Πανεπιστήμιο Κρήτης
Σχολή Θετικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

CREBITEL
Πολύγλωσσος Σημασιολογικός Χάρτης Πρόσβασης σε Βάσεις Γνώσεων Συντήρησης Έργων Τέχνης μέσω του Παγκόσμιου Ιστού

Βασίλειος Ε. Κοντογιάννης

Μεταπτυχιακή Εργασία

Ηράκλειο, Φεβρουάριος 2003

Πανεπιστήμιο Κρήτης
Σχολή Θετικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

CREBITEL
Πολύγλωσσος Σημασιολογικός Χάρτης Πρόσβασης σε Βάσεις Γνώσεων Συντήρησης Έργων Τέχνης μέσω του Παγκόσμιου Ιστού

Εργασία που υποβλήθηκε από τον
Βασίλειο Ε. Κοντογιάννη
ως μερική εκπλήρωση των απαιτήσεων για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Βασίλειος Ε. Κοντογιάννης
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Εισηγητική Επιτροπή:

Πάνος Κωνσταντόπουλος
Καθηγητής, Επόπτης

Αικατερίνη Χούστη
Καθηγήτρια, Μέλος

Δημήτρης Πλεξουσάκης
Αναπληρωτής Καθηγητής, Μέλος

Martin Doerr
Ερευνητής Β΄, Ι.Π. Ι.Τ.Ε. , Επιβλέπων

Δεκτή:

Πάνος Κωνσταντόπουλος, Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Φεβρουάριος 2003

Στους γονείς μου Μανόλη και Μαρία και
στ' αδέλφια μου Ανδρεανή και Ανδρέα

CREBITEL
Πολύγλωσσος Σημασιολογικός Χάρτης Πρόσβασης σε Βάσεις Γνώσεων Συντήρησης Έργων Τέχνης μέσω του Παγκόσμιου Ιστού

Βασίλειος Ε. Κοντογιάννης
Μεταπτυχιακή Εργασία
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Η συντήρηση και αποκατάσταση έργων τέχνης αποτελεί βασικό καθήκον των μουσείων σε ολόκληρο τον κόσμο. Η δραστηριότητα αυτή τα τελευταία χρόνια αξιοποιεί όλο και περισσότερο την πληροφορική τεχνολογία. Στην πρόσκτηση γνώσεων για το συγκεκριμένο πεδίο εφαρμογής πλέον χρησιμοποιούνται μέθοδοι εξέτασης και ανάλυσης βασισμένες σε τεχνικές ψηφιοποίησης, επεξεργασίας εικόνας, διαχείρισης και προστασίας δεδομένων. Παρόλα αυτά υπάρχει σημαντική έλλειψη πληροφοριακών συστημάτων διαχείρισης, οργάνωσης και παρουσίασης πληροφοριών και γνώσεων σχετικών με τη συντήρηση έργων τέχνης.

Το CREBITEL είναι μια πρώτη σχεδίαση και υλοποίηση ενός Πληροφοριακού Συστήματος Βασισμένου στον Παγκόσμιο Ιστό (ΠΣΒΠΙ) με εκπαιδευτικό προσανατολισμό σε θέματα συντήρησης. Η υλοποίηση του συστήματος ακολουθεί την αρχιτεκτονική 4-επιπέδων: $1^{ο}$ ΣΔΒΔ, $2^{ο}$ εξυπηρετητής εφαρμογής, $3^{ο}$ εξυπηρετητής Παγκόσμιου Ιστού, $4^{ο}$ φυλλομετρητής. Ως ΣΔΒΔ έχει επιλεγεί το Σύστημα Σημασιολογικού Ευρετηριασμού (ΣΣΕ) και η μοντελοποίηση βασίζεται στο μοντέλο δεδομένων της Telos. Το μοντέλο υποστηρίζει χαρακτηριστικά πολύγλωσσου θησαυρού. Παράλληλα σχετίζει τη συναφή ορολογία με υπαρκτά παραδείγματα-πίνακες ζωγραφικής, για τα οποία υπάρχουν καταχωρισμένα δελτία συντήρησης και πολυφασματικές εικόνες. Το CREBITEL προσπελάζεται από τον Παγκόσμιο Ιστό μέσω εφαρμογής Ιστού που εκτελείται στον εξυπηρετητή εφαρμογής Apache Tomcat. Η τελευταία είναι πλήρως παραμετρική κι αποτελείται από ένα σύνολο λειτουργιών υλοποιημένων με την τεχνολογία των Java Servlets. Οι λειτουργίες ανακτούν τα δεδομένα από το ΣΣΕ μέσω του Java API που παρέχεται και με κλήσεις native συναρτήσεων για την επικοινωνία με την πολυνηματική έκδοση του εξυπηρετητή του ΣΣΕ. Τα αποτελέσματα των λειτουργιών παράγονται αποκλειστικά σε μορφή XML, σύμφωνα με DTD που έχουμε προκαθορίσει, η δε παρουσίασή τους γίνεται εντελώς ανεξάρτητα, από αρχεία XSL. Η ανεξαρτησία

δεδομένων-παρουσίασης είναι η βάση για τα χαρακτηριστικά πολυγλωσσίας με άμεση εναλλαγή γλώσσας και πολλαπλών παρουσιάσεων που διακρίνουν το σύστημα.

Για την εισαγωγή δεδομένων στο σύστημα προτείνουμε μια σχεδίαση που μπορεί να υλοποιηθεί με τη χρήση "XML Entry Forms", ώστε η φόρτωση της συστήματος να γίνεται με σχετικά απλό τρόπο.

Το σύστημα δοκιμάστηκε με πλασματικά δεδομένα και οι επιδόσεις των λειτουργιών ανάκτησης κρίθηκαν ικανοποιητικές.

Επόπτης: Πάνος Κωνσταντόπουλος
Καθηγητής Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

CREBITEL
Multilingual Semantic Map for Web-Based Access to Art Conservation Knowledge Bases

Bill E. Kontogiannis
Master of Science Thesis
Computer Science Department
University of Crete

Abstract

Conservation and restoration of works-of-art is a major duty for museums all over the world. These processes become increasingly IT-based in the last few years. Knowledge acquisition in this field employs examination and analysis methods based on digitization, image processing and data management and security techniques. However, there is a lack of information systems for managing, organizing and presenting information and knowledge relevant to the conservation of works-of-art.

CREBITEL is an initial design and implementation of a Web-Based Information System (WBIS) with educational orientation concerning conservation. The system implementation follows a 4-tier architecture: 1st DBMS, 2nd application server, 3rd web server, 4th browser. The Semantic Index System (SIS) has been chosen as the DBMS, so modeling is based on the Telos semantic network data model. The model supports features of multilingual thesauri. It also relates the relevant terminology with specific paintings, for which there exist conservation reports and multispectral images. CREBITEL is accessed from the WWW through its Web application, which is executed on the Apache Tomcat application server. The Web application is fully parameterized and comprises a set of operations implemented with Java Servlet technology. The operations retrieve data from SIS through the available Java API, which uses native functions to communicate with the multithread version of the SISServer. The final results of the operations are exclusively formatted in XML according to predefined DTD. The presentation of XML results is independently accomplished using XSL files. The data-presentation independence is the basis for the features of multilinguality (direct language switch) and multipresentation, which differentiates this system from others.

Last but not least, a design for data entry is proposed for the present system. It may be implemented with the use of "XML Entry Forms", so that loading can be relatively easy.

The system was tested with virtual data and the performance of the several operations was judged as satisfactory.

Supervisor: Panos Constantopoulos
Professor of Computer Science
University of Crete

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω θερμά τον επόπτη καθηγητή μου κ. Πάνο Κωνσταντόπουλο για τις αρκετά εποικοδομητικές συζητήσεις που είχαμε στα πλαίσια της παρούσας εργασίας. Επίσης την κα. Αικατερίνη Χούστη και τον κ. Δημήτρη Πλεξουσάκη που δέχθηκαν πρόθυμα να είναι μέλη της εισηγητικής μου επιτροπής και να βοηθήσουν με τις παρατηρήσεις τους στη βελτίωση του συγγράμματος αυτού.

Ιδιαίτερα ευχαριστώ τον Martin Doerr, για την επίβλεψη της σχεδίασης και υλοποίησης του συστήματος που κατασκευάστηκε. Οι τακτικές συζητήσεις που είχαμε βοήθησαν να πάρω τις σωστές αποφάσεις σε καίρια σημεία. Οφείλω να τονίσω ιδιαίτερα τη «γερμανική» νοοτροπία της σκληρής δουλειάς και συνέπειας που τον διακρίνει! Ευχαριστώ ακόμα τους Χρήστο Γεωργή και Μαρία Χατζηδάκη με τους οποίους συνεργάστηκα εξαιρετικά στα πλαίσια της εργασίας αυτής.

Ένα μεγάλο ευχαριστώ ανήκει δικαιωματικά στην εταιρεία DATACRETA όπου εργαζόμουν καθόλη τη διάρκεια των μεταπτυχιακών μου σπουδών. Ευχαριστώ όλους τους συναδέρφους μου που έδειξαν μεγάλη κατανόηση στα «περίεργα» ωράριά μου κατά τις δύσκολες περιόδους και κατά καιρούς άκουγαν με προσοχή τα προβλήματά μου.

ΚΑΙ τώρα στην παρέα μου που απαρτίζεται από τους (αλφαβητικά):

Χαράλαμπο Αποστολίδη, Γιώργο Γεωργιλά, Σοφία Καρβουνιάρη, Αθηνά Κοντογιάννη, Κυριάκο Κρητικό & Ρένα Καγιαλή, Κώστα Λενακάκη, Αιμιλία Μαγκαναράκη, Γιάννη Μαυρικάκη, Ιάκωβο Μαυροειδή, Χαράλαμπο Παπαδάκη, Μαρίνα Παναγιωτάκη, Κώστα Παναγιωτάκη, Απόστολο Παπανικολάου, Γιώργο Σαπουντζή, Γιώργο Σημαντήρη, Μανόλη Τζομπανάκη, Γιάννη Χαριτάκη & Βαγγελιώ Κούκου,

απευθύνω το πιο μεγάλο ΕΥΧΑΡΙΣΤΩ. Και μια ευχή: μετά από 2-3 χρόνια μακάρι να συναντιόμαστε όπως και τώρα και να περνάμε καλά όπως μόνο εμείς ξέρουμε!

Μπαμπά, Μαμά, Αντριάνη, Αντρέα, σε σας αφιερώνω αυτή την εργασία, στην οικογένειά μου...

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Στις μέρες μας ακατάπαυστο εμφανίζεται το φαινόμενο της ανάπτυξης κάθε λογής πληροφοριακών συστημάτων, που καλύπτουν τις ανάγκες πολυάριθμων πεδίων εφαρμογής. Με την πάροδο του χρόνου ολοένα αυξανόμενο ενδιαφέρον παρουσιάζουν τα πληροφοριακά συστήματα εκείνα που θα καλύψουν, θα υποστηρίξουν και θα δώσουν ώθηση στους πιο εξειδικευμένους από τους τομείς της ανθρώπινης δραστηριότητας. Η ανάπτυξη τέτοιων συστημάτων ωστόσο δεν είναι μια τετριμμένη υπόθεση. Απαιτεί τη χρησιμοποίηση και ολοκλήρωση διαφόρων αρχιτεκτονικών, τεχνολογιών, τεχνικών, εργαλείων, ικανοποιώντας ταυτόχρονα τις απαιτήσεις σε απλότητα (simplicity), στοιχειομέρεια (modularity), κλιμάκωση (scalability), αποτελεσματικότητα (effectiveness) και αποδοτικότητα (efficiency).

Πριν ξεκινήσει η ανάπτυξη ενός εξειδικευμένου πληροφοριακού συστήματος, πρέπει πρώτα από όλα να γίνει λεπτομερής καταγραφή

- των δεδομένων που αυτό θα περιέχει
- των δεδομένων του πεδίου που ήδη υπάρχουν σε μη ηλεκτρονική ή αδόμητη ηλεκτρονική μορφή
- των υπηρεσιών και πληροφοριών που θα παρέχει
- των χρηστών που θα το χρησιμοποιούν και πού αυτοί θα βρίσκονται

Η συνέχεια απαιτεί μια γενική θεώρηση της δομής που έχουν τα υπάρχοντα δεδομένα και των συσχετίσεων μεταξύ τους. Αυτό είναι απαραίτητο για να γίνει μια αρχική εκτίμηση του βαθμού δυσκολίας αναπαράστασής τους στα διαφόρων τύπων μοντέλα δεδομένων (σχεσιακό, δικτυωτό ιεραρχικό).

Το επόμενο βήμα είναι η έρευνα αγοράς για εργαλεία που θα αποτελέσουν και βασικές συνιστώσες του συστήματος, τα οποία θα αναλάβουν περίπλοκες λειτουργίες του με τυποποιημένο τρόπο, όπως η αποθήκευση και ανάκτηση δεδομένων, η εξυπηρέτηση αιτήσεων χρηστών, η εμφάνιση των πληροφοριών στους χρήστες κλπ. Στο στάδιο αυτό πρέπει να μελετηθούν προσεκτικά οι δυνατότητες διασύνδεσης (interfacing) των εργαλείων αυτών μεταξύ τους ή και με άλλα για την περίπτωση μελλοντικών επεκτάσεων. Η έρευνα αγοράς είναι επιβεβλημένη καθώς μπορεί να απαλλάξει από αμέτρητες ανθρωποώρες κατά τη φάση της ανάπτυξης του συστήματος. Επίσης είναι απόλυτα σίγουρο ότι τα εργαλεία που κυκλοφορούν στην αγορά έχουν ελεγχθεί σχολαστικά και έχουν προσεχθεί πολύ περισσότερο απ’ όσο θα μπορούσαν να προσεχθούν αν κατασκευάζονταν ταυτόχρονα με την ανάπτυξη του πληροφοριακού συστήματος.

Ακολουθεί η φάση της λεπτομερούς μοντελοποίησης του πεδίου εφαρμογής με την διάκριση όλων των πιθανών οντοτήτων, συσχετίσεων και περιορισμών ακεραιότητας, προσαρμογή των στο επιλεγμένο μοντέλο δεδομένων και φόρτωση του τελικού σημασιολογικού μοντέλου, με τη χρήση των μηχανισμών του εργαλείου που έχει επιλεχθεί για το σκοπό αυτό.

Το τελευταίο βασικό στάδιο στη φάση της ανάπτυξης του πληροφοριακού συστήματος είναι η επιλογή προγραμματιστικής πλατφόρμας και τεχνικών για την κατασκευή της «καρδιάς» του συστήματος. Αυτή θα αποτελείται από το σύνολο των προγραμμάτων με τα οποία θα επιτυγχάνεται η διακίνηση της πληροφορίας (μεταξύ συστήματος και χρηστών) και θα επιτελούνται οι απαιτούμενες λειτουργίες. Τα προγραμματιστικά εργαλεία και οι τεχνικές που θα χρησιμοποιηθούν θα πρέπει να υποστηρίζονται από τα ήδη επιλεχθέντα εργαλεία-συνιστώσες του συστήματος. Το τελευταίο σημείο που πρέπει να προσεχθεί πριν τη φάση της υλοποίησης είναι σε ποια μορφή και σε ποιες μονάδες θα γίνεται η διακίνηση της πληροφορίας και κατά πόσο αυτή θα είναι αποδεσμευμένη από τη μορφή με την οποία θα παρουσιάζεται.

