιχεί σε έναν φοιτητή και μέσα είναι αποθηκευμένα προσωπικά του στοιχεία, και πληροφορίες για μαθήματα τα οποία έδωσε. Ένα τέτοιο αρχείο έχει την παρακάτω μορφή:

```
<!-- 03103618.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<student id="03103618">
        <student-info>
                <name>Sotiris Nicolaou</name>
                <phones>
                        <home>2107488990</home>
                        <mobile>6932248159</mobile>
                </phones>
                <address>Chloes 40, Zografou, Athens</address>
                <entry-year>2003</entry-year>
                <passport-no>E047239</passport-no>
                <email>soterisnicolaou@yahoo.com</email>
        </student-info>

        <courses>
                <course id="3.5.63.8">
                        <name>Diadiktyo kai Efarmoges</name>
                        <semester>8</semester>
                        <grade>10</grade>
                </course>
                …
                <course id="3.4.26.7">
                        <name>Vaseis Dedomenwn</name>
                        <semester>7</semester>
                        <grade>9</grade>
                </course>
        </courses>
</student>
```

Όπως αναφέρθηκε και πιο πάνω, στο παράδειγμα μας έχουμε αρκετά τέτοια αρχεία τα οποία θέλουμε να δούμε συνοπτικά συνδυάζοντας τα σε μια αναφορά. Το πρώτο πράγμα που έχουμε να κάνουμε είναι να ορίσουμε ένα πρωτεύον αρχείο

(master document) στο οποίο θα είναι αποθηκευμένες οι αναφορές (references) για τα πιο πάνω αρχεία τα οποία θέλουμε να συμπεριλάβουμε στην αναφορά (report). Ένα τέτοιο αρχείο έχει την παρακάτω μορφή:

```
<!-- report.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<report>
        <title>Student Course Information</title>
        <stud-file filename="03103602.xml"/>
        <stud-file filename="03103608.xml"/>
        <stud-file filename="03103618.xml"/>
</report>
```

Στο πιο πάνω αρχείο παρατηρούμε ότι υπάρχουν αναφορές για αρχεία τριών φοιτητών καθώς επίσης και ο τίτλος της αναφοράς. Η αναφορές για τα αρχεία των φοιτητών ορίζονται στην ιδιότητα *filename* και η τιμή της είναι το όνομα του αρχείου. Ένας τρόπος με τον οποίο θα μπορούσαμε να προσπελάσουμε τα αρχεία μέσα στο stylesheet μας είναι ο παρακάτω:

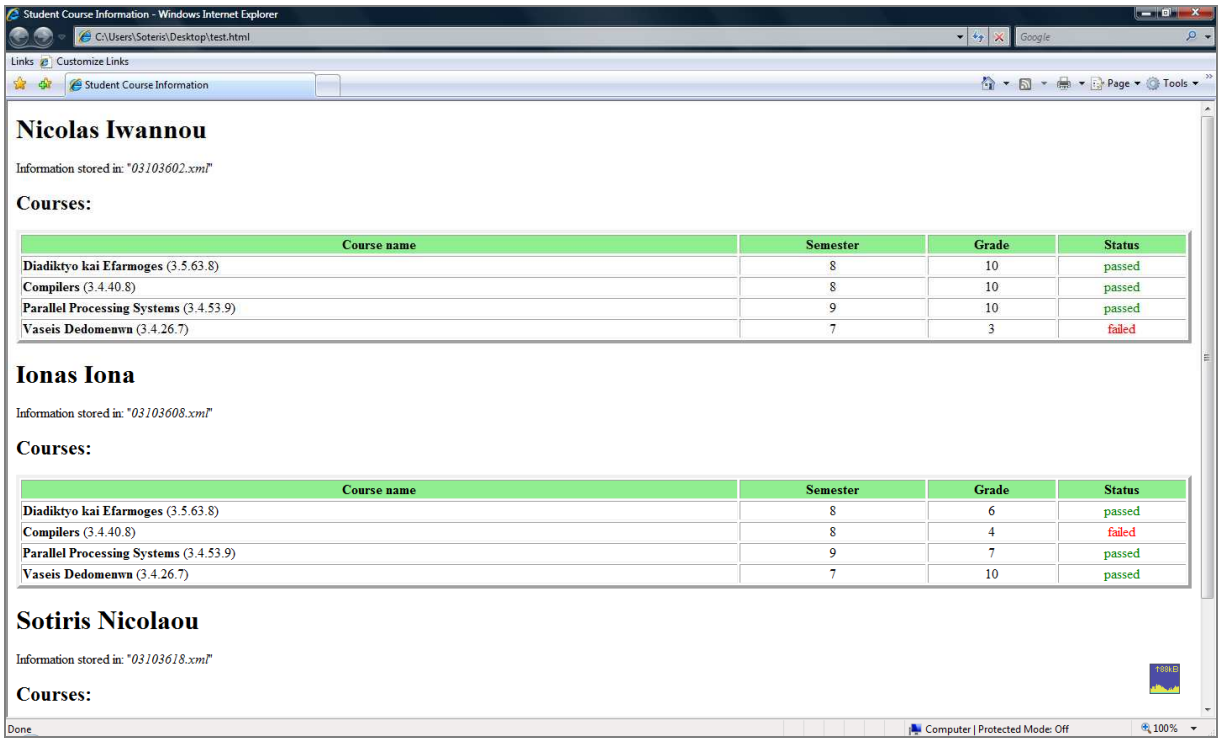
```
<xsl:template match="/">
        <xsl:for-each select="/report/stud-file">
                <xsl:apply-templates select="document(@filename)"/>
        </xsl:for-each>
</xsl:template>
```

Στο πιο πάνω template χρησιμοποιήθηκε η ιδιότητα filename σαν όρισμα στην συνάρτηση *document()*. Ο πιο εύκολος τρόπος για να φτιάξουμε μια συνοπτική αναφορά είναι να ανοίξουμε ένα-ένα τα αρχεία και να γράψουμε τα δεδομένα τους στην έξοδο. Το πιο κάτω stylesheet παρουσιάζει ένα τέτοιο τρόπο με τον οποίο επεξεργαζόμαστε τα αρχεία και δημιουργούμε μια αναφορά σε HTML για να μπορεί να προβληθεί σε ένα πρόγραμμα περιήγησης web (browser).

```
<xsl:stylesheet version="1.0"
	xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="html"/>
	<xsl:strip-space elements="*"/>

	<xsl:template match="/">
		<html>
			<head>
				<title><xsl:value-of select="/report/title"/></title>
			</head>
			<body>
			    <xsl:for-each select="/report/stud-file">
				<xsl:apply-templates  select="document(@filename)/student">
					<xsl:with-param name="fname" select="@filename"/>
				</xsl:apply-templates>
			    </xsl:for-each>
			</body>
		</html>
	</xsl:template>
	<xsl:template match="student">
		<xsl:param name="fname"/>
		<h1>
			<xsl:value-of select="student-info/name"/>
		</h1>
		<p>
			<xsl:text>Information stored in: "</xsl:text>
			<i><xsl:value-of select="$fname"/></i>
			<xsl:text>"</xsl:text>
		</p>
		<h2>Courses:</h2>
		<xsl:apply-templates select="courses"/>
	</xsl:template>
	<xsl:template match="courses">
		<table width="100%" border="4" cols="55% 15% 15% 15%">
			<tr bgcolor="lightgreen">
				<th>Course name</th>
				<th>Semester</th>
				<th>Grade</th>
				<th>Status</th>
			</tr>
			<xsl:for-each select="course">
				<tr>
					<td>
						<b><xsl:value-of select="name"/></b>
						<xsl:text> (</xsl:text>
							<xsl:value-of select="@id"/>
						<xsl:text>)</xsl:text>
					</td>
					<td align="center">
						<xsl:value-of select="semester"/>
					</td>
					<td align="center">
						<xsl:value-of select="grade"/>
					</td>
					<td align="center">
						<xsl:if test="grade &gt;4">
							<font color="green">passed</font>
						</xsl:if>
						<xsl:if test="grade &lt; 5">
							<font color="red">failed</font>
						</xsl:if>
					</td>
				</tr>
			</xsl:for-each>
		</table>
	</xsl:template>
</xsl:stylesheet>
```

Όταν εφαρμόσουμε το πιο πάνω template στο πρωτεύον αρχείο XML το αποτέλεσμα θα είναι αυτό που παρουσιάζεται στην πιο κάτω εικόνα. Το αξιοσημείωτο σε αυτό το παράδειγμα είναι το γεγονός ότι μπορέσαμε να φτιάξουμε μια αναφορά η οποία συνδυάζει δεδομένα από διάφορα αρχεία. Στην

περίπτωση μας τα αρχεία βρίσκονταν στον ίδιο υπολογιστή με το stylesheet. Θα μπορούσαν όμως τα αρχεία να βρίσκονταν σε απομακρυσμένους υπολογιστές και να προσπελάζονταν μέσω του URI τους. Δεν υπάρχει περιορισμός στον αριθμό των αρχείων που μπορούμε να συνδυάσουμε (πέρα των δυνατοτήτων του υπολογιστή μας). Επίσης σημαντικό είναι το γεγονός πως δεν χρειάστηκε να κάνουμε κάποιες αλλαγές στα αρχεία XML των φοιτητών για να φτιάξουμε την αναφορά.



γ. Μαθηματικές λειτουργίες με αναδρομή

Υπάρχουν περιπτώσεις που οι ενσωματωμένες μαθηματικές συναρτήσεις της XSLT δεν είναι ικανές να παράγουν το αποτέλεσμα που επιθυμεί ο χρήστης. Στο παρακάτω παράδειγμα χρησιμοποιείτε η τεχνική της αναδρομής για τον υπολογισμό ενός αθροίσματος και δεν γίνεται χρήση της συνάρτησης *sum()* της XSLT. Έστω το παρακάτω αρχείο XML στο οποίο θέλουμε να υπολογίσουμε την συνολική βαθμολογία ενός μαθήματος το οποίο αποτελείται από μια εξέταση και μια σειρά από ασκήσεις. Η κάθε άσκηση και η εξέταση έχει ένα συγκεκριμένο συντελεστή βαρύτητας ο οποίος πολλαπλασιάζεται με τον αντίστοιχο βαθμό και κατόπιν προστίθενται.

```
<!-- course-sum-rec.xml -->
<?xml version="1.0" encoding="UTF-8"?>

<courses>
	<course name="Diadiktyo kai Efarmoges">
		<semester>8</semester>
		<assignments>
			<assignment>
				<grade>10</grade>
				<weight>0.05</weight>
			</assignment >
			<assignment >
				<grade>8</grade>
				<weight>0.10</weight>
			</assignment>
			<assignment>
				<grade>7</grade>
				<weight>0.05</weight>
			</assignment>
		</assignments>
		<exam>
			<grade>7</grade>
			<weight>0.8</weight>
		</exam>
	</course>
</courses>
```

Εκ πρώτης όψεως, για τον υπολογισμό του αθροίσματος των βαθμών των εργαστηρίων φαίνεται πως η χρησιμοποίηση της *sum()* είναι μια καλή λύση. Θέλουμε να υπολογίσουμε το άθροισμα των βαθμών των ασκήσεων αφού πρώτα πολλαπλασιαστούν με τον κατάλληλο συντελεστή. Αν δοκιμάζαμε την παρακάτω εντολή χρησιμοποιώντας την συνάρτηση *sum()*

<xsl:value-of select=”sum(assignment/grade * assignment/lab)”/>

θα παίρναμε μήνυμα λάθους κατά την εκτέλεση του μετασχηματισμού. Η συνάρτηση *sum()* δέχεται σαν όρισμα ένα σύνολο κόμβων (*node-set*) και μετατρέπει ένα-ένα τους κόμβους σε αριθμούς και αφού τους προσθέσει επιστρέφει το άθροισμα. Η έκφραση assignment/grade * assignment/lab είναι μια απόλυτα ορθή *XPath expression* παρόλα αυτά δεν είναι ένα έγκυρο σύνολο από κόμβους. Έχοντας υπόψη αυτό τον περιορισμό, πρέπει να φτιάξουμε ένα αναδρομικό (recursive) template το οποίο θα αναλαμβάνει να εκτελέσει το άθροισμα αυτό.

```
<!-- sum-recursion.xsl -->
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:output method="text"/>

        <xsl:template match="/courses/course">
                <xsl:text>The total grade(exam and assignment) for the course </xsl:text>
                <xsl:value-of select="@name"/>
                <xsl:text> is: </xsl:text>
                <xsl:variable name="assignmentTotal">
                        <xsl:call-template name="sumGrades">
                                <xsl:with-param name="index" select="'1'"/>
                                <xsl:with-param name="assignments" select="assignments"/>
                                <xsl:with-param name="currentTotal" select="'0'"/>
                        </xsl:call-template>
                </xsl:variable>
                <xsl:value-of select="format-number($assignmentTotal + (exam/grade *
                                                        exam/weight), '.00' )"/>
        </xsl:template>

        <xsl:template name="sumGrades">
                <xsl:param name="index" select="1"/>
                <xsl:param name="assignments"/>
                <xsl:param name="currentTotal" select="0"/>
                <xsl:variable name="currentAssignment">
                        <xsl:value-of select="$assignments/assignment[$index]/grade *
                                        $assignments/assignment[$index]/weight"/>
                </xsl:variable>
                <xsl:variable name="remainingAssignments">
                        <xsl:choose>
                            <xsl:when test="$index=count($assignments/assignment)">
                                <xsl:text>0</xsl:text>
                            </xsl:when>
                            <xsl:otherwise>
                              <xsl:call-template name="sumGrades">
                                <xsl:with-param name="index" select="$index+1"/>
                                <xsl:with-param name="assignments" select="$assignments"/>
                                <xsl:with-param name="currentTotal" select="$currentTotal +
                                                $currentAssignment"/>
                              </xsl:call-template>
                            </xsl:otherwise>
                        </xsl:choose>
                </xsl:variable>
                <xsl:value-of select="$currentAssignment + $remainingAssignments"/>
        </xsl:template>

</xsl:stylesheet>
```

Στο πιο πάνω stylesheet έχουμε φτιάξει ένα template το οποίο αναδρομικά υπολογίζει το ζητούμενο άθροισμα. Ο τρόπος με τον οποίο σχεδιάστηκε το template με το όνομα "sumGrades" είναι ο εξής:

- Αρχικά περνούμε τρεις παραμέτρους στο template.
  i. *assignments*: Το σύνολο κόμβων (node-set) όλων των στοιχείων "assignment" που βρίσκονται μέσα στο στοιχείο "assignments".
  ii. *index*: Η θέση του στοιχείου "assignment" μέσα στο τρέχον σύνολο κόμβων.
  iii. *currentTotal*: Το άθροισμα των στοιχείων "assignment" τα οποία υπολογίστηκαν μέχρι στιγμής.
- Υπολογίζουμε τον συνολικό βαθμό για την συγκεκριμένη άσκηση (assignment) πολλαπλασιάζοντας τον βαθμό της άσκησης με τον αντίστοιχο συντελεστή και το αποθηκεύουμε στην μεταβλητή *currentAssignment.*

```
<xsl:variable name="currentAssignment">
        <xsl:value-of select="$assignments/assignment[$index]/grade *
                                        $assignments/assignment[$index]/weight"/>
</xsl:variable>
```

Χρησιμοποιώντας τις παραμέτρους *$index* και *$assignments* μέσα στην *XPath expression* της ιδιότητας *select* μπορούμε να προσπελάσουμε τα επιθυμητά στοιχεία (assignment) τα οποία ψάχνουμε.