1.1 Κατηγοριοποιήσεις Πληροφοριακών Συστημάτων

Από τα μέσα περίπου της δεκαετίας του ’50 ξεκίνησε η ανάπτυξη πληροφοριακών συστημάτων. Μισό αιώνα αργότερα τα πληροφοριακά συστήματα που χρησιμοποιούνται σε διάφορες δραστηριότητες είναι τόσο πολλά ώστε μπορούν να κατηγοριοποιηθούν με διάφορους τρόπους.

Έτσι λοιπόν τα πληροφοριακά συστήματα μπορούν σε ένα αφηρημένο επίπεδο να κατηγοριοποιηθούν [1] βάσει:

- Οργανωτικής Δομής
- Λειτουργιών
- Παρεχόμενης Υποστήριξης (συνηθέστερη κατηγοριοποίηση)
- Αρχιτεκτονικής
- Υποστηριζόμενης Δραστηριότητας

Ο πίνακας 1.1 δείχνει της κατηγορίες συστημάτων που κατατάσσονται σε κάθε μια από τις παραπάνω κατηγοριοποιήσεις.

Η έκρηξη τα τελευταία χρόνια της δικτυακής τεχνολογίας, η καθιέρωση πρωτοκόλλων επικοινωνίας, η εξάπλωση του Παγκόσμιου Ιστού (ΠΙ), η εμφάνιση ανεξάρτητων-πλατφόρμας γλωσσών και τεχνολογιών προγραμματισμού, έχουν αναδείξει πλέον και μια νέα ειδική κατηγορία, τα Πληροφοριακά Συστήματα Βασισμένα στον Παγκόσμιο Ιστό - ΠΣΒΠΙ (Web-Based Information Systems, WBIS).

Κατηγοριοποίηση βάσει	**Κατηγορίες**
Οργανωτικής Δομής (Organizational Structure)	Κλαδικά (Departmental)
	Επιχειρησιακά (Enterprise)
	Διαεπιχειρησιακά (Interorganizational)
Λειτουργιών	Λογιστικά (Accounting)
	Εμπορικά (Marketing)
	Οικονομικά (Finance)
	Παραγωγής (Production)
	Διαχείρισης Ανθρωπίνων Πόρων (Human Resources Management)
Παρεχόμενης Υποστήριξης	Επεξεργασίας Δοσοληψιών (Transaction Processing, TPS, Early 1950s)
	Διαχείρισης (Management, MIS, 1960s)
	Διαχείρισης Γνώσης (Knowledge Management, KMS) & Έμπειρα Συστήματα (Expert Systems, Mid 1980s)
	Αυτοματισμού Γραφείου (Office Automation, OAS, Late 1960s)
	Υποστήριξης Αποφάσεων (Decision Support, DSS, Early 1970s)
	Υποστήριξης Διοίκησης (Executive Support, ESS, Early 1980s)
	Τεκμηρίωσης (Documentation, DIS)
Αρχιτεκτονικής	Βασισμένα σε mainframe (Mainframe-based)
	Αυτόνομου προσωπικού υπολογιστή (Standalone PC)
	Κατανεμημένα ή Δικτυακά υπολογιστικά συστήματα (Distributed or Networked Computing Systems)
Υποστηριζόμενης Δραστηριότητας	Λειτουργικά – Καθημερινών Λειτουργιών Επιχείρησης (Operational)
	Διαχειριστικά – Βραχυπρόθεσμου σχεδιασμού, ελέγχου και οργάνωσης (Managerial)
	Στρατηγικά – Μακροπρόθεσμου σχεδιασμού και στρατηγικής (Strategic)

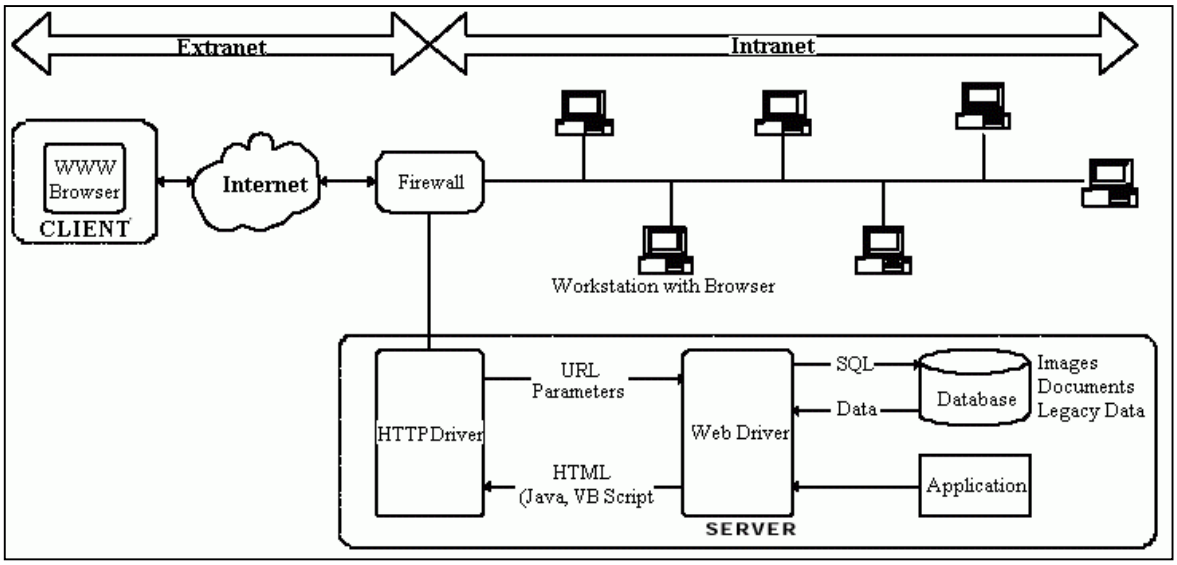
Πίνακας 1.1 - Κατηγοριοποιήσεις και κατηγορίες Πληροφοριακών Συστημάτων

1.2 Πληροφοριακά Συστήματα βασισμένα στον Παγκόσμιο Ιστό

1.2.1 Ορισμός

Τα ΠΣΒΠΙ είναι πληροφοριακά συστήματα που επιτρέπουν τη διαχείριση δεδομένων και την εκτέλεση λειτουργιών μέσω του Διαδικτύου. Διαφέρουν από τους απλούς και στατικούς δικτυακούς τόπους στο ότι παράγουν και διακινούν συνεχώς δυναμικά δεδομένα που περιέχονται σε σελίδες, τις οποίες βλέπουν οι τελικοί χρήστες με τη χρήση φυλλομετρητών του ΠΙ (web-browsers). Το βασικό τους χαρακτηριστικό είναι ότι ολοκληρώνουν δικτυακές εφαρμογές με Συστήματα Διαχείρισης Βάσεων Δεδομένων – ΣΔΒΔ (DBMS), και επιτρέπουν περίπλοκες δραστηριότητες που βασίζονται στα τελευταία, όπως π.χ. το ηλεκτρονικό-επιχειρείν (e-business) [2]. Ο σχεδιασμός, η υλοποίηση και η συντήρηση των ΠΣΒΠΙ είναι φυσικό να βασίζονται σε τεχνικές συστημάτων βάσεων δεδομένων, συστήματα ροής εργασιών (workflow systems) και τεχνολογίες Διαδικτύου.

Αν θέλαμε να ταξινομήσουμε τα ΠΣΒΠΙ κάπου μέσα στον πίνακα 1.1, θα τα κατατάσσαμε βάσει της αρχιτεκτονικής τους στα Κατανεμημένα ή Δικτυακά υπολογιστικά συστήματα. Η γενική δομή ενός ΠΣΒΠΙ ακολουθεί τη δομή των περισσοτέρων "Intranet & Extranet" [3] εφαρμογών η οποία φαίνεται στο σχήμα 1.1.



Σχήμα 1.1 - Δομή Εφαρμογής "Intranet&Extranet "

Ο ΠΙ είναι βασισμένος στην αρχιτεκτονική πελάτη-εξυπηρετητή (client-server architecture). Παρουσιάζει αρκετό ενδιαφέρον πάντως να δει κανείς μια επισκόπηση για το πώς φτάσαμε στην αρχιτεκτονική αυτή και ποιες είναι οι μορφές της που χρησιμοποιούνται σήμερα.

1.2.2 Πριν την αρχιτεκτονική πελάτη-εξυπηρετητή

Πριν την αρχιτεκτονική πελάτη-εξυπηρετητή, που εμφανίστηκε στα μέσα της δεκαετίας του 1980, υπήρχαν 2 βασικές αρχιτεκτονικές, η Αρχιτεκτονική Mainframe και η Αρχιτεκτονική Διαμοιραζόμενων Αρχείων (File Sharing Architecture) [4].

Στην **Αρχιτεκτονική Mainframe** όλη η ισχύς και η «εξυπνάδα» βρίσκεται σε ένα κεντρικό υπολογιστή (host computer). Οι χρήστες αλληλεπιδρούν με τον κεντρικό υπολογιστή συνήθως με τη χρήση «χαζών τερματικών» (dummy terminals), τα οποία «συλλαμβάνουν» πατήματα πλήκτρων και αποστέλλουν την πληροφορία στον κεντρικό υπολογιστή. Η συγκεκριμένη αρχιτεκτονική δεν είναι δεσμευτική όσον αφορά την πλατφόρμα υλικού, π.χ. μπορεί να αποτελείται από Linux-PC servers, Windows-PC workstations και ειδικά προγράμματα που λέγονται εξομοιωτές τερματικών (terminal emulators). Οι περιορισμοί της αρχιτεκτονικής αυτής εντοπίζονται στην αδυναμία υποστήριξης σύγχρονων Γραφικών Διεπαφών Χρήστη (GUI) και στην αδυναμία πολλαπλών συνδέσεων σε κατανεμημένες βάσεις δεδομένων.

Στην **Αρχιτεκτονική Διαμοιραζόμενων Αρχείων** βασίζονταν τα πρώτα δίκτυα υπολογιστών, όπου αρχεία (τόσο προγράμματα όσο και δεδομένα) φορτώνονταν εξ' ολοκλήρου από τη διαμοιραζόμενη τοποθεσία αρχείων στο περιβάλλον εργασίας (desktop environment) του χρήστη σαν να βρίσκονταν στον τοπικό δίσκο του σταθμού εργασίας του. Στη συνέχεια η αιτούμενη εργασία του χρήστη εκτελούνταν τοπικά. Η συγκεκριμένη αρχιτεκτονική μπορεί να λειτουργήσει καλά μόνο αν η κοινή χρήση πόρων είναι σε χαμηλά επίπεδα, ο ρυθμός ανανέωσης των διαμοιραζόμενων αρχείων χαμηλός και ο όγκος δεδομένων προς μεταφορά από τον εξυπηρετητή αρχείων μικρός. Ο σημαντικότερος περιορισμός είναι η συμφόρηση στο τοπικό δίκτυο (network congestion). Για το λόγο αυτό με την αύξηση του αριθμού των "online" χρηστών και την εμφάνιση των GUI, έπαψε η εν λόγω αρχιτεκτονική να χρησιμοποιείται από τις αρχές της δεκαετίας του 1990 οπότε κυριάρχησε πλέον η αρχιτεκτονική πελάτη-εξυπηρετητή.

1.2.3 Ενδιάμεσο Λογισμικό

Καθιερωμένος ορισμός για το τι είναι Ενδιάμεσο Λογισμικό – ΕΛ – δεν έχει διατυπωθεί, αλλά έχουν κατά καιρούς εκφρασθεί διάφοροι που προσεγγίζουν τη σημασία της λειτουργίας του. Δυο απ' αυτούς είναι οι παρακάτω:

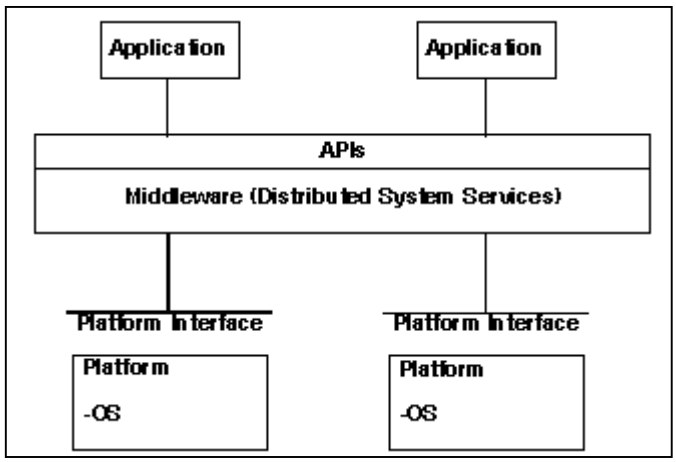
«Το ενδιάμεσο λογισμικό είναι το λογισμικό που αποτελείται από ένα σύνολο υπηρεσιών, οι οποίες εξασφαλίζουν τη συνδεσιμότητα και επιτρέπουν την αλληλεπίδραση μεταξύ πολλαπλών διεργασιών εκτελούμενων σε ένα ή περισσότερους διασυνδεδεμένους σε δίκτυο υπολογιστές» [4].

«Ενδιάμεσο Λογισμικό είναι ένα επίπεδο λογισμικού που βρίσκεται μεταξύ του επιπέδου της εφαρμογής και του επιπέδου δικτύωσης των ετερογενών πρωτοκόλλων και

περιβαλλόντων εργασίας. Διαχωρίζει τις εφαρμογές από οποιεσδήποτε εξαρτήσεις από ετερογενή λειτουργικά συστήματα, πλατφόρμες υλικού και πρωτόκολλα επικοινωνίας [5].

Το ΕΛ είναι απαραίτητο για την τροποποίηση των mainframe εφαρμογών σε εφαρμογές πελάτη-εξυπηρετητή και για την εξασφάλιση της επικοινωνίας μεταξύ ετερογενών περιβαλλόντων. Γνωστό Ενδιάμεσο Λογισμικό αποτελούν τα Distributed Computing Environment (OSF's DCE), Common Object Request Broker Architecture (OMG's CORBA) και Component Object Model (Microsoft's COM/DCOM).

Οι υπηρεσίες του ΕΛ είναι σύνολα κατανεμημένου λογισμικού που υπάρχουν μεταξύ της εφαρμογής και του λειτουργικού συστήματος και των δικτυακών υπηρεσιών σε ένα υπολογιστή-κόμβο δικτύου (Σχήμα 1.2) [6].



Σχήμα 1.2 - Ενδιάμεσο Λογισμικό

Παρέχουν στις εφαρμογές ένα πιο λειτουργικό σύνολο Διεπαφών Προγραμματισμού Εφαρμογών (Application Programming Interfaces, API) από ότι το λειτουργικό σύστημα και οι δικτυακές υπηρεσίες, ώστε να :

- Εντοπίζουν με ξεκάθαρο τρόπο και να αλληλεπιδρούν με άλλες εφαρμογές ή υπηρεσίες πάνω από το δίκτυο
- Είναι ανεξάρτητες από δικτυακές υπηρεσίες
- Είναι αξιόπιστες και διαθέσιμες
- Είναι επεκτάσιμες χωρίς να χάνουν τη λειτουργικότητα τους [7]

Το ΕΛ μπορεί να χρησιμοποιηθεί ως:

- **Επιτηρητής δοσοληψιών (Transaction Processing Monitor, TP),** παρέχοντας εργαλεία και ένα περιβάλλον ανάπτυξης και διάθεσης κατανεμημένων εφαρμογών.
- Μέσο για την **Απομακρυσμένη Κλήση Διαδικασιών (Remote Procedure Call)**, επιτυγχάνοντας την κατανομή της λογικής μιας εφαρμογής στο δίκτυο. Η λογική των προγραμμάτων στους απομακρυσμένους υπολογιστές μπορεί να

εκτελεσθεί τόσο εύκολα όσο απλή είναι η κλήση διαδικασιών στον τοπικό υπολογιστή.

- Μέσο για την **Ανταλλαγή Μηνυμάτων (Message-Oriented Middleware, MOM),** παρέχοντας ασύγχρονη ανταλλαγή δεδομένων μεταξύ προγραμμάτων μιας κατανεμημένης εφαρμογής.
- **Κατανεμητής Αιτήσεων για Αντικείμενα (Object Request Broker),** επιτρέποντας στα αντικείμενα μιας εφαρμογής να είναι κατανεμημένα και διαμοιραζόμενα πάνω σε ετερογενή δίκτυα.
- **ενδιάμεσο λογισμικό για Βάσεις Δεδομένων – ΕΛΒΔ – (Database Middleware),** επιτρέποντας σε εφαρμογές να χειρίζονται συνδέσεις με τις Βάσεις Δεδομένων, καθώς και να ανακτούν και να αποθηκεύουν στοιχεία σ' αυτές.. Το ΕΛΒΔ μπορεί να στηρίζεται σε καθιερωμένα πρότυπα όπως είναι το ODBC, JDBC για τη σύνδεση με σχεσιακές βάσεις δεδομένων, αλλά μπορεί να είναι και ειδικά σχεδιασμένη Διεπαφή Προγραμματισμού Εφαρμογής (API) για συγκεκριμένη βάση δεδομένων.
- **ενδιάμεσο λογισμικό με Εξυπηρετητή Εφαρμογών (Application Server Middleware),** επιτρέποντας αιτήσεις που στέλνονται μέσω φυλλομετρητών του ΠΙ να φτάσουν σε εφαρμογές προορισμένες για το Διαδίκτυο. Οι αιτήσεις αυτές γίνονται σύμφωνα με καθιερωμένα πρωτόκολλα και APIs, μεταφράζονται από τον εξυπηρετητή εφαρμογής σε εκτέλεση λειτουργιών της δικτυακής εφαρμογής και τα αποτελέσματα επιστρέφονται στον πελάτη-φυλλομετρητή βάσει πάντα καθιερωμένων πρωτοκόλλων.

1.2.4 Ν-επίπεδες αρχιτεκτονικές πελάτη-εξυπηρετητή (Ν≥2)

Οι περιορισμοί της αρχιτεκτονικής διαμοιραζόμενων αρχείων οδήγησε στην αρχιτεκτονική πελάτη-εξυπηρετητή. Σε αυτήν την αρχιτεκτονική ο εξυπηρετητής αρχείων αντικαθίσταται από ένα ΣΔΒΔ, το οποίο αναλαμβάνει να απαντάει σε συγκεκριμένες επερωτήσεις των πελατών (clients) κι έτσι αποφεύγεται το φαινόμενο της καθολικής μεταφοράς αρχείων και της συνεπαγόμενης. συμφόρησης στο δίκτυο. Οι επερωτήσεις των πελατών γίνονται είτε με τη χρήση καθιερωμένων γλωσσών άμεσα ή έμμεσα (βλ. ODBC), είτε με τη χρήση ειδικών APIs και πιθανόν RPC.

2-επίπεδη αρχιτεκτονική πελάτη-εξυπηρετητή (2-tier client-server architecture): Στην αρχιτεκτονική αυτή στον πελάτη εγκαθίσταται ειδικό λογισμικό συνήθως περιβαλλόμενο από μια γραφική διεπαφή, το οποίο του παρέχει τη δυνατότητα εκτέλεσης διαφόρων λειτουργιών ανάκτησης και αποθήκευσης δεδομένων στης βάση. Η εκτέλεση των λειτουργιών που ζητούν οι χρήστες γίνεται μερικώς στον εξυπηρετητή από ειδικές υπηρεσίες διαχείρισης της βάσης και μερικώς στον πελάτη (κυρίως επεξεργασία των ήδη ανακτηθέντων δεδομένων).

Οι περιορισμοί που εμφανίζονται σε αυτήν την περίπτωση έχουν να κάνουν με τον αριθμό των πελατών που έχουν ταυτόχρονη πρόσβαση στη βάση αλλά και με τη δυνατότητα αναβάθμισης του εξυπηρετητή. Έτσι σαν ανώτατο όριο μπορούν να τεθούν οι 100 χρήστες, διότι με περισσότερους η απόδοση πέφτει δραματικά εξαιτίας των λεγόμενων "keep-alive" μηνυμάτων που ανταλλάσσονται ακόμα και όταν οι πελάτες αδρανούν. Επίσης πιθανές αναβαθμίσεις στο ΣΔΒΔ (ή ενδεχόμενη αλλαγή του), είναι πολύ δύσκολες αφού πιθανότατα θα απαιτήσουν και αντίστοιχες αναβαθμίσεις στους πελάτες. Για τον λόγο αυτό και η 2-επίπεδη αρχιτεκτονική πελάτη-εξυπηρετητή είναι απαγορευτική να εφαρμοσθεί στο Διαδίκτυο.

3-επίπεδη αρχιτεκτονική πελάτη-εξυπηρετητή (3-tier client-server architecture): Στην αρχιτεκτονική αυτή ένα ενδιάμεσο επίπεδο λογισμικού προστίθεται ανάμεσα στο περιβάλλον εργασίας του πελάτη και στο περιβάλλον διαχείρισης της βάσης. Υπάρχουν διάφοροι τρόποι υλοποίησης του ενδιάμεσου αυτού λογισμικού όπως ως Επιτηρητής Δοσοληψιών, Εξυπηρετητής Μηνυμάτων, Εξυπηρετητής Εφαρμογής κλπ [8] . Η αρχιτεκτονική αυτή χρησιμοποιείται όταν απαιτείται μια αποτελεσματική σχεδίαση πελάτη-εξυπηρετητή που να παρέχει αυξημένη απόδοση (performance), ευελιξία (flexibility), συντηρησιμότητα (maintainability), επαναχρησιμοποιησιμότητα (reusability) και κλιμάκωση (scalability), κρύβοντας ταυτόχρονα την πολυπλοκότητα της κατανεμημένης επεξεργασίας από τον χρήστη ([8], [9]).

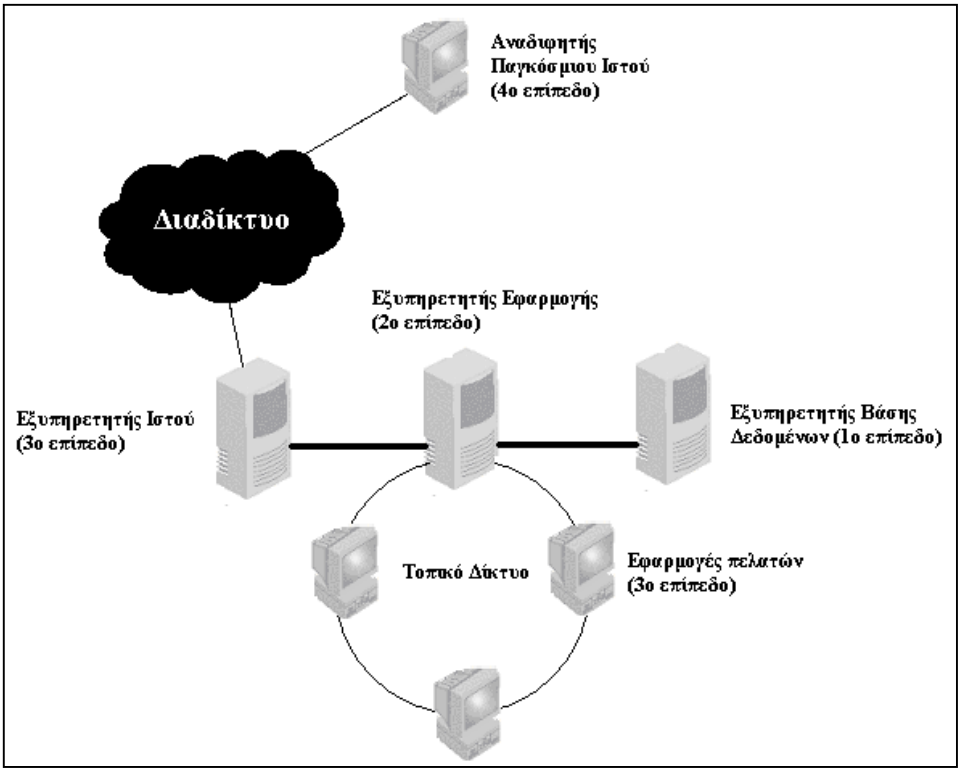
Στο 1^{ο} επίπεδο παρέχεται η λειτουργικότητα διαχείρισης της βάσης δεδομένων. Το τμήμα διαχείρισης της βάσης εγγυάται ότι τα δεδομένα είναι συνεπή στο κατανεμημένο περιβάλλον με τη χρήση χαρακτηριστικών όπως το κλείδωμα δεδομένων (data locking), ο έλεγχος συνέπειας (consistency checking) και η δημιουργία αντιγράφων (replication).

Στο 2^{ο} (μεσαίο) επίπεδο ο εξυπηρετητής εφαρμογής επιτυγχάνει τα χαρακτηριστικά που προαναφέρθηκαν (performance, flexibility etc.) με την κεντρικοποίηση της λογικής επεξεργασίας (process logic or business logic). Η κεντρικοποίηση της λογικής επεξεργασίας κάνει τη διαχείριση και τις αλλαγές ευκολότερες με τον τοπικό περιορισμό της λειτουργικότητας. Με τον τρόπο αυτό οι αλλαγές που χρειάζονται γίνονται μία φορά, τοποθετούνται στο μεσαίο επίπεδο και γίνονται άμεσα διαθέσιμες σε όλους του πελάτες του κατανεμημένου συστήματος.

Στο 3^{ο} επίπεδο υπάρχει ο πελάτης, από τον οποίο έχει αποσπασθεί ο περισσότερος από τον κώδικα επεξεργασίας, που μεταφέρθηκε στο 2^{ο} επίπεδο. Στον πελάτη «τρέχουν» εφαρμογές που παρέχουν απλά γραφικές διεπαφές (μενού, πλαίσια διαλόγου κλπ) με τα οποία ο χρήστης επικοινωνεί με το σύστημα. Τέτοιες εφαρμογές είναι οι φυλλομετρητές του ΠΙ.

Ν-επίπεδες αρχιτεκτονικές (N-tier architectures): Μερικές φορές το μεσαίο επίπεδο στις 3-επίπεδες αρχιτεκτονικές χωρίζεται σε 2 ή περισσότερα επίπεδα με διαφορετικές λειτουργίες και υπηρεσίες. Κλασσικό παράδειγμα αποτελούν οι εφαρμογές που διαχωρίζουν τη λογική επεξεργασίας από τη λογική παρουσίασης με τη χρήση

διαφόρων τεχνολογιών και τεχνικών. Αυτές οι εφαρμογές λέγεται ότι ακολουθούν πολυ-επίπεδες ή N-επίπεδες αρχιτεκτονικές πελάτη-εξυπηρετητή (σχήμα 1.3).



Σχήμα 1.3 - Αρχιτεκτονική 4-επιπέδων

Αρκετές εφαρμογές προσανατολισμένες για το Διαδίκτυο προσπαθούν να χρησιμοποιήσουν αυτές τις αρχιτεκτονικές και ο λόγος είναι η δυνατότητα επεκτασιμότητας (scalability) και μεταφερσιμότητας (portability) των.

1.3 Αντικείμενο Εργασίας

Σκοπός της παρούσας εργασίας είναι ο σχεδιασμός και η μερική υλοποίηση ενός ΠΣΒΠΙ, το οποίο θα αποτελεί βοήθημα και μέσο εκπαίδευσης για τους συντηρητές έργων τέχνης. Ενός εργαλείου που θα τους βοηθάει α) στην προσπάθεια διάγνωσης αλλοιώσεων τις οποίες τα έργα τέχνης έχουν υποστεί, β) στον εντοπισμό των αιτίων που προκάλεσαν τις αλλοιώσεις, γ) στο συσχετισμό των αλλοιώσεων με τις τεχνικές και τα υλικά κατασκευής των έργων και δ) στην λήψη αποφάσεων για πιθανούς τρόπους συντήρησης και αποκατάστασης. Όλα τα παραπάνω θα γίνονται με τη χρήση υπαρκτών παραδειγμάτων – αρχικά πινάκων ζωγραφικής – σημασιολογικά συνδεδεμένων με διάφορους τρόπους με θησαυρό όρων του συγκεκριμένου πεδίου εφαρμογής.

Η επιλογή των συνιστωσών-εργαλείων καθώς και η ανάπτυξη της δικτυακής εφαρμογής[1] πρέπει να γίνει με προσοχή ώστε το σύστημα να είναι:

- **Σταθερό**
  Ο εξυπηρετητής εφαρμογής και ο εξυπηρετητής της βάσης θα πρέπει να μην παρουσιάζουν διακοπές στη λειτουργία τους που να οφείλονται σε προγραμματιστικά λάθη από την πλευρά της εφαρμογής.
- **Αξιόπιστο**
  Τα αποτελέσματα των λειτουργιών ανάκτησης δεδομένων θα πρέπει να συμφωνούν με τα κριτήρια που τίθενται από τους χρήστες. Επίσης οι λειτουργίες καταχώρησης/τροποποίησης δεδομένων θα πρέπει να αφήνουν πάντα το σύστημα σε συνεπή κατάσταση βάσει των περιορισμών ακεραιότητας που έχουν τεθεί.
- **Γρήγορο/Αποδοτικό**
  Ο χρόνος εκτέλεσης κάθε λειτουργίας θα πρέπει να είναι ανάλογος της πολυπλοκότητάς της βάσει του σχεδιασμένου μοντέλου δεδομένων. Η καθυστέρηση λήψης των αποτελεσμάτων οποιασδήποτε λειτουργίας θα πρέπει να μην υπερβαίνει κατά πολύ την καθυστέρηση μεταφοράς των στο χρήστη μέσω του Διαδικτύου.
- **Ευέλικτο στην παρουσίαση**
  Θα πρέπει να υπάρχει η υποδομή για εύκολη τροποποίηση της όψης του δικτυακού τόπου χωρίς επέμβαση στο προγραμματιστικό κομμάτι της εφαρμογής.
- **Επεκτάσιμο**
  Θα πρέπει να μπορεί να γίνεται εύκολα προσθήκη/αφαίρεση λειτουργιών από το σύστημα χωρίς να επηρεάζεται η ήδη υπάρχουσα εφαρμογή.
- **Παραμετροποιήσιμο**
  Θα πρέπει να υπάρχει ένα σύνολο παραμέτρων εύκολα προσβάσιμων και τροποποιήσιμων από τους διαχειριστές του συστήματος. Οι παράμετροι αυτοί θα καθορίζουν στοιχεία όπως δικτυακές διευθύνσεις (IP addresses), πόρτες (ports), ονομασίες φακέλων δικτυακής εφαρμογής (web-application directories), πλήθος αποτελεσμάτων ανά σελίδα, κωδικοί και ονομασίες γλωσσών κ.α.