- Υπολογίζουμε το άθροισμα των υπολειπόμενων κόμβων (assignment). Αν ο κόμβος που υπολογίστηκε είναι ο τελευταίος (όταν η παράμετρος *$index* είναι ίση με το πλήθος των στοιχείων "assignment") τότε το άθροισμα των υπολειπόμενων κόμβων είναι ίσο με μηδέν. Σε αντίθετη περίπτωση το άθροισμα των υπολειπόμενων κόμβων επιστρέφεται από το καλούμενο template (αναδρομή). Όταν καλούμε το template ξανά, αυξάνουμε την θέση του τρέχοντος κόμβου: `<xsl:with-param name="index" select="$index+1"/>`. Επίσης ενημερώνουμε την παράμετρο *$currentTotal* η οποία είναι ίση με την τιμή του τρέχοντος κόμβου συν την παλιά τιμή της: `<xsl:with-param name="currentTotal" select="$currentTotal + $currentAssignment"/>`.

Η προσέγγιση αυτή με την αναδρομική σχεδίαση μας επιτρέπει τον υπολογισμό του συνολικού αθροίσματος για τον σκοπό του παραδείγματος και είναι ισοδύναμη με την κλήση συνάρτησης για κάθε κόμβο που βρίσκεται στο σύνολο κόμβων. Η προσέγγιση αυτή δεν χρειάζεται να κάνει χρήση extensions με αποτέλεσμα το stylesheet του παραδείγματος να είναι συμβατό με τους standard compliant XSLT processors έτσι δεν χρειάζεται να εισάγουμε (import) extension functions και extension elements.

3. Γενικός Προγραμματισμός(Generic Programming)

Μια από τις καλύτερες χρήσεις της XSLT και της XML είναι η δυνατότητα επίλυσης μιας σειράς διαφορετικών προβλημάτων με την χρησιμοποίηση έτοιμου κώδικα. Η υλοποίηση μιας τέτοιας λειτουργίας είναι συνήθως μια επίπονη διαδικασία καθώς η XSLT δεν σχεδιάστηκε ειδικά σαν μια generic γλώσσα ή συναρτησιακή γλώσσα, παρόλα αυτά έχει ενσωματωμένες τέτοιες δυνατότητες οι οποίες επιτρέπουν μια τέτοια χρήση. Μια τέτοια υλοποίηση μοιάζει με γλώσσες σαν την Lisp, ML ή Haskell οι οποίες επιτρέπουν την δημιουργία συναρτήσεων γενικού σκοπού οι οποίες δέχονται σαν όρισμα συναρτήσεις ειδικού σκοπού

Στην συνέχεια θα αναπτυχθούν διάφορα παραδείγματα τα οποία κάνουν χρήση αυτής της ικανότητας της XSLT

α. Γενικά

α.1 Επέκταση περιεχομένων καθολικών μεταβλητών

Η XSLT δίνει την δυνατότητα να συνδυαστούν τα περιεχόμενα μιας καθολικής μεταβλητής που ορίζονται σε ένα imported stylesheet με τα περιεχόμενα μιας καθολικής μεταβλητής του importing stylesheet. Το παρακάτω παράδειγμα ορίζει δυο μεταβλητές, η πρώτη *$variable1* είναι μοναδική στο πιο κάτω stylesheet. Η δεύτερη μεταβλητή *$variable* ορίζεται πάλι στο πρώτο stylesheet αλλά όπως θα φανεί και στην συνέχεια μπορεί να γίνει override.

```
<!—ext-var-courses.xsl -->
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:sot="http://www.s-nicolaou.com">
        <xsl:output method="xml" indent="yes" />

        <sot:course name="Mathimatiki Analisi 1">
                <semester>1</semester>
                <grade>6</grade>
        </sot:course>
        <sot:course name="Fysiki 1">
                <semester>1</semester>
                <grade>7</grade>
        </sot:course>
        <xsl:variable name="variable1" select="document('')/*/sot:*"/>
        <xsl:variable name="variable" select="$variable1"/>
                <xsl:template match="/">
                <courses>
                        <xsl:copy-of select="$variable"/>
                </courses>
        </xsl:template>
</xsl:stylesheet>
```

Στην συνέχεια θα ορίσουμε ένα νέο stylesheet το οποίο θα επεκτείνει τα περιεχόμενα της μεταβλητής *$variable1*. Το δεύτερο stylesheet ορίζει επίσης μια μοναδική μεταβλητή *$variable2* η οποία είναι η ένωση των περιεχομένων της μεταβλητής του πρώτου stylesheet και των τοπικών δεδομένων του δευτέρου stylesheet. Ο τρόπος να επιτευχθεί είναι με τον παρακάτω ορισμό: <xsl:variable name="variable2" select="document('')/*/sot:* | $variable1"/>. Με τον τρόπο αυτό όπως θα δούμε και στην έξοδο του stylesheet μπορούμε να έχουμε πρόσβαση στα δεδομένα του stylesheet με μια κοινή μεταβλητή, εδώ την *$variable*.

```
<!—- ext-var-courses2.xsl -->
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:sot="http://www.s-nicolaou.com"
        exclude-result-prefixes="xsl">

        <xsl:import href="ext-var-courses1.xsl"/>

        <xsl:output method="xml" indent="yes"/>

        <sot:course name="Compilers">
                <semester>8</semester>
                <grade>9</grade>
        </sot:course>
        <sot:course name="Parallel Processing Systems">
                <semester>9</semester>
                <grade>9</grade>
        </sot:course>

        <xsl:variable name="variable2" select="document('')/*/sot:* | $variable1"/>
        <xsl:variable name="variable" select="$variable2"/>

        <xsl:template match="/">
                <grades>
                        <xsl:copy-of select="$variable"/>
                </grades>
        </xsl:template>

</xsl:stylesheet>
```

Το παράδειγμα αυτό μας δίνει στην έξοδο ένα αρχείο XML(βλέπε παρακάτω) το οποίο περιέχει τέσσερα στοιχεία “course” εκ των οποίων τα δυο προέρχονται από το 1ο stylesheet και τα άλλα δυο προέρχονται από το 2ο stylesheet χρησιμοποιώντας μόνο την μεταβλητή *$variable* η οποία ορίζεται στο 2ο stylesheet.

```
<?xml version="1.0"?>
<grades>
	<sot:course name="Compilers">
		<semester>8</semester>
		<grade>9</grade>
	</sot:course>
	<sot:course name="Parallel Processing Systems">
		<semester>9</semester>
		<grade>9</grade>
	</sot:course>
	<sot:course name="Mathimatiki Analisi 1">
		<semester>1</semester>
		<grade>6</grade>
	</sot:course>
	<sot:course name="Fysiki 1">
		<semester>1</semester>
		<grade>7</grade>
	</sot:course>
</grades>
```

Το πιο πάνω παράδειγμα έχει ως σκοπό να δείξει τον τρόπο με τον οποίο μπορεί να επεκτείνει την λειτουργικότητα των stylesheet του. Με πιο τρόπο? Αν στην θέση των στοιχείων "course" υπήρχαν συναρτήσεις οι οποίες είχαν κάποια λειτουργικότητα π.χ. sum, avg κτλ θα μπορούσαν αυτές να προσπελαστούν με μια κοινή μεταβλητή που θα οριζόταν στο importing stylesheet χωρίς να παίζει ρόλο σε ποιό stylesheet ορίστηκαν όπως θα δούμε και στην συνέχεια.

β. Συνάθροιση (Aggregation)

Στην παράγραφο 3.γ παρουσιάστηκε ένας τρόπος για την υλοποίηση μαθηματικών λειτουργιών και πιο συγκεκριμένα ο υπολογισμός ενός αθροίσματος με την βοήθεια ενός αναδρομικού template. Συχνά στις εφαρμογές χρειάζεται να εκτελέσουμε κάποιες μαθηματικές λειτουργίες σε μια συλλογή από δεδομένα όπως για παράδειγμα ο υπολογισμός του μέσου όρου για τα μαθήματα ενός φοιτητή ή ο υπολογισμός των συνολικών εσόδων μιας εταιρίας από τις πωλήσεις. Οι περιπτώσεις είναι πάρα πολλές αλλά η υλοποίηση τέτοιων λειτουργιών δεν παρέχεται απευθείας από την XSLT. Θα μπορούσαμε να φτιάξουμε μια συλλογή από αναδρομικά templates κάθε ένα από τα οποία θα εκτελεί μια συγκεκριμένη λειτουργία, π.χ. square, sum, avg, min, max, variance κ.ά. Οι λειτουργίες αυτές δεν χρειάζεται να είναι μαθηματικές. Θα μπορούσαν για παράδειγμα να είναι λειτουργίες οι οποίες εφαρμόζονται επί συμβολοσειρών.

Ο τρόπος με τον οποίο μπορούν να χρησιμοποιηθούν αυτά τα templates δεν διαφέρει ουσιαστικά από τον τρόπο που χρησιμοποιούμε διάφορες βιβλιοθήκες συναρτήσεων στις γλώσσες προγραμματισμού(εισάγονται στην αρχή του

προγράμματος και καλούνται όταν χρειαστεί). Η συλλογή των templates είναι αποθηκευμένη σε ένα stylesheet το οποίο εισάγουμε στο stylesheet της εφαρμογής μας και μπορούμε να χρησιμοποιήσουμε οποιοδήποτε από τα templates του, καλώντας το με το όνομα του και περνώντας τις αντίστοιχες παραμέτρους.

Στο παρακάτω παράδειγμα παρουσιάζεται ένα τμήμα από τον κώδικα ενός stylesheet το οποίο εκτελεί τις προαναφερθείσες μαθηματικές λειτουργίες:

```
<!—- my-aggr-1.xsl -->
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:generic="http://www.ora.com/XSLTCookbook/namespaces/generic"
        xmlns:aggr="http://www.ora.com/XSLTCookbook/namespaces/aggregate"
        extension-element-prefixes="generic aggr ">

        <xsl:key name="generic:aggr-funcs" match="generic:aggr-func" use="@name"/>
        <xsl:key name="generic:funcs" match="generic:func" use="@name"/>

         <xsl:variable name="aggr:public-generics" select="document('')/*/generic:*"/>
         <xsl:variable name="aggr:generics" select="$aggr:public-generics"/>

        <!-- Primitive generic functions on x -->

        <generic:func name="identity"/>
        <xsl:template match="generic:func[@name='identity']">
                <xsl:param name="x"/>
                <xsl:value-of select="$x"/>
        </xsl:template>

        …

        <generic:aggr-func name="sum" identity="0"/>
        <xsl:template match="generic:aggr-func[@name='sum']">
                <xsl:param name="x"/>
                <xsl:param name="accum"/>
                <xsl:value-of select="$x + $accum"/>
        </xsl:template>

        …

        <!-- Generic aggregation template -->
        <xsl:template name="aggr:aggregation">
                <xsl:param name="nodes"/>
                <xsl:param name="aggr-func" select=" 'sum' "/>
                <xsl:param name="func" select=" 'identity' "/>
                <xsl:param name="func-param1" select="$aggr:generics[self::generic:func and
                                                    @name = $func]/@param1"/>
                <xsl:param name="i" select="1"/>
                <xsl:param name="accum" select="$aggr:generics[self::generic:aggr-func and
                                                    @name = $aggr-func]/@identity"/>
                <xsl:choose>
                        <xsl:when test="$nodes">
                            <!--Compute func($x) -->
                            <xsl:variable name="f-of-x">
                                <xsl:for-each select="$aggr:generics[1]">
                                    <xsl:apply-templates select="key('generic:funcs',$func)">
                                        <xsl:with-param name="x" select="$nodes[1]"/>
                                        <xsl:with-param name="i" select="$i"/>
                                    </xsl:apply-templates>
                                </xsl:for-each>
                            </xsl:variable>
```

```
<!—- my-aggr-1.xsl (συνέχεια) -->
                        <!-- Aggregate current $f-of-x with $accum -->
                        <xsl:variable name="temp">
                            <xsl:for-each select="$aggr:generics[1]">
                                <xsl:apply-templates select="key('generic:aggr-funcs',
                                                                 $aggr-func)">
                                    <xsl:with-param name="x" select="$f-of-x"/>
                                    <xsl:with-param name="accum" select="$accum"/>
                                    <xsl:with-param name="i" select="$i"/>
                                </xsl:apply-templates>
                            </xsl:for-each>
                        </xsl:variable>

                    <!—- Υπολογίζουμε αναδρομικά τους υπολειπόμενους κόμβους
                                χρησιμοποιώντας την συνάρτηση position() -->
                        <xsl:call-template name="aggr:aggregation">
                            <xsl:with-param name="nodes" select="$nodes[position()!=1]"/>
                            <xsl:with-param name="aggr-func" select="$aggr-func"/>
                            <xsl:with-param name="func" select="$func"/>
                            <xsl:with-param name="func-param1" select="$func-param1"/>
                            <xsl:with-param name="i" select="$i + 1"/>
                            <xsl:with-param name="accum" select="$temp"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="$accum"/>
                    </xsl:otherwise>
                </xsl:choose>
        </xsl:template>

</xsl:stylesheet>
```