Με βάση τις παραπάνω αρχές σχεδιάστηκε και αναπτύχθηκε μερικώς ένα ΠΣΒΠΙ που μπορεί να θεωρηθεί ότι προσεγγίζει τα απλά χαρακτηριστικά των συστημάτων Διαχείρισης Θησαυρών – ΣΔΘ (TMS), Τεκμηρίωσης (DIS), Αναπαράστασης και Διαχείρισης Γνώσης (KMS) και Στήριξης Αποφάσεων (DSS) όπως αυτά

[1] Στο εξής όπου χρησιμοποιείται ο όρος δικτυακή εφαρμογή για το παρόν σύστημα θα εννοείται εφαρμογή Παγκόσμιου Ιστού (web-application)

κατηγοριοποιούνται στον πίνακα 1.1. Το σύστημα βασίστηκε στην αρχιτεκτονική τεσσάρων επιπέδων (4-tier) και έχει τα εξής βασικά χαρακτηριστικά:

- Διαχείριση θησαυρού όρων
- Γραφική αναπαράσταση ιεραρχιών όρων
- Αποδέσμευση δεδομένων-παρουσίασης
- Πολυγλωσσία και υποστήριξη άμεσης εναλλαγής γλώσσας
- Άμεση εναλλαγή του στιλ παρουσίασης
- Καθοδηγητική πλοήγηση ως το σημείο ενδιαφέροντος του χρήστη
- Υποστήριξη πολλών χρηστών
- Επεκτασιμότητα λειτουργιών
- Αρχείο παραμέτρων για εύκολες τροποποιήσεις/προσθήκες
- Ενσωμάτωση και επικοινωνία εφαρμογής με δίκτυο στατικών ιστοσελίδων

Τα βήματα που ακολουθήθηκαν ήταν με χρονολογική σειρά :

1. Επιλογή ΣΔΒΔ
2. Επιλογή εργαλείων ανάπτυξης
3. Επιλογή αρχιτεκτονικής για τη δικτυακή εφαρμογή
4. Μοντελοποίηση απλοποιημένου πολύγλωσσου θησαυρού
5. Μοντελοποίηση πεδίου εφαρμογής και σύνδεση του μοντέλου με το θησαυρό
6. Διακριτοποίηση κύριων παραμέτρων συστήματος
7. Απόφαση για τη δομή φακέλων της εφαρμογής
8. Καθορισμός των φορμών (templates) εμφάνισης δεδομένων και των Τύπων Εγγράφων (DTD)
9. Ανάπτυξη εφαρμογής (προγραμματιστικό μέρος)
10. Αυτόνομος προγραμματισμός παρουσίασης XML δεδομένων με χρήση XSL
11. Τροποποίηση για υποστήριξη πολλών χρηστών
12. Τροποποίηση εφαρμογής για υποστήριξη πολλαπλών άμεσα εναλλασσόμενων παρουσιάσεων

1.4 Υλοποίηση και εφαρμογή

Για την μοντελοποίηση του πεδίου εφαρμογής χρησιμοποιήθηκε η γλώσσα αναπαράστασης γνώσης Telos [10] και ως ΣΔΒΔ το Σύστημα Σημασιολογικού Ευρετηριασμού – ΣΣΕ (Semantic Index System, SIS) [11].

Για την υλοποίηση της δικτυακής εφαρμογής χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java και συγκεκριμένα η τεχνολογία των Servlets. Η πλατφόρμα ανάπτυξης ήταν τα Microsoft Windows 2000, ωστόσο η εφαρμογή μπορεί να «τρέξει» σε

κάθε πλατφόρμα με χρήση της Εικονικής Μηχανής της Java (Java Virtual Machine) σε έκδοση 1.3.1 και νεότερη.

Για την εμφάνιση ιεραρχιών όρων με γραφικό τρόπο χρησιμοποιήθηκε η γλώσσα περιγραφής διδιάστατων διανυσματικών γραφικών SVG (Scalable Vector Graphics), γλώσσα παράγωγη από την XML.

Ως ενδιάμεσο λογισμικό χρησιμοποιήθηκε το προϊόν Apache Tomcat 4.0 [12] το οποίο αναλαμβάνει το ρόλο του 2ου και 3ου επιπέδου στην αρχιτεκτονική τεσσάρων επιπέδων που χρησιμοποιήσαμε. Αυτό συμβαίνει μιας και ολοκληρώνει ένα εξυπηρετητή εφαρμογής (application server) και ένα εξυπηρετητή του ΠΙ (web-server).

Οι αιτήσεις των απλών χρηστών φτάνουν μέχρι το μεσαίο επίπεδο μέσω του πρωτοκόλλου μεταφοράς υπερκειμένου HTTP (HyperText Transfer Protocol). Το πρωτόκολλο αυτό είναι ένα πρωτόκολλο επιπέδου εφαρμογών (Application Layer), από τα ανώτερα στρώματα του πρωτοκόλλου ελέγχου μετάδοσης TCP/IP (Transmission Control Protocol/Internet Protocol). Η επικοινωνία μεταξύ του εξυπηρετητή εφαρμογής και των εξυπηρετητή ΠΙ και εξυπηρετητή βάσης δεδομένων βασίζεται στο TCP/IP.

Η συγγραφή των DTD για τις διάφορες φόρμες εμφάνισης δεδομένων (templates) χρησιμοποιήθηκε το προϊόν XML Authority 2.2 της εταιρείας TIBCO Software Inc.

Το σύστημα που κατασκευάσθηκε έχει ήδη χρησιμοποιηθεί ως η πρώτη έκδοση του Δίγλωσσου Εκπαιδευτικού Ηλεκτρονικού Εγχειριδίου Συντήρησης και Αποκατάστασης (Conservation REstoration BIlingual Training ELectronic handbook - CREBITEL), στα πλαίσια του προγράμματος CRISATEL.

1.5 Σχετικά Συστήματα

Συστήματα που να σχετίζονται με το παρόν όσον αφορά το πεδίο εφαρμογής υπάρχουν ελάχιστα και έχουν υλοποιηθεί αποκλειστικά στα πλαίσια ευρωπαϊκών προγραμμάτων σε συνεργασία με το C2RMF (Centre de Recherche et de Restauration des Musees de France) και άλλα πολιτιστικά και ακαδημαϊκά ινστιτούτα.

Αρχή έγινε με το έργο NARCISSE [13] και συνέχεια δόθηκε με το CRISTAL [14]. Στο πρώτο, που έλαβε χώρα στις αρχές της δεκαετίας του '90 κατασκευάστηκε ένας αρχικός πολυγλωσσικός θησαυρός κι ένα Cd-rom με την σχετική ορολογία σε 8 γλώσσες εικονογραφημένο με πολυφασματικές εικόνες. Στο CRISTAL (1999-2001) ο πολυγλωσσικός θησαυρός του NARCISSE αναβαθμίστηκε σε ένα οργανωμένο σύστημα συνεργατικής επεξεργασίας εξειδικευμένων λεξιλογίων (σχετικών με ζωγραφικά έργα, γλυπτά κλπ) μέσα από το Web. Παράλληλα έγινε επανέκδοση του Cd-rom «Τέχνη και Επιστήμη» του NARCISSE. Το νέο Cd-rom περέχει εκτός από την εξειδικευμένη ορολογία σε 8 γλώσσες, επιπλέον πολυφασματικές εικόνες έργων συνδυασμένες με

κείμενα σχετικά με τη συντήρησή τους. Για τις πολυφασματικές εικόνες παρέχεται η δυνατότητα εστίασης και μεγέθυνσης (zoom-in) τμημάτων τους. Επιπλέον παρέχεται η δυνατότητα αναζήτησης πάνω σε δημιουργούς, τίτλους και λέξεις κλειδιά. Τα έργα CRISTAL και NARCISSE ασχολήθηκαν αποκλειστικά με την κατασκευή του θησαυρού για το πεδίο εφαρμογής της συντήρησης. Δεν έγινε περαιτέρω προσπάθεια για σύνδεση του θησαυρού με παραδείγματα έργων, εικόνες και κείμενα σε επίπεδο μοντελοποίησης. Τα Cd-rom που κατασκευάσθηκαν ήταν καθαρά τίτλοι πολυμέσων (Macromedia Director), πολύ δύσκολο να συντηρηθούν, να επεκταθούν, πόσο μάλλον να είναι διαθέσιμοι στο Διαδίκτυο.

Αρκετά πιο αξιόλογη προσπάθεια άρχισε να γίνεται το 2001 από το C2RMF με την ανάπτυξη του συστήματος EROS [15]. Το σύστημα αυτό έχει σαν σκοπό να διαχειριστεί αποτελεσματικά τον ολοένα μεγαλύτερο όγκο ψηφιακών δεδομένων που παράγονται σχετικά με έργα τέχνης. Το νέο σύστημα στηρίζεται αποκλειστικά σε κορυφαία "Open Source" πακέτα λογισμικού: Linux OS, Apache, MySQL και PHP, είναι προσβάσιμο από το Διαδίκτυο και υποστηρίζει τόσο HTML όσο και XML. Ο θησαυρός του είναι οργανωμένος σε σύνολα ιεραρχικών λεξικών για κάθε μεταφραζόμενο πεδίο και για κάθε γλώσσα. Η ορολογία είναι αποθηκευμένη στην βάση με τη μορφή σύντομων κωδικών (short codes) και μεταφράζεται στην κατάλληλη γλώσσα όταν πρόκειται να εμφανιστεί σε αποτελέσματα. Το EROS, προς το παρόν τουλάχιστο, παρέχει μόνο λειτουργίες αναζήτησης. Επίσης δεν γνωρίζουμε αν παρέχει τη δυνατότητα να ανακτά δεδομένα όταν οι αναζητήσεις γίνονται στους ευρύτερους από τους όρους των λεξιλογίων. Ως εκ τούτων αποτελεί κυρίως ένα αρχειακό σύστημα (archiving system) για ψηφιακά δεδομένα έργων τέχνης, ομολογουμένως στηριγμένο στο πλέον σύγχρονο λογισμικό. Ωστόσο διατηρούμε μια επιφύλαξη σχετικά με την επιλογή Σχεσιακού ΣΔΒΔ (RDBMS) για την μοντελοποίηση του θησαυρού όρων.

Ξεφεύγοντας από το πεδίο εφαρμογής της συντήρησης, το σύστημα των Jen-Shin Hong, Bai-Hsuen Chen και Jieh Hsiang που περιγράφεται στο [16] είναι ένα αξιόλογο σύστημα για την έκθεση ψηφιακών συλλογών μουσείων με διάφορους τρόπους, το οποίο βασίζεται στις γλώσσες XML-XSL. Το σύστημα αυτό επικεντρώνεται στην κατασκευή XSL αρχείων ανά κατηγορία χρηστών, στη διαχείριση των χρηστών αυτών και στην διασύνδεση με ψηφιακές συλλογές αντικειμένων μουσείων (XML έγγραφα και αντικείμενα πολυμέσων) με τη χρήση πράκτορα παρουσίασης. Η ιδέα του συστήματος αυτού βοήθησε αρκετά στην σχεδίαση της αρχιτεκτονικής για την δική μας εργασία.

1.6 Οργάνωση της Εργασίας

Στο δεύτερο κεφάλαιο της παρούσας εργασίας παρουσιάζεται και υποστηρίζεται η επιλογή εργαλείων που έγινε για την ανάπτυξη του παρόντος ΠΣΒΠΙ.

Στο τρίτο κεφάλαιο περιγράφεται αναλυτικά το σημασιολογικό μοντέλο που χρησιμοποιήθηκε ως η βάση για το ξεκίνημα της υλοποίησης.

Στο τέταρτο κεφάλαιο περιγράφεται η οργάνωση της δικτυακής εφαρμογής, αναλύεται ο σχεδιασμός, η υλοποίηση και η απόδοση των διαφόρων λειτουργιών, παρουσιάζονται τα διάφορα χαρακτηριστικά-κλειδιά της εφαρμογής και πώς επετεύχθησαν.

Στο πέμπτο κεφάλαιο παρατίθεται ένας προτεινόμενος σχεδιασμός για την υλοποίηση της εισαγωγής δεδομένων στο σύστημα με τη χρήση "XML Entry-Forms".

Στο έκτο και τελευταίο κεφάλαιο αναφέρονται βελτιώσεις και μελλοντικές επεκτάσεις του συστήματος.

Κεφάλαιο 2

Επιλογή Εργαλείων – Τεχνολογιών – Αρχιτεκτονικής

Στο κεφάλαιο αυτό υποστηρίζεται η επιλογή εργαλείων που κάναμε για την ανάπτυξη του παρόντος ΠΣΒΠΙ. Η ύπαρξη πολλών βαθμών ελευθερίας στο πρόβλημα της επιλογής που αρχικά διαφαίνονταν λόγω της πληθώρας των τεχνολογιών ανάπτυξης δικτυακών εφαρμογών, περιορίστηκε μετά την επιλογή του ΣΔΒΔ και εξηγούμε το γιατί. Η επιλογή του ΣΔΒΔ αιτιολογείται με βάση τις ιδιαιτερότητες των προς μοντελοποίηση δεδομένων και τις ανάγκες πρόσβασης σ' αυτά. Η επιλογή αρχιτεκτονικής γίνεται μετά από καθορισμό των απαιτήσεων που υπάρχουν για το συγκεκριμένο ΠΣΒΠΙ και επισκόπηση παλιών αλλά κυρίως εξέταση σύγχρονων τεχνολογιών ανάπτυξης. Ακολουθεί έρευνα αγοράς για τους εξυπηρετητές εφαρμογής και σχολαστική εξέταση του προϊόντος Apache Tomcat και της τεχνολογίας των Servlets. Τέλος μετά από εξέταση των γλωσσών περιγραφής και μορφοποίησης δεδομένων, που αναμένεται να κυριαρχήσουν στο WWW τα επόμενα χρόνια, περιγράφεται συνολικά η αρχιτεκτονική του συστήματός μας.

2.1 Επιλογή ΣΔΒΔ

Η φύση του προς κατασκευή ΠΣΒΠΙ ως «πολύγλωσσος σημασιολογικός χάρτης» και ως εκπαιδευτικό εργαλείο για τους συντηρητές έργων τέχνης γεννάει ένα σύνολο προκλήσεων ως αναφορά την πρόσβαση στα δεδομένα που αυτό θα περιέχει [17].

Αρχικά βασική είναι η ανάγκη για συντήρηση ενός πολύγλωσσου θησαυρού με τους όρους του συγκεκριμένου πεδίου εφαρμογής. Οι όροι πρέπει να είναι οργανωμένοι σε σημασιολογικές πολυ-ιεραρχίες, ώστε να επιτρέπεται η ανάκτηση δεδομένων σχετιζόμενων με στενότερους όρους (narrower terms) όταν επιλέγονται οι ευρύτεροι αυτών (broader terms). Σημαντικό σημείο στη δόμηση του θησαυρού κρίνεται και η σημείωση όρων-κλειδιά βάσει των οποίων θα φτιαχτούν σημασιολογικοί χάρτες για επισκόπηση των περιεχομένων του συστήματος.

Επίσης σημαντική κρίνεται η ύπαρξη πολλαπλών διαστάσεων πρόσβασης στα περιεχόμενα του συστήματος. Οι διαστάσεις αυτές θα είναι χαρακτηριστικά των έργων τέχνης όπως υλικά (Materials) και τεχνικές κατασκευής (Elaboration techniques), φαινόμενα σχετιζόμενα με αυτά όπως αλλοιώσεις (Alterations) και συντηρήσεις (Conservations), αλλά και πολυφασματικές μέθοδοι εξέτασης (Methods). Ο χρήστης-

συντηρητής θα πρέπει ακολουθώντας κάποια από τις διαστάσεις αυτές να μπορεί να φτάσει στο σημείο του ενδιαφέροντός του εύκολα και γρήγορα.