Το πιο πάνω stylesheet στο οποίο έχουμε συμπεριλάβει μόνο την μαθηματική λειτουργία sum (για λόγους οικονομίας χώρου και απλότητας) αποτελείται από τρία βασικά τμήματα:

- Το πρώτο μέρος αποτελείται από generic functions οι οποίες έχουν ετικέτα (tagged) και οι οποίες δρουν επί της απλής μεταβλητής x. Παράδειγμα τέτοιων συναρτήσεων είναι οι: $f(x)=x$, $f(x)=x^2$, $f(x)=x^3$, $f(x)=x+1$, $f(x)=x-1$, $f(x)=1/x$. Ο λόγος για τον οποίο υπάρχουν αυτές οι συναρτήσεις είναι για να αποκτήσει ο κώδικας μεγαλύτερη χρησιμότητα. Και αυτό γιατί με το ίδιο stylesheet θα μπορούσαμε να υπολογίζαμε ένα απλό άθροισμα ή το άθροισμα τετραγώνων ή το άθροισμα των αντίστροφων κ.ο.κ. Το stylesheet αυτό είναι επεκτάσιμο. Ο χρήστης μπορεί να προσθέσει και μόνος του άλλες παρόμοιες συναρτήσεις.
- Το δεύτερο μέρος αποτελείται από συναθροιστικές (aggregation) generic functions οι οποίες και αυτές έχουν ετικέτα. Στο stylesheet έχει συμπεριληφθεί μόνο η συνάρτηση sum. Άλλες παρόμοιες συναρτήσεις είναι το γινόμενο (product), ο μέσος όρος (average) το ελάχιστο και μέγιστο ή ακόμη και η διασπορά. Και εδώ ο χρήστης μπορεί να προσθέσει και άλλους τρόπους συνάθροισης δεδομένων.
- Το τρίτο μέρος είναι και το πιο σημαντικό μέρος για την λειτουργία του stylesheet. Αποτελείται από τον generic aggregation αλγόριθμο. Ο

αλγόριθμος αυτός δεν είναι τίποτα άλλο από ένα named template, το οποίο δέχεται ένα αριθμό παραμέτρων οι οποίες χρησιμεύουν για τον υπολογισμό του επιθυμητού αποτελέσματος του χρήστη. Οι παράμετροι αυτοί είναι το σύνολο κόμβων (node-set) που θέλουμε να συναθροίσουμε, το όνομα της συναθροιστικής συνάρτησης που θέλουμε να χρησιμοποιήσουμε και το όνομα της απλής συνάρτησης που επιδρά στην μεταβλητή x. Πέρα από αυτές τις τρείς, υπάρχουν και άλλες παράμετροι που χρησιμεύουν στην λειτουργία του stylesheet. Η παράμετρος *$i* κρατάει την θέση του κόμβου που επεξεργάστηκε τελευταίος. Η παράμετρος *$accum* αποθηκεύει το τρέχον άθροισμα της συνάθροισης. Σημαντικό στοιχείο είναι ο τρόπος με τον οποίο αρχικοποιείται η τιμή της παραμέτρου *$accum*. Η τιμή της *$accum* αρχικοποιείται από την τιμή της ιδιότητας *identity* η οποία είναι φυλαγμένη στην ετικέτα της συναθροιστικής συνάρτησης, π.χ. για το άθροισμα η τιμή της *identity* είναι ίση με μηδέν, για το γινόμενο είναι ίση με ένα κ.ο.κ. Αυτός ο τρόπος αρχικοποίησης είναι ένα μια από τις σημαντικότερες δυνατότητες της generic προσέγγισης με την οποία μετα-δεδομένα συσχετίζονται με ετικέτες συναρτήσεων.

Για να γίνει πιο κατανοητός ο πιο πάνω κώδικας θα χρησιμοποιήσουμε ένα παράδειγμα το οποίο κάνει χρήση των πιο συναθροιστικών λειτουργιών. Έστω, η βιβλιοθήκη ενός πανεπιστημιακού ιδρύματος κρατάει στοιχεία για τους δανεισμούς και τις επιστροφές βιβλίων σε καθημερινή βάση και επιθυμεί να φτιάξει μια αναφορά με τα στοιχεία να παρουσιάζονται σε μια ιστοσελίδα συνοπτικά ανά εβδομάδα. Τα δεδομένα είναι αποθηκευμένα σε ένα αρχείο XML το οποίο έχει την παρακάτω μορφή:

```
<?xml version="1.0" encoding="UTF-8" ?>
<library>
        <name>National Technical University of Athens Library</name>
        <weeks>
                <week no="1">
                        <start-date>7-1-2008</start-date>
                        <end-date>13-1-2008</end-date>
                        <books>
                                <monday><in>183</in><out>249</out></monday>
                                <tuesday><in>173</in><out>154</out></tuesday>
                                <wednesday><in>134</in><out>56</out></wednesday>
                                <thursday><in>135</in><out>112</out></thursday>
                                <friday><in>113</in><out>239</out></friday>
                        </books>
                </week>
                <week no=”…”>…</week>
        </weeks>
</library>
```

Ο σκοπός του παραδείγματος είναι να υπολογιστούν για κάθε εβδομάδα το σύνολο των βιβλίων που δανείζονται και επιστρέφονται καθώς επίσης και τον ελάχιστο και μέγιστο αριθμό βιβλίων που δανείστηκαν ή επιστράφηκαν σε μια ημέρα ανά εβδομάδα. Για τον υπολογισμό των πιο πάνω στοιχείων θα χρησιμοποιήσουμε στο stylesheet του παραδείγματος τα templates που εκτελούν τις απαιτούμενες συναθροιστικές λειτουργίες. Πριν χρησιμοποιηθούν τα templates πρέπει να εισάγουμε (import) στο stylesheet το stylesheet στο οποίο είναι αποθηκευμένα τα templates. Ο τρόπος για να γίνει αυτό είναι ο παρακάτω:

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:generic="http://www.ora.com/XSLTCookbook/namespaces/generic"
        xmlns:aggr="http://www.ora.com/XSLTCookbook/namespaces/aggregate"
        extension-element-prefixes="generic aggr">

        <xsl:import href="my-aggr-1.xsl"/>
```

Με έντονη γραμματοσειρά είναι οι ιδιότητες και τα στοιχεία τα οποία είναι απαραίτητα ώστε να μπορούν να χρησιμοποιηθούν τα εισαγόμενα templates. Το στοιχείο <xsl:import> εισάγει στο stylesheet το αρχείο "*my-aggr-1.xsl"* στο οποίο είναι αποθηκευμένα τα προαναφερθείσα templates. Απαραίτητο είναι επίσης να οριστούν και οι χώροι ονομάτων (namespaces) του stylesheet του οποίου εισάγουμε. Αν δεν γινόταν αυτό, ο XSLT processor δεν θα μπορούσε να αναγνωρίσει τα templates του εισαγόμενου stylesheet.

Αφού εισαχθεί το stylesheet μπορούμε πλέον να το χρησιμοποιήσουμε στην εφαρμογή μας. Για να το χρησιμοποιήσουμε καλούμε το template το οποίο αντιστοιχεί στην συναθροιστική συνάρτηση που επιθυμούμε να χρησιμοποιήσουμε και ορίζουμε τις αντίστοιχες παραμέτρους όπως πιο κάτω:

```
<xsl:call-template name="aggr:aggregation">
        <xsl:with-param name="nodes" select="books/*/in"/>
        <xsl:with-param name="aggr-func" select=" 'sum' "/>
</xsl:call-template>
```

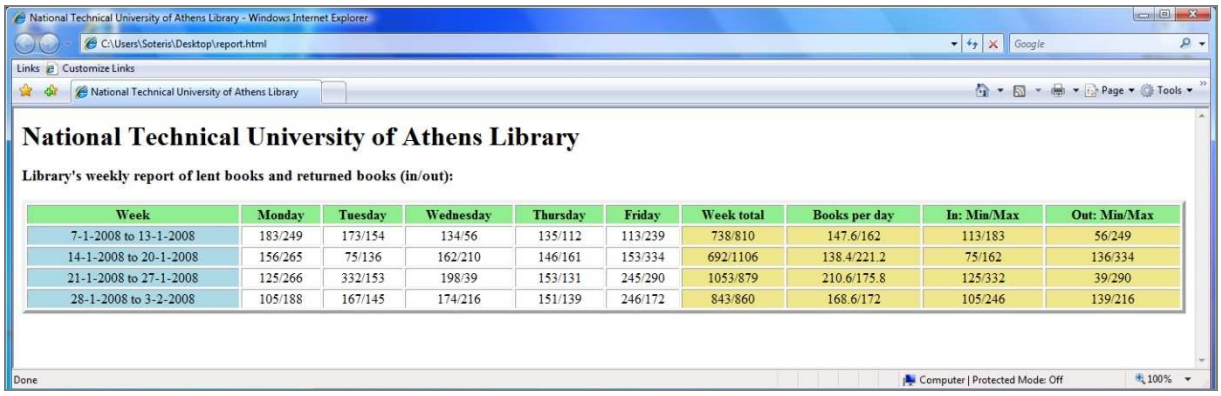
Στην κλήση αυτού του template περνάμε δύο παραμέτρους. Η πρώτη παράμετρος *$nodes* είναι το σύνολο κόμβων που σε αυτή την περίπτωση είναι τα υποστοιχεία "in" των στοιχείων "monday", … , "friday". Η δεύτερη παράμετρος *$aggr-func* είναι το είδος της συναθροιστικής συνάρτησης που θέλουμε να χρησιμοποιήσουμε και στην προκειμένη είναι η συνάρτηση "sum". Με τον τρόπο αυτό παίρνουμε το σύνολο (άθροισμα) των βιβλίων που επιστράφηκαν σε μια εβδομάδα.

Παρακάτω, παρατίθεται ο κώδικας ενός πιο ολοκληρωμένου παραδείγματος που κάνει χρήση των sum, avg, min και max:

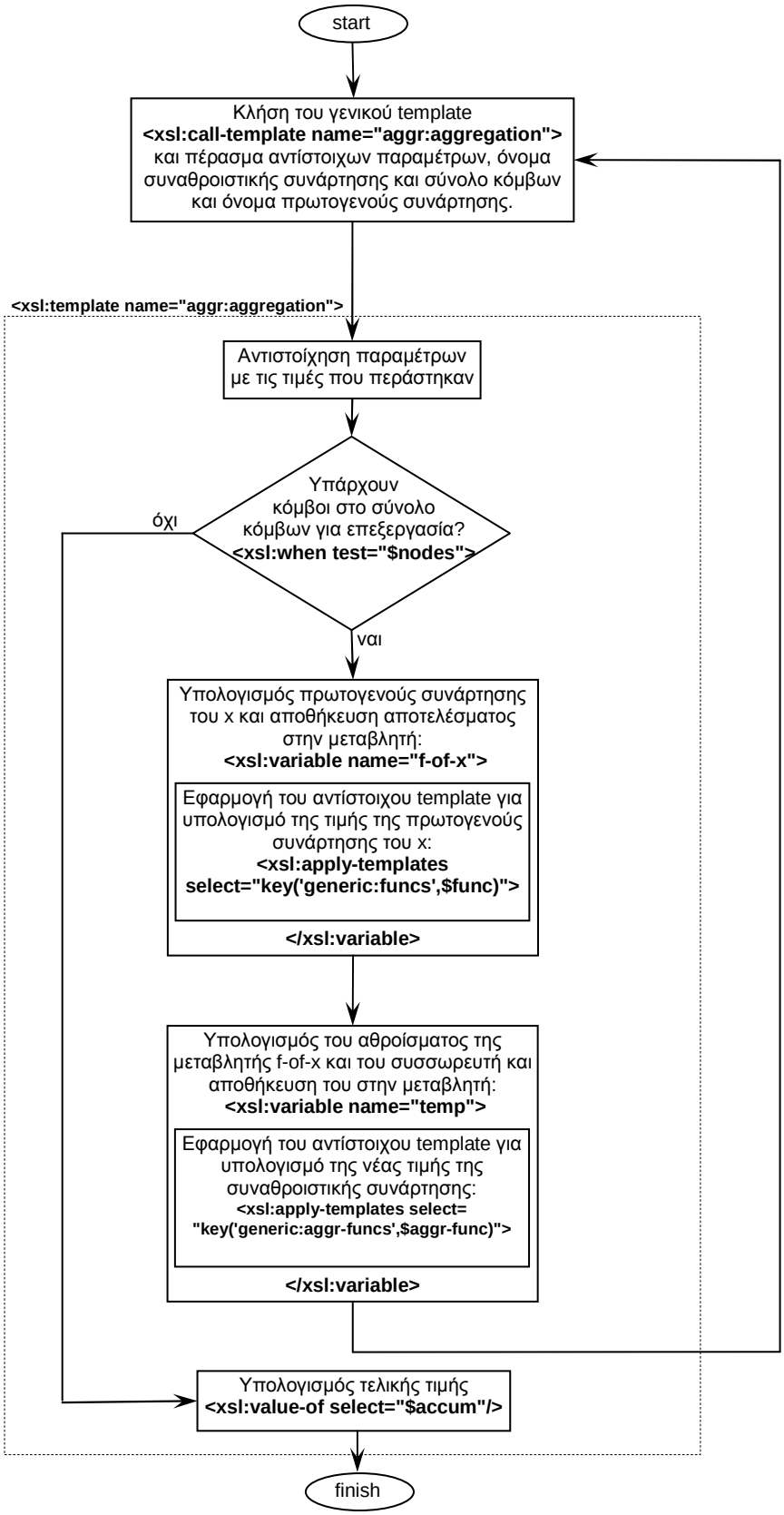
```
<!—- book-library-xsl.xsl -->
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:generic="http://www.ora.com/XSLTCookbook/namespaces/generic"
    xmlns:aggr="http://www.ora.com/XSLTCookbook/namespaces/aggregate"
    extension-element-prefixes="generic aggr">