Τέλος πολύ σημαντικό στοιχείο του συστήματος είναι και η αναπαράσταση της συσσωρευμένης γνώσης των συντηρητών για το τι διάγνωση μπορεί να γίνει και με ποια μέθοδο, με ταυτόχρονη εμφάνιση παραδειγμάτων. Το γεγονός αυτό συνεπάγεται επιπλέον συσχετίσεις μεταξύ των όρων του θησαυρού.

Όλα τα παραπάνω εξειδικευμένα χαρακτηριστικά στην αναπαράσταση των δεδομένων του πεδίου εφαρμογής (θησαυρός, πολυ-ιεραρχίες, αναπαράσταση γνώσης, όροι κλειδιά), φωτογραφίζουν την ανάγκη χρήσης ενός ΣΔΒΔ βασισμένου σε μοντέλο δεδομένων σημασιολογικού δικτύου (semantic network data model). Η έλλειψη των κατάλληλων μηχανισμών στο σχεσιακό μοντέλο δεδομένων καθιστά δύσχρηστα τα εμπορικά RDBMS για μια τέτοια εφαρμογή. Αντίθετα στο SIS υπάρχουν όλοι εκείνοι οι μηχανισμοί για την υποστήριξη των ζητούμενων χαρακτηριστικών.

2.1.1 Σύστημα Σημασιολογικού Ευρετηριασμού – SIS

Το SIS είναι ένα εργαλείο για την περιγραφή και τεκμηρίωση ταχέως αναπτυσσόμενων ετερογενών πληθυσμών δεδομένων (με υψηλό βαθμό συσχέτισης), εννοιών και σύνθετων συσχετίσεων, σε αντίθεση με τα παραδοσιακά ΣΔΒΔ που χειρίζονται ομογενείς πληθυσμούς δεδομένων. Σαν τέτοιο το SIS είναι το πλέον κατάλληλο εργαλείο για την αναπαράσταση κάθε είδους επιστημονικής γνώσης, δεδομένων που πρέπει να πούμε ότι χαρακτηρίζονται από τη σχετικά μικρή συχνότητα ανανεώσεων [11].

Ο μηχανισμός μόνιμης αποθήκευσης δεδομένων του SIS είναι βασισμένος στο οντοκεντρικό μοντέλο δεδομένων σημασιολογικού δικτύου. Η εισαγωγή και ανάκτηση δεδομένων σε διάφορες μορφές γίνεται με τη χρήση διαλογικής διεπαφής χρήστη. Το SIS διαθέτει πλούσιους μηχανισμούς αναφοράς και ενσωματωμένους μηχανισμούς κληρονομικότητας και εμφανίζει πολύ μεγάλη ταχύτητα στις επερωτήσεις που βασίζονται σ' αυτούς.

Τα κυριότερα χαρακτηριστικά του SIS που ενδιαφέρουν άμεσα για την κατασκευή του παρόντος συστήματος είναι:

Ομοιόμορφη μεταχείριση δεδομένων-σχήματος: Αυτό επιτρέπει καθορισμό και τροποποίηση του σχήματος κατά την ώρα εκτέλεσης (at runtime), καθώς και εμφάνιση του σχήματος στον χρήστη και πλοήγηση σ' αυτό.

Ισχυρή Αναπαράσταση Γνώσης: Από το μοντέλο δεδομένων παρέχονται μηχανισμοί αναπαράστασης γνώσης όπως: απεριόριστη ιεραρχία ταξινόμησης, πολλαπλή κληρονομικότητα, σχέσεις γενίκευσης/εξειδίκευσης, πλειότιμα γνωρίσματα που μπορούν με τη σειρά τους να έχουν γνωρίσματα.

Δικτυοκεντρικός μηχανισμός αναζήτησης: Από το σύστημα επερωτήσεων υποστηρίζεται η αναζήτηση με πολλαπλές και αναδρομικές συνθήκες, όπως και η πλοήγηση στο σημασιολογικό δίκτυο των συσχετίσεων.

Έννοια δοσοληψίας (transaction): Οι ενημερώσεις στηρίζονται στην έννοια της δοσοληψίας και έτσι αποκλείονται οι περιπτώσεις ασυνέπειας της βάσης. Υποστηρίζεται επίσης η ταυτόχρονη πρόσβαση πολλών χρηστών (multi-user access).

Διεπαφή Προγραμματισμού Εφαρμογής (SIS-API): βλέπε **§2.1.3**

Μαζική εισαγωγή/εξαγωγή δεδομένων: Δεδομένα μπορούν να φορτωθούν στο SIS με τη μορφή Telos εντολών. Στην ίδια μορφή υπάρχει και εξαγωγή δεδομένων από το SIS.

2.1.2 Telos – Μια γλώσσα αναπαράστασης γνώσης

Η Telos [18] είναι μια γλώσσα αναπαράστασης γνώσης εξέλιξη των γλωσσών RML [19] (Requirements Modeling Language) και CML [20] (Conceptual Modeling Language). Ουσιαστικά αποτελεί μια «καθαρή» έκδοση της CML από πλευράς ορισμού και υλοποιησιμότητας. Έχει υλοποιηθεί και χρησιμοποιηθεί στα πλαίσια των προγραμμάτων LOKI και DAIDA και έχει ολοκληρωθεί με διαφορετικές υλοποιήσεις στα συστήματα ConceptBase [21] και SIS (**§2.1.1).**

Στην Telos τα πάντα είναι *αντικείμενα (Objects)* με κύρια διάκριση των σε *οντότητες (Individuals)* και *γνωρίσματα (Attributes)*.

Οι *οντότητες* παριστάνουν πράγματα, φαινόμενα ή έννοιες και μπορεί να είναι συγκεκριμένες – όταν παριστάνουν ατομικές υπάρξεις – οπότε καλούνται *άτομα (tokens),* ή αφηρημένες – όταν παριστάνουν κατηγορίες-σύνολα ομοειδών αντικειμένων – οπότε καλούνται *κλάσεις (classes)*.

Τα *γνωρίσματα* παριστάνουν ιδιότητες ή διμελείς σχέσεις μεταξύ αντικειμένων. Κάθε γνώρισμα *p* αποτελείται από μια *αφετηρία (from(p)),* μια *ετικέτα (label(p))* και ένα *προορισμό* (*to(p)*) και αναπαρίσταται ως 3-άδα (3-tuple) για παράδειγμα <Bill,homeaddress,"Lysistratis 33">.

Τα αντικείμενα οργανώνονται στη βάση πληροφοριών με χρήση των τριών βασικών μηχανισμών αφαίρεσης της εννοιολογικής μοντελοποίησης που είναι η *ταξινόμηση (Classification)*, η *γενίκευση (Generalization)* και η *συγκρότηση (Aggregation)*.

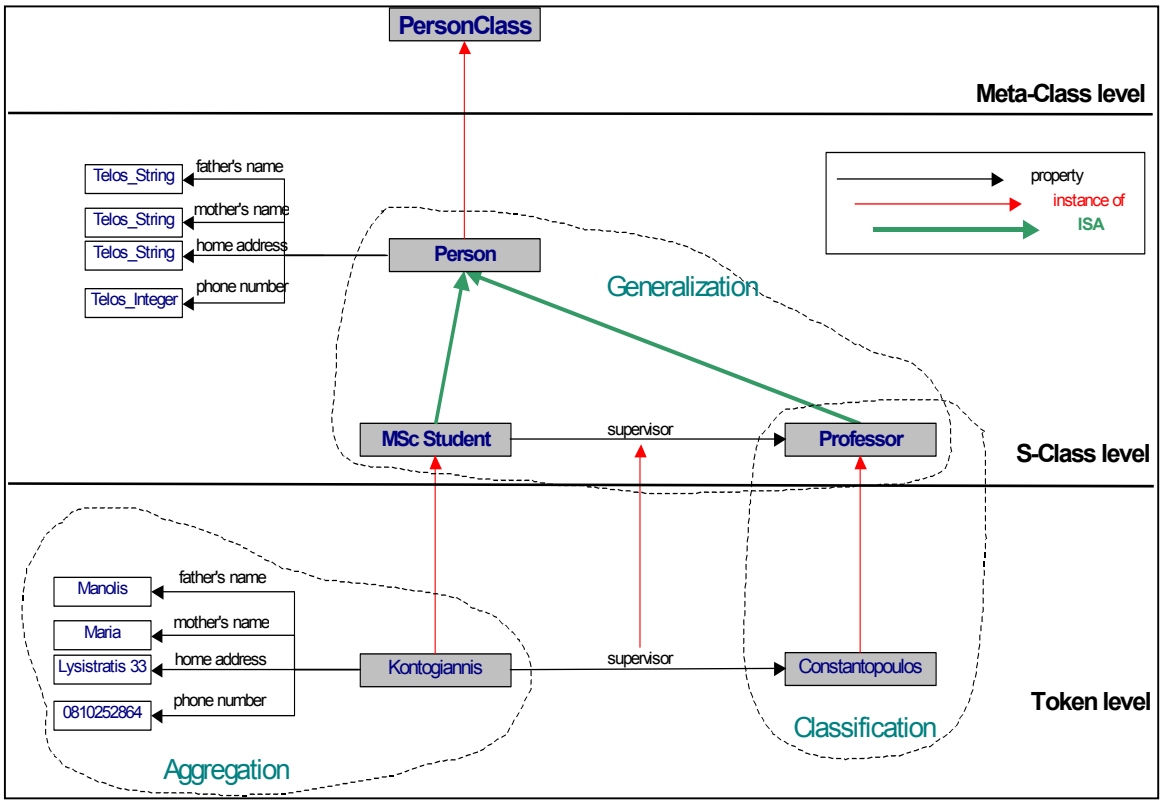
Η *ταξινόμηση* απαιτεί κάθε αντικείμενο να δηλώνεται ως *περίπτωση* ή *μέλος (instance)* μιας ή περισσοτέρων *κλάσεων*. Οι *κλάσεις* είναι και οι ίδιες *αντικείμενα* οπότε είναι κι αυτές περιπτώσεις πιο αφηρημένων *κλάσεων*. Γενικά τα αντικείμενα ταξινομούνται σε ***άτομα (tokens) :*** δεν έχουν *περιπτώσεις/μέλη* και αναπαριστούν διακριτές οντότητες, ***απλές κλάσεις (simple classes):*** αντικείμενα που αποτελούνται μόνο από *άτομα,* ***μετακλάσεις (metaclasses):*** έχουν ως περιπτώσεις μόνο *απλές κλάσεις,* ***μεταμετακλάσεις*** *κ.ο.κ.*. Ο μηχανισμός της *ταξινόμησης* δημιουργεί έτσι μια γραμμική ιεραρχία σταθμών, τη *στάθμη ατόμων (Token level)*, τη *στάθμη απλών κλάσεων (S-Class*

level), τη *στάθμη μετακλάσεων (M1-Class level)* κ.ο.κ. Υπάρχουν επίσης και κλάσεις των οποίων οι *περιπτώσεις/μέλη* μπορούν να ανήκουν σε διαφορετικές στάθμες και λέγονται *ω-κλάσεις (ω-classes)*.

Οι *κλάσεις* μπορούν να εξειδικευθούν μέσα σε *ιεραρχίες γενίκευσης (ISA hierarchies)*. Για παράδειγμα **Person** μπορεί να έχει *υποκλάσεις* **Professor**, **MSc Student.** Η *υπερκείμενη κλάση (superclass)* έχει συνήθως περισσότερες από μία *υποκείμενες (subclasses)* που μπορεί να έχουν ή να μην έχουν επικάλυψη μεταξύ τους. Ένα αντικείμενο που ανήκει σε μια *υποκείμενη κλάση* ανήκει και στην *υπερκείμενή* της. Ο μηχανισμός της γενίκευσης επιτρέπει την συστηματοποίηση των *κλάσεων*, την οικονομία των δηλώσεων και τη μείωση των σφαλμάτων στις δηλώσεις [22].

Η *συγκρότηση* είναι μηχανισμός που βασίζεται στις σχέσεις μέρους-όλου και θεωρεί τα αντικείμενα ως συγκροτούμενα από τα μέρη τους. Πέρα των γνησίων σχέσεων μέρους-όλου, μέρη ενός αντικειμένου μπορούν να θεωρηθούν και τα *γνωρίσματα* που έχουν αποδοθεί σ' αυτό.

Στο σχήμα 2.1 φαίνονται με ένα απλό παράδειγμα και οι τρεις μηχανισμοί της εννοιολογικής μοντελοποίησης όπως εμφανίζονται στη γλώσσα Telos.

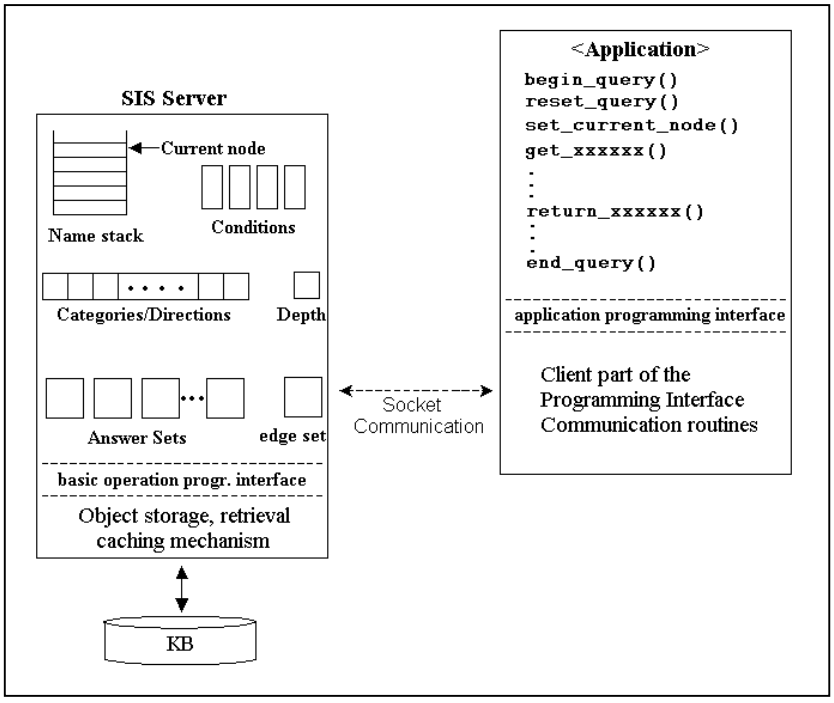


Σχήμα 2.1 - Οι μηχανισμοί αφαίρεσης στην Telos

2.1.3 Διεπαφή Προγραμματισμού Εφαρμογής του SIS

Η Διεπαφή Προγραμματισμού Εφαρμογής του SIS (SIS-API) που σχεδιάστηκε και αναπτύχθηκε στο ICS-FORTH, προσφέρει ένα σύνολο βασικών συναρτήσεων για επερωτήσεις και ανανεώσεις της βάσης του SIS. Οι συναρτήσεις αυτές υλοποιούν συχνά χρησιμοποιούμενους συνδυασμούς στοιχειωδών λειτουργιών στην πλευρά του εξυπηρετητή του SIS [23]. Οι βιβλιοθήκες του SIS-API διατίθενται σε PC-εκδόσεις C, C++ (Borland 5.0.1 βιβλιοθήκες και .dll) και Java.