    <xsl:import href="my-aggr-1.xsl"/>
    <xsl:output method="html" indent="yes"/>

    <xsl:template match="/library">
        <html>
            <head><title><xsl:value-of select="name"/></title></head>
            <body>
                <h1><xsl:value-of select="name"/></h1>
                <h3>Library's weekly report of lent books and returned books (in/out):</h3>
                <xsl:apply-templates select="weeks"/>
            </body>
        </html>
    </xsl:template>

    <xsl:template match="weeks">
        <table width="100%" cols="15% 9% 9% 9% 9% 9% 10% 10% 10% 10%" align="center">
            <tr bgcolor="lightgreen">
                <th>Week</th><th>Monday</th><th>Tuesday</th>
                <th>Wednesday</th><th>Thursday</th><th>Friday</th>
                <th>Week total</th><th>Books per day</th>
                <th>In: Min/Max</th><th>Out: Min/Max</th>
            </tr>
            <xsl:for-each select="week">
                <tr align="center">
                    <td bgcolor="lightblue">
                        <xsl:value-of select="start-date"/>
                        <xsl:text> to </xsl:text>
                        <xsl:value-of select="end-date"/>
                    </td>
                    <xsl:for-each select="books/*">
                        <td>
                            <xsl:value-of select="in"/><xsl:text>/</xsl:text>
                            <xsl:value-of select="out"/>
                        </td>
                    </xsl:for-each>
                    <td bgcolor="khaki">
                        <xsl:call-template name="aggr:aggregation">
                            <xsl:with-param name="nodes" select="books/*/in"/>
                            <xsl:with-param name="aggr-func" select=" 'sum' "/>
                        </xsl:call-template>
                        <xsl:text>/</xsl:text>
                        <xsl:call-template name="aggr:aggregation">
                            <xsl:with-param name="nodes" select="books/*/out"/>
                            <xsl:with-param name="aggr-func" select=" 'sum' "/>
                        </xsl:call-template>
                    </td>
                    <td bgcolor="khaki">
                        <xsl:call-template name="aggr:aggregation">
                            <xsl:with-param name="nodes" select="books/*/in"/>
                            <xsl:with-param name="aggr-func" select=" 'avg' "/>
                        </xsl:call-template>
                        <xsl:text>/</xsl:text>
                        <xsl:call-template name="aggr:aggregation">
                            <xsl:with-param name="nodes" select="books/*/out"/>
                            <xsl:with-param name="aggr-func" select=" 'avg' "/>
                        </xsl:call-template>
                    </td>
```

```
<!—- book-library-xsl.xsl(συνέχεια) -->
                    <td bgcolor="khaki">
                        <xsl:call-template name="aggr:aggregation">
                            <xsl:with-param name="nodes" select="books/*/in"/>
                            <xsl:with-param name="aggr-func" select=" 'min' "/>
                        </xsl:call-template>
                        <xsl:text>/</xsl:text>
                        <xsl:call-template name="aggr:aggregation">
                            <xsl:with-param name="nodes" select="books/*/in"/>
                            <xsl:with-param name="aggr-func" select=" 'max' "/>
                        </xsl:call-template>
                    </td>
                    <td bgcolor="khaki">
                        <xsl:call-template name="aggr:aggregation">
                            <xsl:with-param name="nodes" select="books/*/out"/>
                            <xsl:with-param name="aggr-func" select=" 'min' "/>
                        </xsl:call-template>
                        <xsl:text>/</xsl:text>
                        <xsl:call-template name="aggr:aggregation">
                            <xsl:with-param name="nodes" select="books/*/out"/>
                            <xsl:with-param name="aggr-func" select=" 'max' "/>
                        </xsl:call-template>
                    </td>
                </tr>
            </xsl:for-each>
        </table>
    </xsl:template>
</xsl:stylesheet>
```

Όταν το πιο πάνω stylesheet εφαρμοστεί στο αρχείο XML που είναι αποθηκευμένα τα δεδομένα, θα παραχθεί ένα αρχείο HTML το οποίο όταν το τρέξουμε με κάποιον browser θα μας δώσει την πιο κάτω εικόνα αποτελέσματος. Παρατηρούμε στις τέσσερις τελευταίες στήλες(με κίτρινο χρωματισμό), τα δεδομένα είναι αποτελέσματα τα οποία προέκυψαν από την εφαρμογή των συναθροιστικών συναρτήσεων στα δεδομένα των στηλών που αντιστοιχούν στις ημέρες της εβδομάδας. Η πρώτη κίτρινη στήλη δίνει το άθροισμα των βιβλίων που δανείζονται ή επιστρέφονται σε μια εβδομάδα, η δεύτερη δίνει τον μέσο όρο βιβλίων που δανείζονται ή επιστρέφονται καθημερινά σε εβδομαδιαία βάση, και οι δυο τελευταίες στήλες δίνουν τον μέγιστο αριθμό και τον ελάχιστο αριθμό βιβλίων που δανείστηκαν ή επιστράφηκαν την αντίστοιχη εβδομάδα σε καθημερινή βάση.



β1. Αλγόριθμος κλήσης αναδρομικών templates



Ο τρόπος με τον οποίο λειτουργούν τα αναδρομικά templates στο stylesheet “my-aggr-1.xsl” που παρατέθηκε πιο πάνω, θα μπορούσε να παρασταθεί με το πιο πάνω διάγραμμα αλγόριθμου.

4. Η Βιβλιοθήκη Συνάθροισης

α. Γενικά

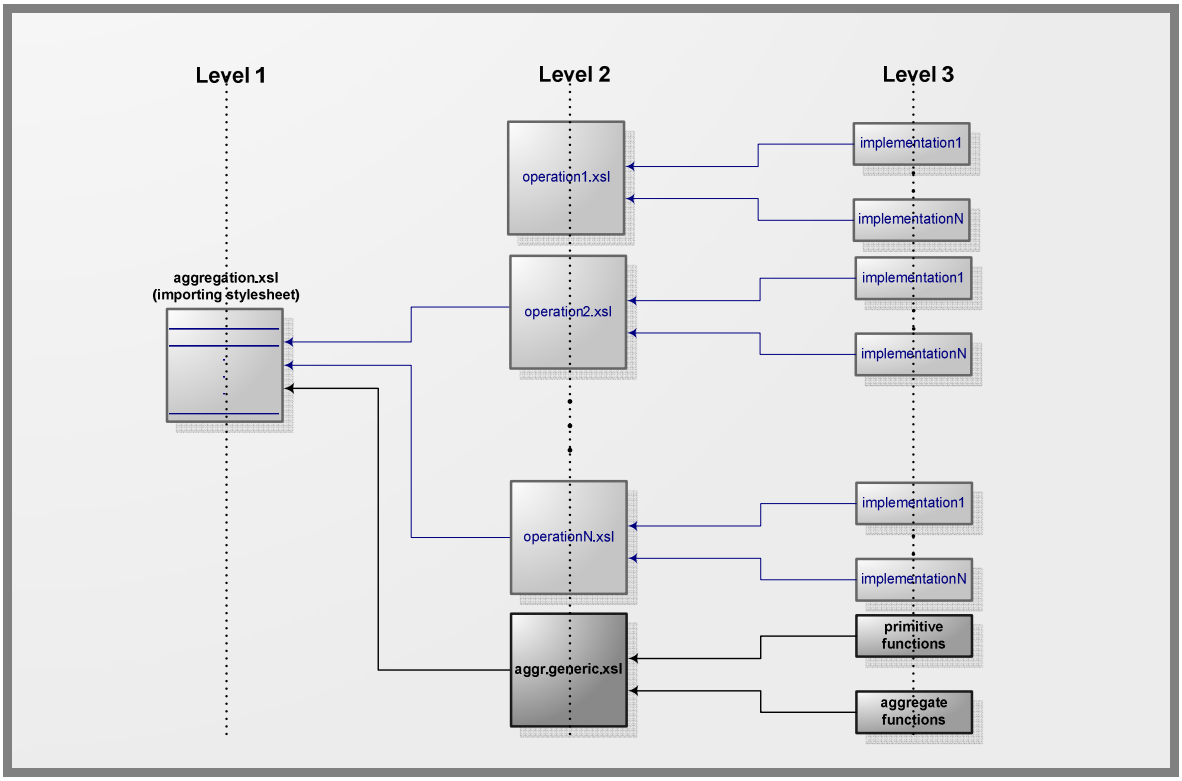
Στην παράγραφο 3.β παρουσιάστηκε ένα τμήμα της βιβλιοθήκης που δημιουργήσαμε το οποίο εκτελεί κάποιες από τις βασικές λειτουργίες συνάθροισης, π.χ. άθροισμα και μέσος όρος. Στις παραγράφους του κεφαλαίου αυτού γίνεται επεξήγηση των λειτουργιών της βιβλιοθήκης και του τρόπου με τον οποίο αυτή σχεδιάστηκε και υλοποιήθηκε. Όπως αναφέρθηκε και πιο πριν, ο σκοπός της εργασίας αυτής είναι η συγγραφή μιας βιβλιοθήκης από templates η οποία θα παρέχει λειτουργίες, μαθηματικές κυρίως, οι οποίες θα βοηθούν τον χρήστη να δημιουργήσει εφαρμογές XSLT σε μικρό χρονικό διάστημα.

Οι λειτουργίες της βιβλιοθήκης υλοποιήθηκαν με δυο τρόπους. Και οι δυο τρόποι υλοποίησης των λειτουργιών έχουν παρουσιαστεί σε προηγούμενα κεφάλαια της εργασίας αυτής. Οι δυο τρόποι υλοποίησης των λειτουργιών είναι, α) με κλήση named template και β) με generic programming. Δεν υπάρχει διάκριση στους τρόπους υλοποίησης όσο αφορά ποιος είναι καλύτερος. Η επιλογή του τρόπου που θα χρησιμοποιηθεί στην εφαρμογή έγκειται στον προγραμματιστή. Όσο αφορά την υλοποίηση, οι δυο τρόποι μοιάζουν μεταξύ τους αφού χρησιμοποιούν αναδρομική κλήση templates και δέχονται τις ίδιες παραμέτρους κατά την κλήση τους. Αυτό που στην ουσία διαφέρει είναι ο τρόπος κλήσης των templates.

β. Δομή της Βιβλιοθήκης

Η βιβλιοθήκη έχει δημιουργηθεί ακολουθώντας τα πρότυπα των extensions που ακολουθούν οι διάφοροι οργανισμοί (π.χ. www.exslt.org). Η δομή του μπορεί να χαρακτηριστεί ως "ιεραρχική" καθώς αποτελείται από ένα stylesheet ανώτερου επίπεδου και ένα αριθμό stylesheet κατώτερου επιπέδου τα οποία αντιστοιχούν στις διάφορες λειτουργίες που υλοποιούνται στην βιβλιοθήκη. Επίσης, αυτά τα stylesheet, με την σειρά τους θα μπορούσε να πει κανείς ότι έχουν ως παιδιά τους άλλα stylesheet τα οποία αφορούν τους διαφορετικούς

τρόπους υλοποίησης των λειτουργιών της βιβλιοθήκης (π.χ. functions, named templates, JavaScript κ.α.). Η δομή της βιβλιοθήκης θα μπορούσε να αναπαρασταθεί με το πιο κάτω διάγραμμα.



Στο πιο πάνω διάγραμμα αναπαρίσταται η δομή της βιβλιοθήκης. Με έντονη γραμμή φαίνεται το κομμάτι που υλοποιεί τις λειτουργίες βιβλιοθήκης με generic programming. Όπως αναφέρθηκε και πιο πριν η υλοποίηση των λειτουργιών με generic programming πραγματοποιείται με ακριβώς τρία stylesheet σε αντίθεση με τις υλοποιήσεις των λειτουργιών με named templates, όπως φαίνεται και στο διάγραμμα. Το stylesheet *aggr.generic.xsl* το οποίο βρίσκεται στο level 2 αποτελεί τον αλγόριθμο εκτέλεσης και επιλογής της κατάλληλης πρωτογενούς συνάρτησης και της κατάλληλης συναθροιστικής συνάρτησης από τα δυο stylesheet, *primitive functions και aggregate functions* αντίστοιχα, που βρίσκονται στο level 3.

Οι υπόλοιπες λειτουργίες (operation 1 – operation n), με μη έντονη γραμμή στο level 2, αφορούν συναθροιστικές λειτουργίες οι οποίες η κάθε μια υλοποιείται με ένα named template το οποίο είναι αποθηκευμένο σε ένα αρχείο xsl στο επίπεδο 3.

γ. Συναθροιστικές Λειτουργίες

Στους δυο πίνακες που ακολουθούν, συνοψίζονται οι λειτουργίες που περιλαμβάνει η βιβλιοθήκη. Ο πρώτος πίνακας περιλαμβάνει τα named templates και ο δεύτερος περιλαμβάνει το generic template.

γ.1 Named Templates

Named Templates			
Α/α	Λειτουργία	Όνομα template	Παράμετροι
1	Άθροισμα	"aggr:sum"	**(1)** **<xsl:param name="nodes" select="/.." />** Το σύνολο των κόμβων που θα αθροιστούν. Κατά την κλήση του template, μια xpath έκφραση δίνεται ως παράμετρος και αντιπροσωπεύει το σύνολο των κόμβων **(2)** **<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε σε κάθε κλήση του template να προσπελάζουμε τον επόμενο κόμβο. **(3)** **<xsl:param name="accum" select="0"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.
2	Άθροισμα τετραγώνων	"aggr:sum-square"	**(1)** **<xsl:param name="nodes" select="/.." />** Το σύνολο των κόμβων που θα αθροιστούν. Κατά την κλήση του template, μια xpath έκφραση δίνεται ως παράμετρος και αντιπροσωπεύει το σύνολο των κόμβων **(2)** **<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε σε κάθε κλήση του template να προσπελάζουμε τον επόμενο κόμβο. **(3)** **<xsl:param name="accum" select="0"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.
3	Άθροισμα κύβων	"aggr:sum-cube"	**(1)** **<xsl:param name="nodes" select="/.." />** Το σύνολο των κόμβων που θα αθροιστούν. Κατά την κλήση του template, μια xpath έκφραση δίνεται ως παράμετρος και αντιπροσωπεύει το σύνολο των κόμβων **(2)** **<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε σε κάθε κλήση του template να προσπελάζουμε τον επόμενο κόμβο. **(3)** **<xsl:param name="accum" select="0"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.
4	Άθροισμα αντίστροφου	"aggr:sum-reciprocal"	**(1)** **<xsl:param name="nodes" select="/.." />** Το σύνολο των κόμβων που θα αθροιστούν. Κατά την κλήση του template, μια xpath έκφραση δίνεται ως παράμετρος και αντιπροσωπεύει το σύνολο των κόμβων **(2)** **<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε σε κάθε κλήση του template να προσπελάζουμε τον επόμενο κόμβο. **(3)** **<xsl:param name="accum" select="0"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.
5	Γινόμενο	"aggr:prod"	**(1)** **<xsl:param name="nodes" select="/.." />** Το σύνολο των κόμβων που θα αθροιστούν. Κατά την κλήση του template, μια xpath έκφραση δίνεται ως παράμετρος και αντιπροσωπεύει το σύνολο των κόμβων **(2)** **<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε

				σε κάθε κλήση του template να προσπελάζουμε τον επόμενο κόμβο.
			(3)	**<xsl:param name="accum" select="1"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.
6	Γινόμενο αντίστροφου	"aggr:prod-reciprocal"	**(1)**	**<xsl:param name="nodes" select="/.." />** Το σύνολο των κόμβων που θα αθροιστούν. Κατά την κλήση του template, μια xpath έκφραση δίνεται ως παράμετρος και αντιπροσωπεύει το σύνολο των κόμβων
			(2)	**<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε σε κάθε κλήση του template να προσπελάζουμε τον επόμενο κόμβο.
			(3)	**<xsl:param name="accum" select="1"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.
7	Μέσος όρος	"aggr:avg"	**(1)**	**<xsl:param name="nodes" select="/.." />** Το σύνολο των κόμβων που θα αθροιστούν. Κατά την κλήση του template, μια xpath έκφραση δίνεται ως παράμετρος και αντιπροσωπεύει το σύνολο των κόμβων
			(2)	**<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε σε κάθε κλήση του template να προσπελάζουμε τον επόμενο κόμβο.
			(3)	**<xsl:param name="accum" select="0"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.
8	Απόκλιση	"aggr:var"	**(1)**	**<xsl:param name="nodes" select="/.." />** Το σύνολο των κόμβων που θα αθροιστούν. Κατά την κλήση του template, μια xpath έκφραση δίνεται ως παράμετρος και αντιπροσωπεύει το σύνολο των κόμβων
			(2)	**<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε σε κάθε κλήση του template να προσπελάζουμε τον επόμενο κόμβο.
			(3)	**<xsl:param name="accum" select="0"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.

γ.2 Generic Template

Generic Template			
Βασικό template			
Α/α	Λειτουργία	Όνομα template	Παράμετροι
-	Βασικός αλγόριθμος επιλογής και εκτέλεσης των συναθροιστικών συναρτήσεων	"generic:aggregation"	**(1)** **<xsl:param name="nodes"/>** Το σύνολο των κόμβων που θα αθροιστούν. Κατά την κλήση του template, μια xpath έκφραση δίνεται ως παράμετρος και αντιπροσωπεύει το σύνολο των κόμβων. **(2)** **<xsl:param name="aggr-func" select=" 'sum' "/>** Η παράμετρος αυτή αντιπροσωπεύει την συναθροιστική συνάρτηση που θα χρησιμοποιηθεί **(3)** **<xsl:param name="func" select=" 'identity' "/>** Η παράμετρος αυτή αντιπροσωπεύει την πρωτογενή συνάρτηση που θα εφαρμοστεί στην μεταβλητή X. **(4)** **<xsl:param name="func-param1" select="$generic:generics-func[self::generic:func and @name = $func]/@param1"/>** Σε μερικές λειτουργίες χρειάζεται και μια δεύτερη παράμετρος, όπως για παράδειγμα στην μείωση ή αύξηση της μεταβλητής X. **(5)** **<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε σε κάθε κλήση του template να προσπελάζουμε τον επόμενο κόμβο. **(6)** **<xsl:param name="accum" select="$generic:generics-aggr-func[self::generic:aggr-func and @name = $aggr-func]/@identity"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.
Πρωτογενής Συναρτήσεις			
1	Ταυτότητα	"identity"	**(1)** **<xsl:param name="x"/>** Η παράμετρος X είναι η τιμή της μεταβλητής X και επιστρέφεται με την ίδια τιμή, δηλαδή X.
2	Τετράγωνο	"square"	**(1)** **<xsl:param name="x"/>** Η παράμετρος X είναι η τιμή της μεταβλητής X και επιστρέφεται η τιμή στο τετράγωνο X^2.
3	Κύβος	"cube"	**(1)** **<xsl:param name="x"/>** Η παράμετρος X είναι η τιμή της μεταβλητής X και επιστρέφεται η τιμή στον κύβο X^3.
4	Αύξηση	"incr"	**(1)** **<xsl:param name="x"/>** Η παράμετρος X είναι η τιμή της μεταβλητής X και επιστρέφεται η τιμή αυξημένη κατά την τιμή της 2ης παραμέτρου. **(2)** **<xsl:param name="param1" select="@param1"/>** Η παράμετρος αυτή έχει την τιμή κατά την οποία θα αυξηθεί η μεταβλητή X.
5	Μείωση	"decr"	**(1)** **<xsl:param name="x"/>** Η παράμετρος X είναι η τιμή της μεταβλητής X και επιστρέφεται η τιμή μειωμένη κατά την τιμή της 2ης παραμέτρου. **(2)** **<xsl:param name="param1" select="@param1"/>** Η παράμετρος αυτή έχει την τιμή κατά την οποία θα μειωθεί η μεταβλητή X.
6	Αντίστροφο	"reciprocal"	**(1)** **<xsl:param name="x"/>** Η παράμετρος X είναι η τιμή της μεταβλητής X και επιστρέφεται η τιμή 1/X.
Συναθροιστικές Συναρτήσεις			
1	Άθροισμα	“sum”	**(1)** **<xsl:param name="x"/>** Η τιμή της μεταβλητής X που θα προστεθεί στο συσσωρευτή accum. **(2)** **<xsl:param name="accum"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates
2	Μέσος όρος	“avg”	**(1)** **<xsl:param name="x"/>**

			(2) **(3)**	Η τιμή της μεταβλητής X που θα προστεθεί στο συσσωρευτή accum. **<xsl:param name="index" select="1"/>** Η παράμετρος index χρειάζεται για να μπορούμε σε κάθε κλήση του template να υπολογίσουμε την ποσοστιαία συμβολή της μεταβλητής X στην μέση τιμή. **<xsl:param name="accum" select="0"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates.
3	Γινόμενο	“prod”	**(1)** **(2)**	**<xsl:param name="x"/>** Η τιμή της μεταβλητής X που θα πολλαπλασιαστεί με τον συσσωρευτή accum. **<xsl:param name="accum"/>** Η παράμετρος accum χρησιμοποιείται ως συσσωρευτής και κρατάει το τρέχον αποτέλεσμα μεταξύ των διαδοχικών κλήσεων των templates
4	Ελάχιστο	“min”	**(1)** **(2)**	**<xsl:param name="x"/>** Η τιμή της μεταβλητής X που θα ελεχθεί αν είναι . η ελάχιστη. **<xsl:param name="accum"/>** Η παράμετρος accum κρατάει την τρέχουσα ελάχιστη τιμή.
5	Μέγιστο	“max”	**(1)** **(2)**	**<xsl:param name="x"/>** Η τιμή της μεταβλητής X που θα ελεχθεί αν είναι . η μεγιστη. **<xsl:param name="accum"/>** Η παράμετρος accum κρατάει την τρέχουσα μέγιστη τιμή.

δ. Εισαγωγή βιβλιοθήκης σε εφαρμογές XSLT

Ο τρόπος με τον οποίο μπορεί κανείς να χρησιμοποιήσει την βιβλιοθήκη είναι πολύ απλός, φτάνει να εισάγει στο stylesheet της εφαρμογής του τα παρακάτω:

- Το namespace *aggr* το οποίο χρησιμοποιείται για την κλήση των *named templates* όπου το κάθε ένα αντιστοιχεί σε μια λειτουργία. Για να το εισάγουμε πρέπει να συμπεριλάβουμε το ακόλουθο attribute στο element του stylesheet:

```
xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
```

- Το namespace *generic* το οποίο χρησιμοποιείται για την κλήση του *generic template*. Για να το εισάγουμε πρέπει να συμπεριλάβουμε το ακόλουθο attribute στο element του stylesheet:

```
xmlns:generic="http://www.ece.ntua.gr/exslt/generic"
```

- Για να μην γραφτούν τα ακρωνύμια των δυο πιο πάνω namespaces στην έξοδο, πρέπει να συμπεριλάβουμε στο element του stylesheet το ακόλουθο attribute:

```
extension-element-prefixes="aggr generic"
```

- Το τελευταίο που πρέπει να συμπεριλάβουμε στο stylesheet της εφαρμογής μας είναι η εισαγωγή του *importing stylesheet* "aggregation.xsl". Το path αντιστοιχεί στην τοποθεσία του αρχείου "aggregation.xsl" και μπορεί να είναι σχετικό ως προς την τοποθεσία της εφαρμογής μας είτε απόλυτό.

```
<xsl:import href="path" />
```

Συνεπώς μια εφαρμογή θα μπορούσε να χρησιμοποιήσει την βιβλιοθήκη, αν έμοιαζε με το παρακάτω τμήμα κώδικα, όπου με έντονη γραμματοσειρά είναι τα στοιχεία τα οποία πρέπει να εισαχθούν:

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        xmlns:generic="http://www.ece.ntua.gr/exslt/generic"
        extension-element-prefixes="aggr generic">

        <xsl:import href="aggregation.xsl" />
```

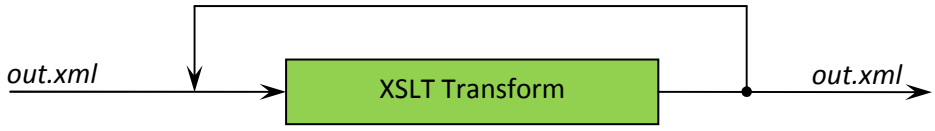
4. Άλλα θέματα

Οι δυνατότητες της XSLT είναι πάρα πολλές και ποικίλουν ανάλογα με το πώς θέλει κανείς να την χρησιμοποιήσει. Όπως έχουμε δει και πιο πάνω, έχουμε χρησιμοποιήσει την XSLT για να μορφοποιήσουμε ιστοσελίδες γραμμένες σε HTML (η οποία είναι και η κυριότερη χρήση), την έχουμε χρησιμοποιήσει για να εκτελέσουμε συναθροιστικές λειτουργίες όπως εύρεση μέσου όρου στις τιμές ενός συνόλου από κόμβους σε αρχεία xml. Υπάρχουν και άλλες λειτουργίες όπως η δημιουργία αρχείων pdf, jpeg, zip archives ή αρχεία με sql ή java source code.

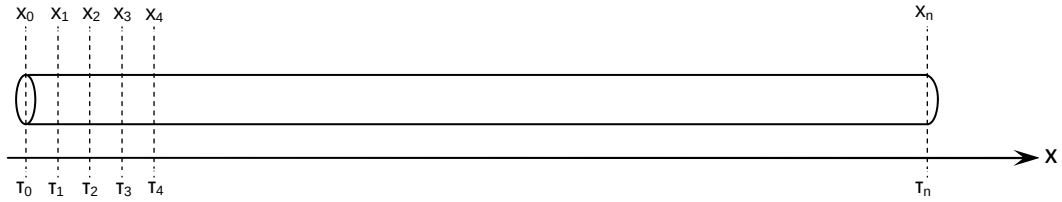
Ο σκοπός για τον οποίο συντάχθηκε η XSLT είχε ως αποτέλεσμα να έχει μειωμένες υπολογιστικές δυνατότητες ώστε να μην προτιμάτε σε προβλήματα με μαθηματικής και φυσικής. Παρόλο αυτό, οι υπολογιστικές δυνατότητες υπάρχουν και παρακάτω θα γίνει μια μικρή αναφορά σε δοκιμές που έγιναν για εύρεση λύσης σε κάποια προβλήματα στα οποία χρησιμοποιήθηκε η XSLT για την επίλυση τους.

Σε πρώτο στάδιο δοκιμάσαμε να θέσουμε ένα αρχείο xml σαν είσοδο(βλ. σημείωση) σε ένα μετασχηματισμό και η έξοδος του μετασχηματισμού να γραφτεί στο ίδιο αρχείο xml όπως το πιο κάτω σχήμα:

Πρέπει να σημειωθεί ότι ο μετασχηματισμός XSLT που χρησιμοποιήθηκε δίνει στην έξοδο ένα αρχείο xml με την ίδια δομή. Οπότε, θα μπορούσαμε να θεωρήσουμε ότι το αρχείο xml περιγράφει μια κατάσταση η οποία αλλάζει όταν εκτελείται ένας μετασχηματισμός XSLT επί αυτού και το νέο αρχείο θα περιγράφει την νέα κατάσταση. Που θα μπορούσε να χρησιμοποιηθεί ένα τέτοιο σενάριο? Αν κάθε μετασχηματισμός XSLT εκφράζει μια μεταβολή κάποιου μεγέθους ΔΜ όταν μεσολαβήσει ένα χρονικό διάστημα ΔΤ, θα μπορούσαμε να χρησιμοποιήσουμε την XSLT σε συνδυασμό με αρχεία xml για να περιγράψουμε ένα φυσικό φαινόμενο το οποία μεταβάλλεται με τον χρόνο. Στην περίπτωση αυτή το πιο πάνω σχήμα αλλάζει καθώς ο μετασχηματισμός XSLT πρέπει να εκτελεστεί πολλές φορές, και το σενάριο αλλάζει όπως παρακάτω:



Σαν κατάσταση ενός προβλήματος, δηλαδή το αρχείο xml, θα μπορούσαμε να θεωρήσουμε την θερμοκρασία μιας ράβδου, τ_0, τ_1,…, τ_n, στα διάφορα σημεία x_0, x_1,…, x_n, της όπως το παρακάτω σχήμα.