Υπάρχουν δυο μοντέλα επερωτήσεων για ανάκτηση και τροποποίηση δεδομένων στη βάση του SIS από εξωτερική εφαρμογή. Σημαντικότερο είναι εκείνο του πελάτη-εξυπηρετητή. Σε αυτό το μοντέλο επερωτήσεων ο εξυπηρετητής της βάσης είναι μια ξεχωριστή διεργασία που μπορεί να εκτελείται σε διαφορετικό μηχάνημα από την εφαρμογή που είναι η εφαρμογή-πελάτης (σχήμα 2.2). Η διεργασία αυτή «ακούει» σε κάποια θύρα του μηχανήματος που εκτελείται. Η εφαρμογή στέλνει τις επερωτήσεις που θέλει στον εξυπηρετητή μέσω Windows Sockets και εκείνος αναλαμβάνει να εκτελέσει τις στοιχειώδεις λειτουργίες που αντιστοιχούν σε αυτές. Το μεγαλύτερο ποσοστό των αποτελεσμάτων επερωτήσεων τοποθετείται σε σύνολα-απαντήσεων (answer-sets) τα οποία δεν μεταφέρονται στην εφαρμογή-πελάτη παρά μόνο αν αυτό ζητηθεί μέσω άλλων ειδικών συναρτήσεων. Έτσι μια σύνθετη λειτουργία της εφαρμογής μπορεί να παράξει σύνολα προσωρινών αποτελεσμάτων στον εξυπηρετητή τα οποία όμως δε μεταφέρονται στην εφαρμογή αποφεύγοντας έτσι το κόστος της μεταφοράς άχρηστων δεδομένων μέσω sockets.



Σχήμα 2.2 - Μοντέλο επερωτήσεων του SIS

Επιστρέφοντας στις διατιθέμενες βιβλιοθήκες συναρτήσεων που παρέχονται από το SIS, πρέπει να πούμε ότι στην περίπτωση εφαρμογών Java το λεγόμενο SIS-JAPI βασίζεται στην κλάση QClass που παρέχει ως δημόσιες (public) ένα σύνολο από συναρτήσεις που αντιστοιχούν σε ρουτίνες επερωτήσεων. Ωστόσο επειδή οι τελευταίες είναι υλοποιημένες σε γλώσσα C, υπάρχει ειδικό "wrapper software" που επιτρέπει την διαφανή κλήση των μέσα από εφαρμογές Java ως native συναρτήσεις. Οι ειδικές δομές που χρησιμοποιούνται ως ορίσματα στις ρουτίνες της C έχουν αντιστοιχηθεί με ειδικές κλάσεις υλοποιημένες σε Java (πλην λίγων εξαιρέσεων). Όλες οι απαραίτητες κλάσεις και η δυναμική βιβλιοθήκη με τις C ρουτίνες (.dll) διατίθενται με τη μορφή ενός .jar αρχείου (japi13.jar).

Η κατασκευή του SIS-JAPI με την ανάπτυξη "wrapper software" ήταν απαραίτητη για την εκτέλεση επερωτήσεων ανάκτησης/τροποποίησης δεδομένων από δικτυακές εφαρμογές. Αυτή η προγραμματιστική διεπαφή χρησιμοποιήθηκε τελικά στην παρούσα εργασία.

2.2 Επιλογή Αρχιτεκτονικής Ανάπτυξης

Η επιλογή αρχιτεκτονικής ανάπτυξης έγινε με βασικό γνώμονα τις απαιτήσεις ολοκλήρωσης – σε χρόνο, κόπο και χρήμα – των διαφόρων τεχνολογιών και εργαλείων με το SIS, το οποίο εξηγήσαμε γιατί επιλέξαμε στην παράγραφο **§2.1.** Ωστόσο κάνουμε μια επισκόπηση των αρχιτεκτονικών ανάπτυξης δικτυακών εφαρμογών με πιο συγκεκριμένο τρόπο από το να πούμε ότι είναι N-επίπεδες αρχιτεκτονικές πελάτη-εξυπηρετητή. Επίσης περιληπτικά παρουσιάζουμε παλιές και σύγχρονες τεχνολογίες ανάπτυξης δικτυακών εφαρμογών, ποια εμείς επιλέξαμε και γιατί αποφύγαμε τις υπόλοιπες.

2.2.1 Απαιτήσεις Αρχιτεκτονικής

Οι βασικές απαιτήσεις από την αρχιτεκτονική που θα χρησιμοποιηθεί για το παρόν ΠΣΒΠΙ είναι αυτή να είναι:

Γρήγορη: Η επικοινωνία μεταξύ των διαφόρων εργαλείων που θα ολοκληρωθούν στο σύστημα δεν πρέπει να καθιστούν την αρχιτεκτονική αργή. Το ίδιο ισχύει και για τις τεχνολογίες ανάπτυξης (γλώσσες προγραμματισμού ή scripting).

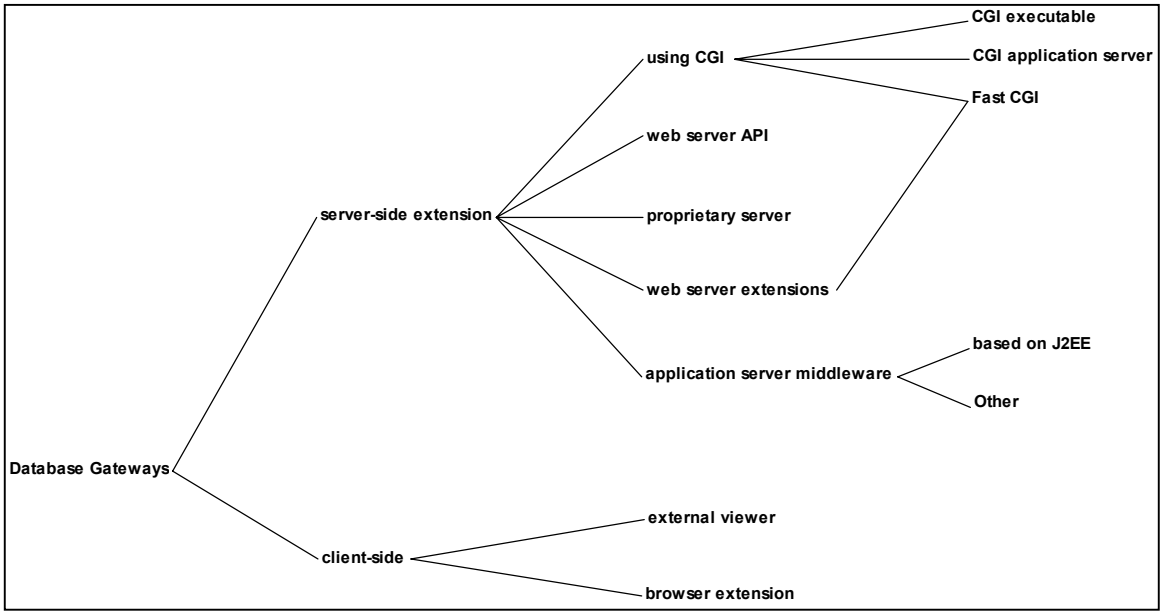
Σταθερή: Τα χρησιμοποιούμενα εργαλεία θα πρέπει να μπορούν να ανταποκριθούν σε περιπτώσεις μεγάλου φόρτου από αιτήσεις χρηστών ή τουλάχιστο να υποστηρίζουν χαρακτηριστικά κατανομής φόρτου που θα χρησιμοποιηθούν αν αυτό χρειαστεί.

Κλιμακούμενη: Τα χρησιμοποιούμενα εργαλεία θα πρέπει να μπορούν να αντικατασταθούν από άλλα, χωρίς να επηρεασθεί το προγραμματιστικό κομμάτι της εφαρμογής, ή/και να υποστηρίζουν διάφορες τεχνολογίες ανάπτυξης δικτυακών

εφαρμογών. Επίσης η ανταλλαγή δεδομένων πρέπει να στηρίζεται σε πρότυπους μορφότυπους δεδομένων.

2.2.2 Επισκόπηση Αρχιτεκτονικών

Στο σχήμα 2.3 φαίνεται μια επέκταση της ταξινόμησης του Kim [24] των αρχιτεκτονικών πυλών διασύνδεσης βάσεων δεδομένων με τον ΠΙ.



Σχήμα 2.3 - Ταξινόμηση αρχιτεκτονικών πυλών διασύνδεσης βάσεων δεδομένων

Συνοπτικά να πούμε ότι οι πύλες διασύνδεσης βάσεων δεδομένων με τον ΠΙ μπορούν να υλοποιηθούν με τοποθέτηση της λογικής επεξεργασίας (process logic) είτε στον εξυπηρετητή είτε στον πελάτη με συνηθέστερο σήμερα το πρώτο. Ακολουθεί μια σύντομη περιγραφή των βασικών στοιχείων των σχετικά παλαιότερων αρχιτεκτονικών.

CGI executable: Αυτή η αρχιτεκτονική χρησιμοποιεί το CGI (Common Gateway Interface), ένα πρότυπο για τη διασύνδεση εκτελέσιμων προγραμμάτων με εξυπηρετητές του ΠΙ. Το πρότυπο αυτό το υποστηρίζουν όλοι οι γνωστοί εξυπηρετητές. Λειτουργεί ως εξής: Κάθε αίτηση χρήστη που φτάνει στον εξυπηρετητή κωδικοποιημένη σε URL τον ενεργοποιεί ώστε να κάνει "fork" την αντίστοιχη διεργασία-εκτελέσιμο πρόγραμμα. Στη συνέχεια ο εξυπηρετητής περνάει μέσω του CGI στο εκτελέσιμο πρόγραμμα τμήμα του URL (αυτό με τις επιπλέον παραμέτρους της αίτησης) και περιμένει την ικανοποίηση της αίτησης. Το εκτελέσιμο πρόγραμμα, που μπορεί να είναι γραμμένο σε κάποια μεταφραζόμενη γλώσσα προγραμματισμού όπως ESQL/C, C, C++, Java, αναλαμβάνει πλήρως τη σύνδεση με την υποκείμενη βάση δεδομένων, το σχηματισμό των επερωτήσεων βάσει του URL, τη συλλογή των αποτελεσμάτων και τη μορφοποίησή τους

ως σελίδες HTML. Πλεονεκτήματα της αρχιτεκτονικής είναι η απλότητα λόγω έλλειψης διαμοιρασμού πόρων και η επεκτασιμότητα της λειτουργικότητας της πύλης διασύνδεσης. Μειονεκτήματα είναι η άπληστη κατανάλωση πόρων του συστήματος και τα χρονικά κόστη δημιουργίας και τερματισμού κάθε διεργασίας, αντιγραφής δεδομένων μεταξύ διεργασιών και ανοίγματος και κλεισίματος συνδέσεων με τη βάση δεδομένων.

CGI application server: Στην αρχιτεκτονική αυτή αποσπάται από τα εκτελέσιμα CGI η λογική επεξεργασίας και σύνδεσης με τη βάση. Έτσι μια πύλη διασύνδεσης που ακολουθεί αυτή την αρχιτεκτονική αποτελείται από δύο επίπεδα. Στο ένα υπάρχουν μικρά εκτελέσιμα CGI που λέγονται διανεμητές (dispatchers) και στο άλλο υπάρχει ένας ή και περισσότεροι εξυπηρετητές εφαρμογής που ασχολούνται αποκλειστικά με τις επερωτήσεις στη βάση και τη μορφοποίηση των αποτελεσμάτων. Κάθε αίτηση χρήστη δημιουργεί πλέον 1 μόνο διεργασία με μικρές απαιτήσεις σε πόρους, τον διανεμητή, ο οποίος αποφασίζει σταθμίζοντας το φόρτο των εξυπηρετητών (load balancing) σε ποιον θα αποστείλει την αίτηση. Ο/οι εξυπηρετητές εφαρμογής τρέχουν ως «δαίμονες» (daemons), οπότε οι αντίστοιχες διεργασίες δημιουργούνται μόνο μια φορά. Πλεονεκτήματα της αρχιτεκτονικής είναι η μικρότερη κατανάλωση πόρων και η αποφυγή του χρονικού κόστους ανοίγματος/κλεισίματος συνδέσεων με τον εξυπηρετητή της βάσης. Μειονεκτήματα είναι η ανάγκη συνεργασίας πολλών εφαρμογών για την εξυπηρέτηση μιας αίτησης (χρονική καθυστέρηση), αλλά και η ανάγκη αναμονής για την εξυπηρέτηση μιας αίτησης όταν προηγούνται άλλες και δεν υπάρχουν ελεύθεροι εξυπηρετητές εφαρμογής (το πρόβλημα είναι οξύ όταν υπάρχει μόνο ένας εξυπηρετητής και ο δικτυακός τόπος είναι δημοφιλής).

Fast CGI [25]**:** Η αρχιτεκτονική αυτή στηρίζεται στο σχήμα των επίμονων (persistent) CGI διεργασιών. Σημαντικό είναι ότι υποστηρίζεται από τους περισσότερους web-servers με την ενσωμάτωση κάποιου module (όπως το *mod_fastcgi* για τον Apache). Σ' αυτή την αρχιτεκτονική υπάρχουν 2 είδη Fast CGI διεργασιών, οι στατικές και οι δυναμικές. Οι στατικές δημιουργούνται με την εκκίνηση του web-server και «μπλοκάρουν» περιμένοντας αιτήσεις. Για κάθε αίτηση που φτάνει στον web-server, αναλαμβάνει μια στατική διεργασία να περάσει την αίτηση σε κάποια δυναμική διεργασία ανοίγοντας σύνδεση με αυτή. Αν δεν υπάρχει κατάλληλη δυναμική διεργασία να εκτελείται ή αν δεν υπάρχει κατάλληλη ελεύθερη, τότε η στατική διεργασία αναλαμβάνει να ξεκινήσει μια επιπλέον δυναμική διεργασία του απαιτούμενου τύπου. Σε αρχεία ρυθμίσεων καθορίζεται το μέγιστο πλήθος δυναμικών διεργασιών που μπορούν να γίνουν "fork". Οι δυναμικές διεργασίες δεν τερματίζονται όταν τελειώσουν με την εξυπηρέτηση μιας αίτησης απλά κλείνουν τη σύνδεση με τη στατική διεργασία και περιμένουν για άλλη ίδιου τύπου αίτηση. Αν σε κάποια στιγμή λόγο φόρτου ξεκινήσουν πολλές δυναμικές διεργασίες και στη συνέχεια μειωθεί ο φόρτος τότε κάποιες δυναμικές διεργασίες «σκοτώνονται».

Επίσης όταν κάποια αίτηση δεν μπορεί να εξυπηρετηθεί άμεσα, διότι δεν υπάρχουν ελεύθερες στατικές διεργασίες ή υπάρχει μέγιστος αριθμός δυναμικών διεργασιών, τότε τοποθετείται σε ουρά στατικής διεργασίας ή σε ουρά μέσα στον web-server αντίστοιχα. Πλεονεκτήματα της αρχιτεκτονικής είναι ο περιορισμός της δημιουργίας και καταστροφής διεργασιών, η απομόνωση διεργασιών και ελαχιστοποίηση έτσι της πιθανότητας server crash και η κατανομή φόρτου με την απομακρυσμένη εκτέλεση διεργασιών. Μειονέκτημα όπως και στην περίπτωση του CGI application server είναι η ανάγκη συνεργασίας πολλών διεργασιών για την εξυπηρέτηση μιας αίτησης (χρονική καθυστέρηση).

Server API: Μερικοί web-servers παρέχουν επεκτάσιμη Διεπαφή Προγραμματισμού Εφαρμογής (API) με την οποία οι προγραμματιστές δικτυακών εφαρμογών μπορούν να επαυξήσουν τη λειτουργικότητά τους. Οι εφαρμογές που γράφονται με τη χρήση Server API συνδέονται δυναμικά στους web-servers και φορτώνονται κατά την εκκίνησή αυτών, απαλλάσσοντας εντελώς από το κόστος διαχείρισης διεργασιών (το βασικό πρόβλημα του CGI). Η αρχιτεκτονική αυτή είναι γρήγορη σε περιπτώσεις χαμηλού φόρτου, αλλά καθιστά τον web-server σημείο συμφόρησης σε περιπτώσεις μεγάλου φόρτου. Επίσης δεν υπάρχουν καθιερωμένα Server APIs που να προσφέρουν όλοι οι web-servers, οπότε μια δικτυακή εφαρμογή αναπτυγμένη με αυτή την αρχιτεκτονική είναι προσκολλημένη στον web-server στον οποίο αναπτύχθηκε. Παραδείγματα Server APIs είναι το NSAPI του Netscape [26] και το ISAPI του Microsoft IIS (Internet Information Server) [27].

Proprietary Server: Στην αρχιτεκτονική αυτή ο πελάτης της βάσης δεδομένων (database client) και ο εξυπηρετητής του ΠΙ είναι ολοκληρωμένοι σε μια εφαρμογή. Με απλά λόγια η αρχιτεκτονική αυτή είναι εξειδικευμένη για ένα συγκεκριμένο ΣΔΒΔ και καταλαβαίνει την τεχνολογία του Διαδικτύου υποστηρίζοντας άμεσα το πρωτόκολλο HTTP. Πλεονέκτημα της αρχιτεκτονικής είναι η ταχύτητα. Μειονεκτήματα είναι η δυσκολία ανάπτυξης (πρέπει να φτιαχτούν έστω και απλουστευμένες οι λειτουργίες ενός web-server) και συντήρησης (σε περιπτώσεις αλλαγών σε πρότυπα και πρωτόκολλα).

External Viewer: Η αρχιτεκτονική αυτή εκμεταλλεύεται τη δυνατότητα των φυλλομετρητών του ΠΙ να εκτελούν εξωτερικές εφαρμογές αν αυτό οριστεί κατάλληλα. Έτσι μπορεί μέσω φυλλομετρητή να γίνει κλήση προγραμμάτων-πελατών βάσεων δεδομένων και με αυτό τον τρόπο να γίνει η ανάκτηση/τροποποίηση των δεδομένων. Η επικοινωνία μεταξύ πελάτη και εξυπηρετητή είναι άμεση χωρίς παρεμβολή εξυπηρετητή του ΠΙ (ταχύτητα) και χρήση πρωτοκόλλων και προτύπων όπως το HTTP και η HTML. Μειονεκτήματα αυτής της αρχιτεκτονικής είναι η δυσκολία εφαρμογής τεχνικών εξισορρόπησης φόρτου και αποτελεσματικής διαχείριση συνδέσεων με τον εξυπηρετητή της βάσης.

Browser Extension: Στην αρχιτεκτονική αυτή οι περιορισμοί της γλώσσας HTML αντιμετωπίζονται σε κάποιο βαθμό από επεκτάσεις του φυλλομετρητή του ΠΙ. Τέτοιες επεκτάσεις μπορεί να είναι διερμηνευτές γλωσσών scripting ή συναρτήσεις πρόσβασης σε βάσεις δεδομένων. Έτσι μια αίτηση του πελάτη μιας δικτυακής εφαρμογής μπορεί να έχει σαν αποτέλεσμα τη λήψη μιας σελίδας HTML με ειδικές ετικέτες διαμόρφωσης "script tags", που διερμηνευόμενες από τον φυλλομετρητή ενεργοποιούν κατάλληλες συναρτήσεις πρόσβασης σε απομακρυσμένες βάσεις δεδομένων. Μειονέκτημα αυτής της αρχιτεκτονικής είναι η έλλειψη συντονισμένων δραστηριοτήτων για καθιέρωση συγκεκριμένων επεκτάσεων στους φυλλομετρητές του ΠΙ. Σ' αυτή την αρχιτεκτονική μπορούν να καταταχθούν και τα λεγόμενα Java applets, δυαδικός κώδικας (bytecode) που φορτώνεται στον πελάτη όταν περιλαμβάνεται στη σελίδα HTML που λαμβάνει κατάλληλη ετικέτα διαμόρφωσης. Ο δυαδικός κώδικας εκτελείται από την ενσωματωμένη στον φυλλομετρητή εικονική μηχανή Java (Java Virtual Machine).

2.2.3 Σύγχρονες τάσεις

Οι αρχιτεκτονικές που περιγράφηκαν προηγουμένως εμφανίζουν για κάποιους λόγους είτε προβλήματα απόδοσης, είτε προβλήματα κλιμάκωσης, είτε προβλήματα μεταφερσιμότητας. Αυτά τα χαρακτηριστικά όμως είναι επιθυμητά από την αρχιτεκτονική μιας δικτυακής εφαρμογής. Άλλο πολύ σημαντικό χαρακτηριστικό, ειδικά σε ΠΣΒΠΙ πολύ μεγάλης κλίμακας (very large scale) είναι η κεντρικοποίηση της λογικής επεξεργασίας σε ένα εξυπηρετητή ή σε μια ομάδα εξυπηρετητών (cluster of servers). Αυτό είναι απαραίτητο για την αποτελεσματική διαχείριση συνδέσεων χρηστών καθώς και τον έλεγχο και κατανομή του φόρτου στους εξυπηρετητές.

Τα παραπάνω αιτιολογούν γιατί έχει επικρατήσει τα τελευταία 2-3 χρόνια η ανάπτυξη πυλών διασύνδεσης βάσεων δεδομένων με τον ΠΙ με "server side extensions" (βλ. σχήμα 2.3). Ειδικότερα ο λεγόμενος δικτυακός προγραμματισμός (web-programming) τείνει να ακολουθεί τις αρχιτεκτονικές που βασίζονται στις επεκτάσεις που κάναμε στην ταξινόμηση του Kim [24] δηλαδή στα **web server extensions** και στο **application server middleware**.

web server extensions: Αυτή η αρχιτεκτονική στηρίζεται στην ιδιότητα διαφόρων εξυπηρετητών του ΠΙ να ενσωματώνουν επεκτάσεις-υπομονάδες (modules) χειρισμού αιτήσεων ή τμημάτων αιτήσεων χρηστών. Οι αιτήσεις των χρηστών φτάνουν στον εξυπηρετητή του ΠΙ και συγκεκριμένα στο τμήμα-πυρήνα (core) και η εξυπηρέτησή τους περνά από ένα σύνολο φάσεων, ελεγχόμενων από τον πυρήνα (URI to filename translation, Authorization, MIME type determination etc). Σε κάθε μια απ' αυτές τις φάσεις κάθε υπομονάδα μπορεί να έχει ορίσει μια ενέργεια χειρισμού (handler) για τις αιτήσεις, που ενεργοποιείται όταν έρθει η συγκεκριμένη φάση. Παραδείγματα υπομονάδων είναι μεταξύ άλλων και οι διερμηνευτές γλωσσών scripting που

αναλαμβάνουν να διερμηνεύσουν είτε ολόκληρες σελίδες scripting, είτε το περιεχόμενο ειδικών ετικετών που υπάρχουν σε σελίδες HTML. Οι γλώσσες scripting συνήθως παρέχουν τρόπους διασύνδεσης με γνωστές βάσεις δεδομένων μέσω "native drivers" ή μέσω ODBC drivers.

application server middleware: Οι πύλες διασύνδεσης βάσεων δεδομένων που χρησιμοποιούν εξυπηρετητές εφαρμογής ακολουθούν N-επίπεδη αρχιτεκτονική πελάτη-εξυπηρετητή (N≥3). Ένας εξυπηρετητής εφαρμογής είναι ένα προϊόν βασισμένο σε συνιστώσες (component-based) το οποίο βρίσκεται στο μεσαίο επίπεδο μιας αρχιτεκτονικής κεντρικοποιημένης στον εξυπηρετητή (server-centric) [28]. Παρέχει ένα σύνολο υπηρεσιών για την ολοκλήρωση της λογικής παρουσίασης, επεξεργασίας και συνδεσιμότητας με βάσεις δεδομένων. Επιτρέπει στον προγραμματιστή μιας δικτυακής εφαρμογής να ασχολείται μόνο με την αποστολή επερωτήσεων ανάκτησης/τροποποίησης δεδομένων προς το υποκείμενο ΣΔΒΔ και τη συλλογή, επεξεργασία και μορφοποίηση των αποτελεσμάτων. Τον απαλλάσσει από επίπονες προγραμματιστικές τεχνικές παρέχοντάς του χαρακτηριστικά όπως:

- Έλεγχος ροής και προσωρινής αποθήκευσης δεδομένων (data flow caching and control)
- Πολυνηματικές διεργασίες (multithreading, thread pooling)
- Διαχείριση συνόδων (session management)
- Υποστήριξη βάσεων δεδομένων μέσω ODBC, JDBC, native drivers (database support)
- Υποστήριξη πρωτοκόλλων όπως CORBA, RMI κλπ (protocol support)
- Εξισορρόπηση και κατανομή φόρτου (load balancing, dispatching)
- Υποστήριξη δέσμης εξυπηρετητών και μηχανισμών ξεπεράσματος βλαβών (clustering and failover mechanisms).
- Διαχείριση ασφάλειας (security management)
- Διαχείριση δοσοληψιών (transaction management)

Πολλοί εξυπηρετητές εφαρμογής παρέχουν και εργαλεία ταχείας ανάπτυξης και δημοσιοποίησης δικτυακών εφαρμογών, κονσόλες παρακολούθησης πελατών και φόρτου ομάδας εξυπηρετητών (server cluster). Μερικοί επίσης ολοκληρώνουν μέσα τους και εξυπηρετητές ΠΙ οπότε τότε καλούνται εξυπηρετητές εφαρμογής ΠΙ.

Οι δυο προαναφερθείσες αρχιτεκτονικές ανάπτυξης ΠΣΒΠΙ ενσαρκώνονται κατά κόρον στις μέρες μας με τη χρήση των τεχνολογιών προγραμματισμού που παρουσιάζουμε στις επόμενες δυο παραγράφους.

2.2.3.1 Scripting γλώσσες προγραμματισμού

Υπάρχουν διάφορες γλώσσες scripting που χρησιμοποιούνται ευρέως στην ανάπτυξη δικτυακών εφαρμογών εκμεταλλευόμενες την υποστήριξη των διαφόρων εξυπηρετών του ΠΙ. Οι γνωστότερες απ' αυτές είναι οι Perl, PHP, ASP, JSP [29].

Perl: είναι μια από τις παλαιότερες και πιο επιτυχημένες διερμηνευόμενες γλώσσες προγραμματισμού (script-based), υποστηριζόμενη σε όλες σχεδόν τις πλατφόρμες και η οποία τυγχάνει ιδιαίτερης εκτίμησης από τους ειδικούς σε θέματα προγραμματισμού. Υπάρχει πληθώρα επεκτάσεων (modules) για την Perl που την κάνουν να υπερέχει ως μέσο έκφρασης-συγγραφής προγραμμάτων. Χαρακτηριστικό παράδειγμα η ανεξάρτητη-βάσης βιβλιοθήκη DBI της Perl που ανεξαρτητοποιεί πλήρως τον προγραμματιστή από τη λειτουργία των εξειδικευμένων σε συγκεκριμένα ΣΔΒΔ οδηγών (drivers). Η Perl είναι η καλύτερη επιλογή για την ανάπτυξη μεγάλου όγκου λογισμικού και τη συντήρησή του μακροπρόθεσμα, ωστόσο είναι πιο δύσκολη από άλλες γλώσσες στην εκμάθησή της και δεν ενδείκνυται για την κατασκευή απλών δυναμικών εργασιών σε δικτυακούς τόπους. Ο πιο δημοφιλής εξυπηρετητής του ΠΙ Apache ενσωματώνει διερμηνευτή της Perl στην επέκτασή του mod_perl.

PHP: είναι μια από τις πιο δημοφιλείς διερμηνευόμενες γλώσσες ανοικτού κώδικα. Οι σελίδες PHP δουλεύουν πολύ καλά τόσο σε πλατφόρμες Unix-like όσο και σε πλατφόρμες Windows, με αποτέλεσμα να είναι πρόσφορη η ανάπτυξή τους σε επιτραπέζιους προσωπικούς υπολογιστές (desktop PCs) και έπειτα δημοσιοποίησή τους σε βιομηχανικής ισχύος (industrial-strength) εξυπηρετητές που συνήθως τρέχουν Unix-Linux. Τα δυνατά χαρακτηριστικά της PHP είναι η ευκολία χρήσης και ανάπτυξης, καθώς και η σημαντική υποστήριξή της από τη βιομηχανία. Αρνητικά της σημεία είναι η έλλειψη ομοιόμορφης πρόσβασης στις διάφορες βάσεις δεδομένων, ο ελάχιστα αναπτυγμένος μηχανισμός βιβλιοθηκών της και κάποιες γλωσσικές ιδιοτροπίες. Ο πιο δημοφιλής εξυπηρετητής του ΠΙ Apache ενσωματώνει διερμηνευτή της PHP στην επέκτασή του mod_php.

ASP(Active Server Pages) / VBscript: είναι ένα προϊόν της Microsoft πολύ διαδεδομένο για προγραμματισμό στον εξυπηρετητή. Παρόλο που αποτελεί ένα πλαίσιο προγραμματισμού (programming framework) στο οποίο μπορούν να χρησιμοποιηθούν πολλές γλώσσες (όπως η Perl), στην πλειονότητα των περιπτώσεων χρησιμοποιείται η VBscript, γλώσσα ειδικά κατασκευασμένη για την ανάπτυξη δικτυακών εφαρμογών σε πλατφόρμες Microsoft Windows. Υποστηρίζεται πλήρως μόνο στο προϊόν IIS (Internet Information Server) της ίδιας εταιρείας. Δυνατά σημεία του πλαισίου προγραμματισμού ASP είναι η ανεξαρτησία από βάσεις δεδομένων με τη χρήση ODBC και ADO (Active Data Objects), η πληθώρα εργαλείων υποστήριξης και η καλή τεκμηρίωση. Μειονεκτήματα είναι η έλλειψη ανεξαρτησίας πλατφόρμας και το προβληματικό σε θέματα ασφάλειας προϊόν IIS.

JSP(Java Server Pages): Η JSP είναι μια γλώσσα scripting για τους οπαδούς της γλώσσας Java, σχεδιασμένη για μεγάλους δικτυακούς τόπους που συνδέονται σε επιχειρησιακές βάσεις δεδομένων (enterprise databases). Έχει μια πληθώρα εργαλείων που λέγονται Enterprise JavaBeans (EJBs) που είναι σχεδιασμένα για τη μετακίνηση όλου του scripting κώδικα σε βιβλιοθήκες οι οποίες συνδέονται με τις σελίδες JSP. Οι σελίδες JSP είναι κυρίως σελίδες HTML με διάσπαρτο κώδικα Java μέσα σε ειδικά σύμβολα. Μεγάλο πλεονέκτημα αυτής της γλώσσας είναι η ανεξαρτησία πλατφόρμας.