Η περιγραφή αυτή της κατάστασης θα μπορούσε να αντιστοιχεί στο πιο κάτω xml αρχείο, στο οποίο οι κόμβοι <value> αντιστοιχούν σε ένα σημείο της ράβδου και η τιμή του κόμβου στην τιμή του κάθε σημείου:

```
<!—- rod-temp-values.xml -->
<?xml version="1.0" encoding="UTF-8" ?>
<values>
        <value>10</value>
        <value>15</value>
        <value>12</value>
        <value>18</value>
        <value>17</value>
        <value>24</value>
        <value>33</value>
        <value>23</value>
        <value>21</value>
        <value>15</value>
</values>
```

Η επαναληπτική εφαρμογή του μετασχηματισμού XSLT στο πιο πάνω αρχείο προσομοιώνει την διάδοση της θερμότητας διαμέσου μιας ράβδου. Η τιμή για κάθε σημείο προκύπτει από τον μέσο όρο των τιμών της θερμοκρασίας των δυο γειτονικών σημείων. Δηλαδή για το σημείο x_k η θερμοκρασία του $\tau_\kappa = (\tau_{\kappa-1} + \tau_{\kappa+1})/2$. Όσο αφορά τα δυο άκρα, διακρίνουμε δυο περιπτώσεις. Η 1η περίπτωση είναι η διάδοση θερμοκρασίας σε ράβδο της οποίας τα άκρα διατηρούν την θερμοκρασία τους. Στην περίπτωση αυτή ένα "κύμα θερμότητας" διαδίδεται δια μέσου της ράβδου. Η 2η περίπτωση είναι η διάχυση της θερμότητας των σημείων

της ράβδου ομοιόμορφα σε αυτήν. Τα σημεία των άκρων δεν διατηρούν την θερμοκρασία τους αλλά μεταβάλλεται και η τιμή της θερμοκρασίας των άκρων προκύπτει από τον μέσο όρο των τιμών της θερμοκρασίας του ακρινού σημείου και του γειτονικού σημείου, είναι δηλαδή για το σημείο x_0 η θερμοκρασία $\tau_0 = (\tau_0 + \tau_1)/2$.

Αν και δεν έχει ιδιαίτερη σημασία ο μετασχηματισμός XSLT που χρησιμοποιήθηκε καθώς είναι πολύ συγκεκριμένος καθώς φτιάχτηκε στα μέτρα του προβλήματος, παρατίθεται ο κώδικας του. Υπενθυμίζεται ότι ο μετασχηματισμός αφήνει αμετάβλητη την δομή του.

```
<!—- values-xsl.xsl -->
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

        <xsl:output method="xml" indent="yes"/>

        <xsl:template match="/">
            <xsl:text>&#10;</xsl:text> <!-- linefeed -->
            <xsl:choose>
                <xsl:when test="count(values/*) > 2">
                        <values>
                        <xsl:call-template name="action">
                                <xsl:with-param name="index" select="1"/>
                                <xsl:with-param name="last" select="count(values/*)"/>
                                <xsl:with-param name="nodes" select="values"/>
                        </xsl:call-template>
                        </values>
                </xsl:when>
                <xsl:otherwise>
                        <xsl:copy-of select="."/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:template>

        <xsl:template name="action">
                <xsl:param name="index"/>
                <xsl:param name="last"/>
                <xsl:param name="nodes"/>
                <xsl:if test="$index &lt;= $last">
                        <value>
                        <xsl:choose>
                                <xsl:when test="$index = 1">
                                        <xsl:value-of select="($nodes/value[$index])"/>
                                </xsl:when>
                                <xsl:when test="$index = $last">
                                        <xsl:value-of select="($nodes/value[$index])"/>
                                </xsl:when>
```

```
<!—- values-xsl.xsl  (συνέχεια) -->
                                <xsl:otherwise>
                                        <xsl:value-of select="($nodes/value[$index - 1] +
$nodes/value[$index + 1]) div 2"/>
                                </xsl:otherwise>
                        </xsl:choose>
                        </value>
                        <xsl:call-template name="action">
                                <xsl:with-param name="index" select="$index+1"/>
                                <xsl:with-param name="last" select="$last"/>
                                <xsl:with-param name="nodes" select="$nodes"/>
                        </xsl:call-template>
                </xsl:if>
        </xsl:template>
</xsl:stylesheet>
```

Το πιο πάνω stylesheet είναι για την περίπτωση όπου η θερμοκρασία των άκρων παραμένει σταθερή. Για την περίπτωση όπου τα άκρα μεταβάλλουν την θερμοκρασία τους το αντίστοιχο stylesheet είναι σχεδόν το ίδιο, το μόνο που αλλάζει είναι οι δυο εντολές που είναι με έντονη (bold) γραμματοσειρά. Οι δυο εντολές που αντικαθιστούν τις προηγούμενες εντολές είναι:

```
<xsl:value-of select="($nodes/value[$index] + $nodes/value[$index + 1]) div 2"/>
```

για το 1ο σημείο της ράβδου x_0 και για το τελευταίο σημείο της ράβδου x_n είναι:

```
<xsl:value-of select="($nodes/value[$index] + $nodes/value[$index - 1]) div 2"/>
```

Ο πιο πάνω μετασχηματισμός XSLT εφαρμόστηκε επί του αρχικού αρχείου αρκετές φορές και τα αντίστοιχα αποτελέσματα για τις δυο περιπτώσεις είναι τα παρακάτω:

- Θερμοκρασία σταθερή στα δυο άκρα:

```
<!-- out.xml  -->
 <?xml version="1.0" encoding="UTF-8"?>
<values>
        <value>10</value>
        <value>10.717658661953534</value>
        <value>11.423875998174164</value>
        <value>12.077128609460015</value>
        <value>12.70142168558869</value>
        <value>13.244542178171564</value>
        <value>13.754750125078637</value>
        <value>14.1935501828847</value>
        <value>14.610879020747861</value>
        <value>15</value>
</values>
```

Οι τιμές της θερμοκρασίας για τα σημεία της ράβδου κυμαίνονται όπως είναι λογικό μεταξύ των δυο τιμών της θερμοκρασίας στα δυο άκρα (10 και 15)

- Θερμοκρασία των δυο άκρων ελεύθερη:

```
<!-- out.xml  -->
 <?xml version="1.0" encoding="UTF-8"?>
<values>
        <value>18.38417653122375</value>
        <value>18.41904125655887</value>
        <value>18.50827966611738</value>
        <value>18.599429272242304</value>
        <value>18.743884990177555</value>
        <value>18.856297239825196</value>
        <value>19.00083322572118</value>
        <value>19.0915704602546</value>
        <value>19.180938746101788</value>
        <value>19.215548611777365</value>
</values>
```

Οι τιμές της θερμοκρασίας στα σημεία της ράβδου τείνουν στον μέσο όρο της θερμοκρασίας όλων των σημείων πριν την έναρξη της διάχυσης της θερμότητας. Στην προκειμένη περίπτωση ο μέσος όρος ήταν 18.8, και όντως η θερμοκρασία των σημείων τείνει προς αυτή την τιμή.

Θα μπορούσε αυτό το είδος προγραμματισμού να επεκταθεί? Όπως είδαμε στο συγκεκριμένο πρόβλημα μελετήθηκαν δυο περιπτώσεις αλλά οι περιπτώσεις είναι πάρα πολλές. Θα μπορούσε να γραφεί μια βιβλιοθήκη από templates ή ένα generic template τα οποία να επιλύουν μερικές διαφορικές εξισώσεις όπως και στο παράδειγμα μας? Αν ο τρόπος επίλυσης κάποιου προβλήματος εμπεριέχει υποπεριπτώσεις θα μπορούσε να χρησιμοποιηθεί generic programming για την επίλυση του όπως αυτό παρουσιάστηκε σε προηγούμενο κεφάλαιο. Ποίες θα ήταν αυτές οι υποπεριπτώσεις? Θα μπορούσαν να είναι για παράδειγμα, διάδοση θερμοκρασίας όχι σε μια ράβδο αλλά σε ένα επίπεδο ή ένα κύβο. Ή θα μπορούσε να είναι διάδοση θερμότητας σε μια ράβδο της οποίας μόνο το ένα άκρο διατηρεί την θερμοκρασία του ενώ του άλλου άκρου η θερμοκρασία μεταβάλλεται ελεύθερα.

Με άλλα λόγια θα μπορούσαμε να έχουμε ένα generic template το οποίο θα δέχεται παραμέτρους, π.χ. σχήμα σώματος (ράβδος, επίπεδο, κύβος), συμπεριφορά στα άκρα(σταθερή τιμή, μεταβαλλόμενη τιμή) και άλλες παραμέτρους ανάλογα με το πρόβλημα. Για παράδειγμα θα μπορούσαμε να έχουμε την ακόλουθη κλήση του generic template που υποθέσαμε:

```
<xsl:call-template name="Διάδοση-θερμοκρασίας">
        <xsl:with-param name="shape" select="ράβδος"/>
        <xsl:with-param name="on-left-edge" select="fixed-value or free-value"/>
        <xsl:with-param name="on-right-edge" select="fixed-value or free-value"/>
        <xsl:with-param name="nodes" select="τα-σημεία-της-ράβδου ή αναφορά σε αρχείο"/>
</xsl:call-template>
```

Σημ. Όσο αφορά την είσοδο ενός αρχείου σε ένα μετασχηματισμό XSLT το οποίο είναι ταυτόχρονα και έξοδος του μετασχηματισμού υπάρχει θέμα συμβατότητας με τον XSLT processor. Στην αγορά κυκλοφορούν αρκετές υλοποιήσεις XSLT processor αλλά δεν υποστηρίζουν όλοι αυτή την λειτουργία. Στο παράδειγμα μας χρησιμοποιήθηκε ο xalan C++ ο οποίος υποστηρίζει αυτή την λειτουργία. Δοκιμάστηκε επίσης και ο xalan – j οποίος όπως αποδείχτηκε δεν υποστηρίζει αυτή την λειτουργία.

5. Παραρτήματα

α. Παράρτημα Α – Κώδικας Βιβλιοθήκης

Το πιο κάτω stylesheet ("aggregation.xsl") είναι το βασικό stylesheet το οποίο κάνουμε import στις εφαρμογές μας. Το stylesheet αυτό με την σειρά του εισάγει ένα – ένα τα κομμάτια της βιβλιοθήκης που αντιστοιχούν στις διάφορες συναθροιστικές λειτουργίες.

```
<!-- aggregation.xsl -->
<?xml version="1.0" encoding="UTF-8" ?>

<!--    Αυτό το stylesheet είναι που κάνουμε import για να μπορέσουμε να χρησιμοποιήσουμε
        τις συναθροιστικές λειτουργίες που έχουμε δημιουργήσει. Το stylesheet αποτελείται
        από 2 μέρη, Α και Β. Στο μέρος Α εισάγονται τα αρχεία τα οποία περιέχουν τα templates
        τα οποία μπορούν να κληθούν με το όνομα τους κατευθείαν και να εκτελέσουν μια από
        τις λειτουργίες sum, avg, prod,  και var. Στο μέρος Β υπάρχει το template το οποίο
        καλείται με παράμετρο το όνομα της λειτουργίας που θέλουμε να εκτελέσουμε  -->

<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation" version="1.0"
        extension-element-prefixes="aggr">

        <!-- ΜΕΡΟΣ Α -->
        <import href="functions/avg/aggr.avg.xsl"/>
        <import href="functions/prod/aggr.prod.xsl"/>
        <import href="functions/prod-reciprocal/aggr.prod-reciprocal.xsl"/>
        <import href="functions/sum/aggr.sum.xsl"/>
        <import href="functions/sum-square/aggr.sum-square.xsl"/>
        <import href="functions/sum-cube/aggr.sum-cube.xsl"/>
        <import href="functions/sum-reciprocal/aggr.sum-reciprocal.xsl"/>
        <import href="functions/var/aggr.var.xsl"/>

        <!-- ΜΕΡΟΣ Β -->
        <import href="functions/generic/aggr.generic.xsl"/>

</stylesheet>
```

α.1 Κώδικας Βιβλιοθήκης – Μέρος Α

Στο παράρτημα αυτό παρατίθενται οι λειτουργίες οι οποίες μπορούν να κληθούν με το όνομα τους. Ο τρόπος υλοποίησης τους έγινε στα πρότυπα των extensions όπως αυτά εκδίδονται από τους διάφορους οργανισμούς. Κάθε λειτουργία αποτελείται από ένα importing template το οποίο εισάγει με την σειρά τους διαφορετικούς τρόπους υλοποίησης της λειτουργίας. Μια λειτουργία θα μπορούσε να υλοποιηθεί με ένα named template, με μια συνάρτηση ή ακόμη και με script (π.χ. JavaScript). Στην περίπτωση της βιβλιοθήκης που έχουμε δημιουργήσει υπάρχουν για κάθε λειτουργία δυο templates, το importing template και τον named template το οποίο εκτελεί την αντίστοιχη λειτουργία. Επειδή δεν έχει νόημα η παράθεση του importing template, πιο κάτω θα παρατεθεί μόνο το template που υλοποιεί την αντίστοιχη λειτουργία.

sum

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
	xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
	xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
	exclude-result-prefixes="aggr">

<xsl:template name="aggr:sum">

	<xsl:param name="nodes" select="/.." />
	<xsl:param name="index" select="1"/>
	<xsl:param name="accum" select="0"/>

	<xsl:choose>
		<xsl:when test="not($nodes)">Nan</xsl:when>
		<xsl:otherwise>
			<xsl:variable name="currentNode">
				<xsl:value-of select="$nodes[$index]"/>
			</xsl:variable>
			<xsl:variable name="remainingNodes">
				<xsl:choose>
				    <xsl:when test="$index=count($nodes)">
				        <xsl:text>0</xsl:text>
				    </xsl:when>
				    <xsl:otherwise>
				        <xsl:call-template name="aggr:sum">
				            <xsl:with-param name="nodes" select="$nodes"/>
				            <xsl:with-param name="index" select="$index + 1"/>
				            <xsl:with-param name="accum" select="$accum +
$currentNode"/>
				        </xsl:call-template>
				    </xsl:otherwise>
				</xsl:choose>
			</xsl:variable>
			<xsl:value-of select="$currentNode + $remainingNodes"/>
		</xsl:otherwise>
	</xsl:choose>
</xsl:template>
```

sum square

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
```

```
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        exclude-result-prefixes="aggr">

<xsl:template name="aggr:sum-square">

        <xsl:param name="nodes" select="/.." />
        <xsl:param name="index" select="1"/>
        <xsl:param name="accum" select="0"/>

        <xsl:choose>
                <xsl:when test="not($nodes)">Nan</xsl:when>
                <xsl:otherwise>
                        <xsl:variable name="currentNode">
                                <xsl:value-of select="$nodes[$index] * $nodes[$index]"/>
                        </xsl:variable>
                        <xsl:variable name="remainingNodes">
                                <xsl:choose>
                                        <xsl:when test="$index=count($nodes)">
                                                <xsl:text>0</xsl:text>
                                        </xsl:when>
                                        <xsl:otherwise>
                                                <xsl:call-template name="aggr:sum-square">
                                                        <xsl:with-param name="nodes"
select="$nodes"/>
                                                        <xsl:with-param name="index"
select="$index + 1"/>
                                                        <xsl:with-param name="accum"
select="$accum + $currentNode"/>
                                                </xsl:call-template>
                                        </xsl:otherwise>
                                </xsl:choose>
                        </xsl:variable>
                        <xsl:value-of select="$currentNode + $remainingNodes"/>
                </xsl:otherwise>
        </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

sum cube

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        exclude-result-prefixes="aggr">

<xsl:template name="aggr:sum-cube">

        <xsl:param name="nodes" select="/.." />
        <xsl:param name="index" select="1"/>
        <xsl:param name="accum" select="0"/>

        <xsl:choose>
                <xsl:when test="not($nodes)">Nan</xsl:when>
                <xsl:otherwise>
                        <xsl:variable name="currentNode">
                                <xsl:value-of select="$nodes[$index] * $nodes[$index] *
$nodes[$index]"/>
                        </xsl:variable>
                        <xsl:variable name="remainingNodes">
                                <xsl:choose>
                                        <xsl:when test="$index=count($nodes)">
                                                <xsl:text>0</xsl:text>
                                        </xsl:when>
                                        <xsl:otherwise>
                                                <xsl:call-template name="aggr:sum-cube">
                                                        <xsl:with-param name="nodes"
select="$nodes"/>
                                                        <xsl:with-param name="index"
select="$index + 1"/>
                                                        <xsl:with-param name="accum"
select="$accum + $currentNode"/>
                                                </xsl:call-template>
                                        </xsl:otherwise>
                                </xsl:choose>
```

```
                        </xsl:variable>
                        <xsl:value-of select="$currentNode + $remainingNodes"/>
                </xsl:otherwise>
        </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

sum reciprocal

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        exclude-result-prefixes="aggr">

<xsl:template name="aggr:sum-reciprocal">

        <xsl:param name="nodes" select="/.." />
        <xsl:param name="index" select="1"/>
        <xsl:param name="accum" select="0"/>

        <xsl:choose>
                <xsl:when test="not($nodes)">Nan</xsl:when>
                <xsl:otherwise>
                        <xsl:variable name="currentNode">
                                <xsl:value-of select="1 div $nodes[$index]"/>
                        </xsl:variable>
                        <xsl:variable name="remainingNodes">
                                <xsl:choose>
                                        <xsl:when test="$index=count($nodes)">
                                                <xsl:text>0</xsl:text>
                                        </xsl:when>
                                        <xsl:otherwise>
                                                <xsl:call-template name="aggr:sum-reciprocal">
                                                        <xsl:with-param name="nodes"
select="$nodes"/>
                                                        <xsl:with-param name="index"
select="$index + 1"/>
                                                        <xsl:with-param name="accum"
select="$accum + $currentNode"/>
                                                </xsl:call-template>
                                        </xsl:otherwise>
                                </xsl:choose>
                        </xsl:variable>
                        <xsl:value-of select="$currentNode + $remainingNodes"/>
                </xsl:otherwise>
        </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

average

```
<!-- aggr.avg.template.xsl -->
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        exclude-result-prefixes="aggr">

<xsl:template name="aggr:avg">

        <xsl:param name="nodes" select="/.." />
        <xsl:param name="index" select="1"/>
        <xsl:param name="accum" select="0"/>
        <xsl:variable name="nodeNumber" select="count($nodes)"/>
        <xsl:choose>
                <xsl:when test="not($nodes)">Nan</xsl:when>
                <xsl:otherwise>
                        <xsl:variable name="currentNode">
                                <xsl:value-of select="$nodes[$index] div $nodeNumber"/>
                        </xsl:variable>
                        <xsl:variable name="remainingNodes">
                                <xsl:choose>
```

```
                                        <xsl:when test="$index=count($nodes)">
                                                <xsl:text>0</xsl:text>
                                        </xsl:when>
                                        <xsl:otherwise>
                                                <xsl:call-template name="aggr:avg">
                                                        <xsl:with-param name="nodes"
select="$nodes"/>
                                                        <xsl:with-param name="index"
select="$index + 1"/>
                                                        <xsl:with-param name="accum"
select="$accum + $currentNode"/>
                                                </xsl:call-template>
                                        </xsl:otherwise>
                                </xsl:choose>
                        </xsl:variable>
                        <xsl:value-of select="($currentNode + $remainingNodes)"/>
                </xsl:otherwise>
        </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

product

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        exclude-result-prefixes="aggr">

<xsl:template name="aggr:prod">

        <xsl:param name="nodes" select="/.." />
        <xsl:param name="index" select="1"/>
        <xsl:param name="accum" select="1"/>
        <xsl:choose>
                <xsl:when test="not($nodes)">Nan</xsl:when>
                <xsl:otherwise>
                        <xsl:variable name="currentNode">
                                <xsl:value-of select="$nodes[$index]"/>
                        </xsl:variable>
                        <xsl:variable name="remainingNodes">
                                <xsl:choose>
                                        <xsl:when test="$index=count($nodes)">
                                                <xsl:text>1</xsl:text>
                                        </xsl:when>
                                        <xsl:otherwise>
                                                <xsl:call-template name="aggr:prod">
                                                        <xsl:with-param name="nodes"
select="$nodes"/>
                                                        <xsl:with-param name="index"
select="$index + 1"/>
                                                        <xsl:with-param name="accum"
select="$accum * $currentNode"/>
                                                </xsl:call-template>
                                        </xsl:otherwise>
                                </xsl:choose>
                        </xsl:variable>
                        <xsl:value-of select="($currentNode * $remainingNodes)"/>
                </xsl:otherwise>
        </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

product reciprocal

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        exclude-result-prefixes="aggr">

<xsl:template name="aggr:prod-reciprocal">

        <xsl:param name="nodes" select="/.." />
```

```
        <xsl:param name="index" select="1"/>
        <xsl:param name="accum" select="1"/>

        <xsl:choose>
                <xsl:when test="not($nodes)">Nan</xsl:when>
                <xsl:otherwise>
                        <xsl:variable name="currentNode">
                                <xsl:value-of select="1 div $nodes[$index]"/>
                        </xsl:variable>
                        <xsl:variable name="remainingNodes">
                                <xsl:choose>
                                        <xsl:when test="$index=count($nodes)">
                                                <xsl:text>1</xsl:text>
                                        </xsl:when>
                                        <xsl:otherwise>
                                                <xsl:call-template name="aggr:prod-reciprocal">
                                                        <xsl:with-param name="nodes"
select="$nodes"/>
                                                        <xsl:with-param name="index"
select="$index + 1"/>
                                                        <xsl:with-param name="accum"
select="$accum * $currentNode"/>
                                                </xsl:call-template>
                                        </xsl:otherwise>
                                </xsl:choose>
                        </xsl:variable>
                        <xsl:value-of select="$currentNode * $remainingNodes"/>
                </xsl:otherwise>
        </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

variance

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        exclude-result-prefixes="aggr">

<xsl:template name="aggr:var">

        <xsl:param name="nodes" select="/.." />
        <xsl:param name="index" select="1"/>
        <xsl:param name="accum" select="0"/>
        <xsl:variable name="nodeNumber" select="count($nodes)"/>
        <xsl:variable name="nodesAvg">
                <xsl:call-template name="aggr:avg">
                   <xsl:with-param name="nodes" select="$nodes"/>
                </xsl:call-template>
        </xsl:variable>
        <xsl:choose>
                <xsl:when test="not($nodes)">Nan</xsl:when>
                <xsl:otherwise>
                        <xsl:variable name="currentNode">
                                <xsl:value-of select="($nodes[$index] - $nodesAvg) *
($nodes[$index] - $nodesAvg) div $nodeNumber"/>
                        </xsl:variable>
                        <xsl:variable name="remainingNodes">
                                <xsl:choose>
                                        <xsl:when test="$index=count($nodes)">
                                                <xsl:text>0</xsl:text>
                                        </xsl:when>
                                        <xsl:otherwise>
                                                <xsl:call-template name="aggr:var">
                                                        <xsl:with-param name="nodes"
select="$nodes"/>
                                                        <xsl:with-param name="index"
select="$index + 1"/>
                                                        <xsl:with-param name="accum"
select="$accum + $currentNode"/>
                                                </xsl:call-template>
                                        </xsl:otherwise>
                                </xsl:choose>
                        </xsl:variable>
```

```
                        <xsl:value-of select="($currentNode + $remainingNodes)"/>
                </xsl:otherwise>
        </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

α.2 Κώδικας Βιβλιοθήκης – Μέρος Β (generic templates)

Το πιο κάτω template, είναι ένα generic template το οποίο αποτελεί ένα αλγόριθμο επιλογής και εκτέλεσης της κατάλληλης συναθροιστικής λειτουργίας που εισάγει ο χρήστης ως παράμετρο.

```
<!-- aggr.generic.xsl -->
<?xml version="1.0" encoding="UTF-8"?>
<!--    Το template αυτό περιέχει τον βασικό αλγόριθμο εκτέλεσης των συναθροιστικών
Λειτουργιών. Τα βασικά στοιχεία του αλγόριθμου αυτού είναι οι πρωτογενής συναρτήσεις που δρουν
στην μεταβλητή Χ και οι συναθροιστικές λειτουργίες στην μεταβλητή Χ. Τα δυο αυτά στοιχεία
είναι αποθηκευμένα σε δυο ξεχωριστά αρχεία και εισάγονται στην αρχή αυτού του stylesheet. -->

<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"  version="1.0"
        xmlns:generic="http://www.ece.ntua.gr/exslt/generic"
        extension-element-prefixes="aggr">
        <!-- Εισαγωγή πρωτογενών συναρτήσεων που δρουν στην μεταβλητή Χ -->
        <import href="aggr.generic.functions.templates.xsl"/>
        <!-- Εισαγωγή συναθροιστικών συναρτήσεων που δρουν στην μεταβλητή Χ -->
        <import href="aggr.generic.aggregate-functions.templates.xsl"/>

        <xsl:variable name="generic:generics-func" select="$generic:generics-func-local"/>
        <xsl:variable name="generic:generics-aggr-func" select="$generic:generics-aggr-func-
local"/>

        <!-- Το παρακάτω template είναι ο βασικός αλγόριθμος εκτέλεσης των συναθροιστικών
λειτουργιών -->
        <xsl:template name="generic:aggregation">
                <xsl:param name="nodes"/>
                <xsl:param name="aggr-func" select=" 'sum' "/>
                <xsl:param name="func" select=" 'identity' "/>
                <xsl:param name="func-param1" select="$generic:generics-func[self::generic:func
and @name = $func]/@param1"/>
                <xsl:param name="index" select="1"/>
                <xsl:param name="accum" select="$generic:generics-aggr-func[self::generic:aggr-
func and @name = $aggr-func]/@identity"/>

                <xsl:choose>
                        <xsl:when test="$nodes">
                                <xsl:variable name="f-of-x">
                                        <xsl:apply-templates select="$generic:generics-
func[self::generic:func and @name = $func]">
                                           <xsl:with-param name="x" select="$nodes[1]"/>
                                           <xsl:with-param name="index" select="$index"/>
                                           <xsl:with-param name="param1" select="$func-param1"/>
                                        </xsl:apply-templates>
                                </xsl:variable>
                                <xsl:variable name="temp">
                                        <xsl:apply-templates select="$generic:generics-aggr-
func[self::generic:aggr-func and @name = $aggr-func]">
                                           <xsl:with-param name="x" select="$f-of-x"/>
                                           <xsl:with-param name="accum" select="$accum"/>
                                           <xsl:with-param name="index" select="$index"/>
                                        </xsl:apply-templates>
                                </xsl:variable>
                                <xsl:call-template name="generic:aggregation">
                                   <xsl:with-param name="nodes" select="$nodes[position()!=1]"/>
                                   <xsl:with-param name="aggr-func" select="$aggr-func"/>
                                   <xsl:with-param name="func" select="$func"/>
                                   <xsl:with-param name="func-param1" select="$func-param1"/>
                                   <xsl:with-param name="index" select="$index + 1"/>
                                   <xsl:with-param name="accum" select="$temp"/>
                                </xsl:call-template>
                        </xsl:when>
                        <xsl:otherwise>
                                <xsl:value-of select="$accum"/>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:template>
</stylesheet>
```

Όπως είδαμε και πιο πριν το generic template εισάγει δυο αρχεία. Στο πρώτο αρχείο υπάρχουν οι πρωτογενείς συναρτήσεις που δρουν στην μεταβλητή X και στο δεύτερο αρχείο υπάρχουν οι συναθροιστικές συναρτήσεις που δρουν στην μεταβλητή X.

Πρωτογενής Συναρτήσεις

```
<!-- aggr.generic.functions.templates.xsl -->
<?xml version="1.0"?>
<!--    Αυτό το stylesheet περιέχει τις πρωτογενής συναρτήσεις που δρουν
        στην μεταβλητή X. Δηλαδή, το X, το X τετράγωνο, το X κύβος, η
        αύξηση και μείωση του X, το αντίστροφο του X κ.ά.    -->
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        xmlns:generic="http://www.ece.ntua.gr/exslt/generic"
        exclude-result-prefixes="aggr generic">

        <xsl:variable name="generic:public-generics-funcs" select="document('')/*/generic:*"/>
        <xsl:variable name="generic:generics-func-local" select="$generic:public-generics-
funcs"/>

        <generic:func name="identity"/>
        <xsl:template match="generic:func[@name='identity']">
                <xsl:param name="x"/>
                <xsl:value-of select="$x"/>
        </xsl:template>

        <generic:func name="square"/>
        <xsl:template match="generic:func[@name='square']">
                <xsl:param name="x"/>
                <xsl:value-of select="$x * $x"/>
        </xsl:template>

        <generic:func name="cube"/>
        <xsl:template match="generic:func[@name='cube']">
                <xsl:param name="x"/>
                <xsl:value-of select="$x * $x * $x"/>
        </xsl:template>

        <generic:func name="incr" param1="1"/>
        <xsl:template match="generic:func[@name='incr']">
                <xsl:param name="x"/>
                <xsl:param name="param1" select="@param1"/>
                <xsl:value-of select="$x + $param1"/>
        </xsl:template>

        <generic:func name="decr" param1="1"/>
        <xsl:template match="generic:func[@name='decr']">
                <xsl:param name="x"/>
                <xsl:param name="param1" select="@param1"/>
                <xsl:value-of select="$x - $param1"/>
        </xsl:template>

        <generic:func name="reciprocal"/>
        <xsl:template match="generic:func[@name='reciprocal']">
                <xsl:param name="x"/>
                <xsl:value-of select="1 div $x"/>
        </xsl:template>

</xsl:stylesheet>
```

Συναθροιστικές Συναρτήσεις

```
<!-- aggr.generic.aggregate-functions.templates.xsl -->
<?xml version="1.0"?>
```

```
<!--	Αυτό το template περιέχει τις συναθροιστικές συναρτήσεις που δρουν
	στην μεταβλητή Χ. Δηλαδή, sum, avg, prod, var, min και max. -->
<xsl:stylesheet version="1.0"
	xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
	xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
	xmlns:generic="http://www.ece.ntua.gr/exslt/generic"
	exclude-result-prefixes="aggr generic">

	<!--	Με την  χρήση των δυο πιο κάτω μεταβλητών μπορούν να προσπελαστούν
		οι συναθροιστικές συναρτήσεις που ορίζοται πιο κάτω -->
	<xsl:variable name="generic:public-generics-aggr-funcs"
select="document('')/*/generic:*"/>
	<xsl:variable name="generic:generics-aggr-func-local" select="$generic:public-
generics-aggr-funcs"/>

	<generic:aggr-func name="sum" identity="0"/>
	<xsl:template match="generic:aggr-func[@name='sum']">
		<xsl:param name="x"/>
		<xsl:param name="accum"/>
		<xsl:value-of select="$x + $accum"/>
	</xsl:template>

	<generic:aggr-func name="avg" identity="0"/>
	<xsl:template match="generic:aggr-func[@name='avg']">
		<xsl:param name="x"/>
		<xsl:param name="accum"/>
		<xsl:param name="index"/>
		<xsl:value-of select="(($index - 1) * $accum + $x ) div $index"/>
	</xsl:template>

	<generic:aggr-func name="prod" indentity="1"/>
	<xsl:template match="generic:aggr-func[@name='prod']">
		<xsl:param name="x"/>
		<xsl:param name="accum"/>
		<xsl:value-of select="$x * $accum"/>
	</xsl:template>

	<generic:aggr-func name="min" identity=""/>
	<xsl:template match="generic:aggr-func[@name='min']">
		<xsl:param name="x"/>
		<xsl:param name="accum"/>
		<xsl:choose>
			<xsl:when test="$accum = @identity or $accum >= $x">
				<xsl:value-of select="$x"/>
			</xsl:when>
			<xsl:otherwise>
				<xsl:value-of select="$accum"/>
			</xsl:otherwise>
		</xsl:choose>
	</xsl:template>

	<generic:aggr-func name="max" identity=""/>
	<xsl:template match="generic:aggr-func[@name='max']">
		<xsl:param name="x"/>
		<xsl:param name="accum"/>
		<xsl:choose>
			<xsl:when test="$accum = @identity or $accum &lt; $x">
				<xsl:value-of select="$x"/>
			</xsl:when>
			<xsl:otherwise>
				<xsl:value-of select="$accum"/>
			</xsl:otherwise>
		</xsl:choose>
	</xsl:template>
</xsl:stylesheet>
```

β. Παράρτημα Β – Κώδικας Τελικών Παραδειγμάτων

Πιο κάτω παρουσιάζονται κάποια ολοκληρωμένα παραδείγματα που κάνουν χρήση της βιβλιοθήκης συνάθροισης.

Χρήση της λειτουργίας Average

Το παρακάτω παράδειγμα χρησιμοποιεί την λειτουργία Average της βιβλιοθήκης που δημιουργήσαμε και για ένα συγκεκριμένο φοιτητή υπολογίζει τον τελικό βαθμό πτυχίου και παρουσιάζει τα αποτελέσματα μαθημάτων σε ένα πίνακα.

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
        xmlns:generic="http://www.ece.ntua.gr/exslt/generic"
        extension-element-prefixes="aggr generic">

        <xsl:import href="C:/Aggregation/aggregation.xsl" />
        <xsl:output method="html"/>

        <xsl:template match="/student">
                <html>
                        <head>
                                <title>Grades - Sotiris Nicolaou</title>
                        </head>
                        <body>
                        <h3 style="text-align:center">Student Information</h3>
                        <xsl:apply-templates select="student-info"/>
                        <xsl:apply-templates select="kormos|roes"/>
                        </body>
                </html>
        </xsl:template>

        <xsl:template match="student-info">
                <p>
                        <br/><xsl:text>Name: </xsl:text><xsl:value-of select="name"/>
                        <br/><xsl:text>Student id: </xsl:text><xsl:value-of select="sid"/>
                        <br/><xsl:text>Etos eisagwgis: </xsl:text><xsl:value-of select="entry-
year"/>
                        <br/><u><xsl:text>Contact details</xsl:text></u>
                        <br/><xsl:text>Phones:</xsl:text>
                        <br/><xsl:text>    - Home: </xsl:text><xsl:value-of
select="phones/home"/>
                        <br/><xsl:text>    - Mobile: </xsl:text><xsl:value-of
select="phones/mobile"/>
                        <br/><xsl:text>Address: </xsl:text><xsl:value-of select="address"/>
                        <br/><xsl:text>E-mail: </xsl:text><xsl:value-of select="email"/>
                        <br/><br/><u><xsl:text>Analysh Vathmologias</xsl:text></u>

                        <xsl:variable name="mesos-oros-kormou">
                        <xsl:call-template name="generic:aggregation">
                           <xsl:with-param name="nodes" select="../kormos/course/grade" />
                           <xsl:with-param name="aggr-func" select=" 'avg' "/>
                        </xsl:call-template>
                        </xsl:variable>
                        <xsl:variable name="mesos-oros-rown">
                        <xsl:call-template name="generic:aggregation">
                           <xsl:with-param name="nodes" select="../roes/course/grade" />
                           <xsl:with-param name="aggr-func" select=" 'avg' "/>
                        </xsl:call-template>
                        </xsl:variable>
                        <xsl:variable name="mesos-oros-olwn">
                        <xsl:call-template name="generic:aggregation">
                           <xsl:with-param name="nodes" select="../*/course/grade" />
```

```
                        <xsl:with-param name="aggr-func" select=" 'avg' "/>
                    </xsl:call-template>
                    </xsl:variable>
                    <xsl:variable name="vathmos-diplomatikis">
                            <xsl:value-of select="../diplomatiki/grade"/>
                    </xsl:variable>
                    <br/><xsl:text>Mesos oros mathimatwn kormou: </xsl:text><xsl:value-of
select="$mesos-oros-kormou"/>
                    <br/><xsl:text>Mesos oros mathimatwn rown: </xsl:text><xsl:value-of
select="$mesos-oros-rown"/>
                    <br/><xsl:text>Vathmos diplomatikis: </xsl:text><xsl:value-of
select="$vathmos-diplomatikis"/>
                    <br/><b><xsl:text>Synolikos mesos oros: </xsl:text><xsl:value-of
select="$mesos-oros-olwn * 0.8 + $vathmos-diplomatikis * 0.2"/></b>

            </p>
        </xsl:template>

        <xsl:template match="kormos|roes">
            <br/>
            <table width="100%" border="4" cols="55% 15% 15% 15%">
                    <tr>
                            <th bgcolor="lightgreen" colspan="4"><xsl:value-of
select="@description"/></th>
                    </tr>
                    <tr bgcolor="lightgreen">
                            <th>Course name</th>
                            <th>Semester</th>
                            <th>Grade</th>
                            <th>Status</th>
                    </tr>
                    <xsl:for-each select="course">
                            <tr>
                                    <td>
                                            <b><xsl:value-of select="@name"/></b>
                                    </td>
                                    <td align="center">
                                            <xsl:value-of select="semester"/>
                                    </td>
                                    <td align="center">
                                            <xsl:value-of select="grade"/>
                                    </td>
                                    <td align="center">
                                            <xsl:if test="grade &gt;4">
                                                    <font color="green">passed</font>
                                            </xsl:if>
                                            <xsl:if test="grade &lt; 5">
                                                    <font color="red">failed</font>
                                            </xsl:if>
                                    </td>
                            </tr>
                    </xsl:for-each>
            </table>
            <br/>
        </xsl:template>

</xsl:stylesheet>
```

Χρήση της λειτουργίας sum σε συνδυασμό με την συνάρτηση Document()

Στο πιο κάτω παράδειγμα ο μετασχηματισμός φτιάχνει μια αναφορά πωλήσεων μιας εταιρίας για το έτος 2007 για τις διάφορες γεωγραφικές περιοχές. Σημαντικό στοιχείο είναι ότι τα συνολοικά αθροίσματα προκύπτουν από την επεξεργασία μιας συλλογής από αρχείων τα οποία προσπελάζονται με την συνάρτηση Document().

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:aggr="http://www.ece.ntua.gr/exslt/aggregation"
```

```
        xmlns:generic="http://www.ece.ntua.gr/exslt/generic"
        extension-element-prefixes="aggr generic">

        <xsl:import href="C:/Aggregation/aggregation.xsl" />
        <xsl:output method="html"/>

        <xsl:variable name="tax" select="0.15"/>

        <xsl:variable name="revenues">
                <xsl:call-template name="generic:aggregation">
                        <xsl:with-param name="nodes"
select="document(/report/regions/*/@filename)/region/revenues" />
                </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="items-sold">
                <xsl:call-template name="generic:aggregation">
                        <xsl:with-param name="nodes"
select="document(/report/regions/*/@filename)/region/items-sold" />

                </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="expenses">
                <xsl:call-template name="generic:aggregation">
                        <xsl:with-param name="nodes"
select="document(/report/regions/*/@filename)/region/expenses" />

                </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="profit">
                <xsl:call-template name="generic:aggregation">
                        <xsl:with-param name="nodes"
select="document(/report/regions/*/@filename)/region/profit" />

                </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="profit-after-tax">
                <xsl:call-template name="generic:aggregation">
                        <xsl:with-param name="nodes"
select="document(/report/regions/*/@filename)/region/profit" />

                        <xsl:with-param name="func" select=" 'perc-decr' "/>
                        <xsl:with-param name="func-param1" select="$tax"/>
                </xsl:call-template>
        </xsl:variable>



        <xsl:template match="/report">
                <html>
                        <head>
                                <title><xsl:value-of select="title"/></title>
                        </head>
                        <body>
                                <h1><xsl:value-of select="title"/></h1>
                                <h2>N.T.U.A International Retail Company</h2>
                                <h3>Sales by region for year 2007</h3>
                                <br/><br/>
                                <xsl:apply-templates select="regions"/>
                                <p><b>
                                Total sales: <xsl:value-of select="$revenues"/><br/>
                                Total expenses: <xsl:value-of select="$expenses"/><br/>
                                Profit: <xsl:value-of select="$profit"/><br/>
                                Profit after taxation: <xsl:value-of select="$profit-after-
tax"/><br/>
                                Tax Amount: <xsl:value-of select="$profit - $profit-after-
tax"/><br/></b></p>
                        </body>
                </html>
        </xsl:template>

        <xsl:template match="regions">
                <table width="100%" border="4" cols="8% 27% 13% 13% 13% 13% 13%">
                        <tr bgcolor="lightgreen">
                                <th>Region Code</th>
                                <th>Region Name</th>
                                <th>Item Price</th>
                                <th>Items Sold</th>
```

```
                        <th>Revenues</th>
                        <th>Expenses</th>
                        <th>Profit</th>
                </tr>
                <xsl:for-each select="reg">
                        <xsl:apply-templates select="document(@filename)/region"/>
                </xsl:for-each>
                <tr>
                        <td colspan="3" align="right">Total:</td>
                        <td align="center">
                                <xsl:value-of select="$items-sold"/>
                        </td>
                        <td align="center">
                                <xsl:value-of select="$revenues"/>
                        </td>
                        <td align="center">
                                <xsl:value-of select="$expenses"/>
                        </td>
                        <td align="center">
                                <xsl:value-of select="$profit-after-tax"/>
                        </td>
                </tr>
        </table>
</xsl:template>

<xsl:template match="region">
<xsl:param name="fname"/>
        <tr>
                <td align="center">
                        <b><xsl:value-of select="region-code"/></b>
                </td>
                <td>
                        <xsl:value-of select="region-name"/>
                </td>
                <td align="center">
                        <xsl:value-of select="item-price"/>
                </td>
                <td align="center">
                        <xsl:value-of select="items-sold"/>
                </td>
                <td align="center">
                        <xsl:value-of select="revenues"/>
                </td>
                <td align="center">
                        <xsl:value-of select="expenses"/>
                </td>
                <td align="center">
                        <xsl:value-of select="profit * ( 1 - $tax )"/>
                </td>
        </tr>
</xsl:template>

</xsl:stylesheet>
```

6. Βιβλιογραφία

(1) Frank P. Coyle, "XML, Web Services and the Data Revolution", Addison – Wesley Information Technology Series, 2002

(2) Michael Fitzgerald, "Learning XSLT", O' Reilly, 2003

(3) Doug Tidwell, "XSLT", O' Reilly, 2001

(4) Sal Mangano, "XSLT Cookbook", O' Reilly, 2002

(5) Robert B. Mellor, "XML, Learning by Example", Franklin, Beedle & Associates, 2002

(6) www.exslt.org