2.2.3.2 J2EE τεχνολογίες ανάπτυξης

Η Java 2 Platform Enterprise Edition (J2EE) [n29] παρέχει ένα προγραμματιστικό πλαίσιο για την ανάπτυξη εφαρμογών κατανεμημένης αρχιτεκτονικής. Μηχανές που υλοποιούν πλήρως ή μερικώς αυτό το προγραμματιστικό πλαίσιο είναι οι βασισμένοι στη Java εξυπηρετητές εφαρμογών (Java Application Servers).

Οι τεχνολογίες που συνθέτουν την J2EE είναι οι :

1. JDBC (Java Database Connectivity)
2. JSP (Java Server Pages)
3. Java Servlets
4. RMI (Remote Method Invocation)
5. Java IDL/CORBA (Java Interface Definition Language)
6. EJBs (Enterprise JavaBeans)
7. JNDI (Java Naming Directory Interface)
8. JMS (Java Messaging Service)
9. JTS (Java Transaction Service)
10. JTA (Java Transaction Architecture)
11. JavaMail
12. JAF (Java Activation Framework)
13. JAXP (Java API for XML)

Οι τεχνολογίες αυτές είναι υπεραρκετές για τη δημιουργία οποιασδήποτε κλίμακας ΠΣΒΠΙ. Στην ουσία τις περισσότερες φορές αρκεί ένα μικρό υποσύνολό τους για ένα πολύ καλό αποτέλεσμα. Για παράδειγμα ένα ΠΣΒΠΙ με σχεσιακή βάση δεδομένων μπορεί να φτιαχτεί με τη χρήση JSPs και JDBC ή Java Servlets και JDBC. Αν ωστόσο υπάρχει έτοιμος κώδικας για πρόσβαση στα δεδομένα της βάσης γραμμένος σε άλλη γλώσσα προγραμματισμού αυτός μπορεί να χρησιμοποιηθεί με τη βοήθεια της Java IDL/CORBA. Επίσης αν υπάρχουν αντικείμενα που «ζουν» σε απομακρυσμένους υπολογιστές με χρήσιμες λειτουργίες, αυτές μπορούν να κληθούν από μακριά με τη χρήση RMI.

Το σημαντικό είναι ότι η πλατφόρμα J2EE παρέχεται ως πλήρες πακέτο τουλάχιστο στους εμπορικούς Java εξυπηρετητές εφαρμογής και προσφέρει στον προγραμματιστή τη δυνατότητα να χρησιμοποιήσει τις τεχνολογίες αυτές με σχετικά απλό τρόπο.

2.2.4 Η τελική επιλογή

Η τελική επιλογή μας βασίστηκε κατά κύριο λόγο στο γεγονός ότι το ΣΔΒΔ που θα χρησιμοποιήσουμε για την ανάπτυξη του παρόντος ΠΣΒΠΙ θα είναι το SIS. Αυτό περιορίζει τις επιλογές μας στη χρήση μιας εκ των γλωσσών προγραμματισμού C, C++ και Java.

Τις γλώσσες προγραμματισμού C, C++ θα μπορούσαμε να τις χρησιμοποιήσουμε με κάποια από τις based on cgi αρχιτεκτονικές. Ωστόσο τα προβλήματα απόδοσης, σταθερότητας, κλιμάκωσης, αλόγιστης κατανάλωσης πόρων, εξάρτησης από πλατφόρμα κ.α. που παρουσιάστηκαν σε προηγούμενες παραγράφους μας απέτρεψαν. Αυτός είναι και ο λόγος για τον οποίο δεν προσπαθήσαμε να χρησιμοποιήσαμε το μηχανισμό που περιγράφεται στο [5].

Θεωρητικά θα μπορούσε να κατασκευαστεί ένας proprietary server σε C,C++, ωστόσο κάτι τέτοιο θα ήταν πολύ δύσκολο στην υλοποίηση όσο και στη συντήρησή του, αφού θα έπρεπε να ενσωματώνει έστω και ένα μικρό ποσοστό των λειτουργιών ενός εξυπηρετητή του ΠΙ.

Οι γλώσσες προγραμματισμού scripting (με εξαίρεση την JSP) θα απαιτούσαν για την ολοκλήρωση με το SIS την ιδιαίτερα επίπονη και χρονοβόρα διαδικασία της κατασκευής ενός wrapper για την κλήση C, C++ διαδικασιών μέσα από Perl ή PHP.

Για τους παραπάνω λόγους τελικά προτιμήθηκε η αρχιτεκτονική με **application server middleware**. Υπάρχει μεγάλη γκάμα δοκιμασμένων, αξιόπιστων, σταθερών και γρήγορων εξυπηρετητών εφαρμογής στην παγκόσμια αγορά από τους οποίους τις ανάγκες μας κάλυπταν πλήρως όλοι όσοι ήταν βασισμένοι στην Java.

2.3 Επιλογή Εξυπηρετητή Εφαρμογής

Η πλειονότητα των εξυπηρετητών εφαρμογής είναι βασισμένοι και σχεδιασμένοι γύρω από την πλατφόρμα J2EE και τη γλώσσα προγραμματισμού Java.

Κατά την έρευνα αγοράς που έγινε εντοπίστηκε και ένας αξιόλογος εξυπηρετητής εφαρμογής, με την ονομασία Zope [30], που δεν είναι βασισμένος σ' αυτήν την πλατφόρμα. Ωστόσο οι δυνατότητες για scripting μόνο σε Perl και Python που παρέχει δεν επιτρέπουν την ολοκλήρωσή του με το SIS (κάτι που απαιτεί την κλήση συναρτήσεων C,C++) με απλό τρόπο.

Μεταξύ των Java εξυπηρετών εφαρμογής επιλέξαμε με κριτήρια την καλή τεκμηρίωση, την καλή φήμη, την υποστήριξη των απολύτως απαραίτητων τεχνολογιών για την ανάπτυξη του παρόντος ΠΣΒΠΙ, την υποστήριξη χαρακτηριστικών για την αντοχή σε μεγάλο φόρτο, την διασύνδεση (interfacing) με άλλα εργαλεία και την ύπαρξη εργαλείων για διευκόλυνση του κύκλου ανάπτυξης. Στον πίνακα 2.1 φαίνονται κάποια

στοιχεία για τις τελευταίες εκδόσεις των πιο δημοφιλών από τους εξυπηρετητές εφαρμογής.

Vendor Product	J2EE support		Features
	Cert	Technologies	
BEA Weblogic Server v7.0 [31]	√1.3	Servlet 2.3, JSP1.2, EJB2.0, JMS1.0.2, JDBC2.0, JCA, RMI-IIOP, JAXP1.1	scalability, performance, fault tolerance, security standards support, load balancing tools, connection pooling, own Web server, interface with Apache & IIS
Borland Enterprise Server v5.1 [32]	√1.3	Servlet2.3, JSP1.2, EJB2.0, JMS1.0.2, JDBC2.0, JCA, RMI-IIOP, CORBA, XML	leading-performance, Linear scalability, Application server partitioning, clustering, load balancing, session fail-over, distributed transactions, legacy connectivity, security standards support, application management & monitoring, hot deployment, Borland Web server based on Apache
IBM Websphere Enterprise v4.1 [33]	√1.2.1	Servlet2.2 JSP1.1, EJB1.1, JMS1.0.2, JCA, CORBA, XML	Web services SOAP, WSDL, connection management pooling, clustering, directory services LDAP, falut tolerance, caching for faster transactions, full remote administration, ActiveX, firewall support, Apache-based web server
JBoss JBoss v3.0 [34]	---	EJB container, JMQ (JMS), JMX (mail), JTX (JTA/JTS), JSX (JAAS), JCX (JCA), Jboss CMP2.0, RMI/IIOP, CORBA	embedded HTTP1.1 Web server, hot deploy microkernel, clustering with super-server base, open source.
Macromedia Jrun Server v4 [35]	√1.3	Servlet 2.3, JSP 1.2, EJB 2.0, JMS 1.0.2, JTA1.2, JAAS 1.0, JCA 1.0, JAXP 1.1, JavaMail 1.1, JDBC 2.0, JNDI 1.2, RMI/IIOP1.0	Web services SOAP, WSDL, UDDI, Superior performance, server clustering, load balancing and automatic failover, performance tracing, web-based management console (JMC), point-click server cluster administration, one-step deployment at all clusters, built-in Macromedia Flash MX player connectivity, built-in web server, clustering-enabled web server connectors for Apache, IIS, Netscape Server, iPlanet.

Apache Tomcat v5.0 [12]	---	Servlet 2.4, JSP2.0, JMX, AJP 1.3, JNI	clustering and load balancing using Apache Web server, embedded Web server, Performance and memory efficiency, Jasper JSP page compiler, web application for administration, Enhanced manager application support for integration with development tools, Custom **Ant** interaction with manager application directly with.xml scripts, Security manager/User Realms, SSL, excellent installation and configuration documentation, good reputation, open source.
Zope Zope v2.4.2 [30]	---	---	web development using Python or Perl, DTML markup language, own Web server Zserver, transactional object database, search engine, web page templating system, web-based development and management tools, ODBC, XML, XML-RPC, Fast-CGI, DOM, SOAP

Πίνακας 2.1 - Δημοφιλείς εξυπηρετητές εφαρμογής

Η τελική επιλογή μας ήταν ο εξυπηρετητής εφαρμογής Apache Tomcat ο οποίος ικανοποιεί τα κριτήρια που εξ' αρχής τέθηκαν και εμφανίζει μια ιδιαίτερη δυναμική εξέλιξης σε χαρακτηριστικά, που είναι συνυφασμένη με την ιδιότητά τους ως λογισμικό «ανοικτού κώδικα».

2.3.1 Jacarta Apache Tomcat

Ο εξυπηρετητής Apache Tomcat είναι ένας Servlet Container με αρθρωτή αρχιτεκτονική [36], ο οποίος χρησιμοποιείται στην επίσημη Υλοποίηση Αναφοράς (Reference Implementation) για τις τεχνολογίες Java Servlet και Java Server Pages.

Υπάρχουν τέσσερις οικογένειες συνιστωσών που συνθέτουν την αρχιτεκτονική του εν λόγω εξυπηρετητή, οι *Connector Components*, *Container Components, Utility Components* και *Session Management Components,* οι οποίες περιγράφονται από *Java interface* κλάσεις.

Connector Components: είναι οι διάφοροι τρόποι με τους οποίους ο Tomcat Servlet Container συνδέεται με τον υπόλοιπο κόσμο. Οι βασικές *interface* κλάσεις αυτής της οικογένειας είναι οι *Connector*, *Request* και *Response*. Η κλάση *Connector* είναι η συνιστώσα εκείνη η οποία βάσει ενός πρωτοκόλλου επικοινωνίας ή API αρχικά δέχεται αιτήσεις από εφαρμογές πελατών (μέσω φυλλομετρητή ή εξυπηρετητή του ΠΙ). Στη

συνέχεια δημιουργεί κατάλληλα στιγμιότυπα *Request* και *Response* και θέτει τις ιδιότητές τους βασισμένη στα περιεχόμενα της εισερχόμενης αίτησης. Ακολουθεί η εξεύρεση ενός κατάλληλου *Container* που θα χρησιμοποιηθεί για την επεξεργασία της αίτησης και κλήση της μεθόδου *invoke()* αυτού με ταυτόχρονο πέρασμα των στιγμιότυπων των *Request* και *Response* ως παραμέτρων. Τελικά επιστρέφονται οι επικεφαλίδες και τα δεδομένα της απάντησης *Response* στην εφαρμογή-πελάτη που απέστειλε την αίτηση.

Container Components: Η πρωταρχική ευθύνη ενός *Container (interface)* είναι η εκτέλεση αιτήσεων που λαμβάνονται από κάποιον *Connector* ή από κάποιο πατρικό *Container* και η επιστροφή της αντίστοιχης απάντησης. Αυτό πραγματοποιείται με κλήση της μεθόδου *invoke()* του *Container*. Κάθε *Container* έχει ως χαρακτηριστικά:

- Ένα όνομα που τον διακρίνει από τους *Containers* που έχουν τον ίδιο πατέρα
- Ένα πατέρα *Container* που είναι υπεύθυνος για μεγαλύτερο σύνολο από αιτήσεις από ότι ο ίδιος.
- Ένα σύνολο παιδιών (*Container objects)* που αποτελούν το επόμενο ιεραρχικά επίπεδο στην επεξεργασία αιτήσεων. Για παράδειγμα ένας *Container* που αναπαριστά ένα εικονικό δικτυακό τόπο (virtual host) έχει ως παιδιά τις δικτυακές εφαρμογές (web-applications) που έχουν οριστεί γι' αυτόν το δικτυακό τόπο.
- Μια στοίβα από *Interceptor objects* που παρέχουν πολλαπλά επίπεδα επεξεργασίας για ένα συγκεκριμένο *Context* (web-application). Συμμετέχουν στην εξυπηρέτηση αιτήσεων με κλήσεις των μεθόδων *preservice()* και *postservice()* αντίστοιχα πριν και μετά την εκτέλεση της βασικής μεθόδου *service()* του *Container*.

Υπάρχουν τέσσερις εξειδικεύσεις του *Container interface*, τα *Engine*, *Host*, *Context* και *Wrapper*.

Το ***Engine interface*** αναπαριστά ολόκληρη τη μηχανή Tomcat Servlet η οποία σχετίζεται με ένα ή περισσότερα virtual *Hosts* ή με ένα ή περισσότερα *Contexts* (web applications) εφόσον υπάρχει μόνο ένα virtual *Host*.

Το ***Host interface*** αναπαριστά ένα virtual *Host* που σχετίζεται με έναν αριθμό από *Contexts*. Ο πατρικός *Container* του *Host* γενικά είναι τύπου *Engine.*

Το ***Context interface*** αναπαριστά μια δικτυακή εφαρμογή που σχετίζεται με ένα *Wrapper Container* για κάθε ορισμένο Servlet.

Το ***Wrapper interface*** αναπαριστά ένα και μόνο ορισμένο Servlet από το οποίο μπορεί να προέρχονται πολλαπλά νήματα κατά το χρόνο εκτέλεσης.

Utility Components: Κάθε *Container Component* μπορεί προαιρετικά να σχετίζεται με μοναδικό στιγμιότυπο κάποιων από τις *Utility Components, Loader, Logger, Realm, Resources*, στις οποίες μπορούμε να έχουμε πρόσβαση μέσω μεθόδων JavaBeans.

Το ***Loader interface*** αναπαριστά μια κλάση-φορτωτή Java που χρησιμοποιείται για την φόρτωση εκτελέσιμων Servlets και άλλου κώδικα που χρησιμοποιείται μέσα σε έναν *Container*. Το *interface* αυτό επίσης παρέχει υποστήριξη εντοπισμού αλλαγών στην υποκείμενη αποθήκη Servlets και το φόρτωμα και ξεφόρτωμα όλων των Java κλάσεων που φορτώνονται από αυτόν τον φορτωτή.

Το ***Logger interface*** αναπαριστά τις μεθόδους *log()* που ορίζονται σε ένα *ServletContext interface*. Οι διάφορες υλοποιήσεις μπορούν να αποθηκεύσουν έξοδο log όπου αυτό είναι επιθυμητό.

Το ***Realm interface*** αναπαριστά μια περιοχή ασφαλείας (security realm) όπου μπορεί ένας χρήστης να πιστοποιηθεί και από την οποία μπορεί να εντοπισθεί ένα σύνολο ρόλων που σχετίζονται με αυτόν.

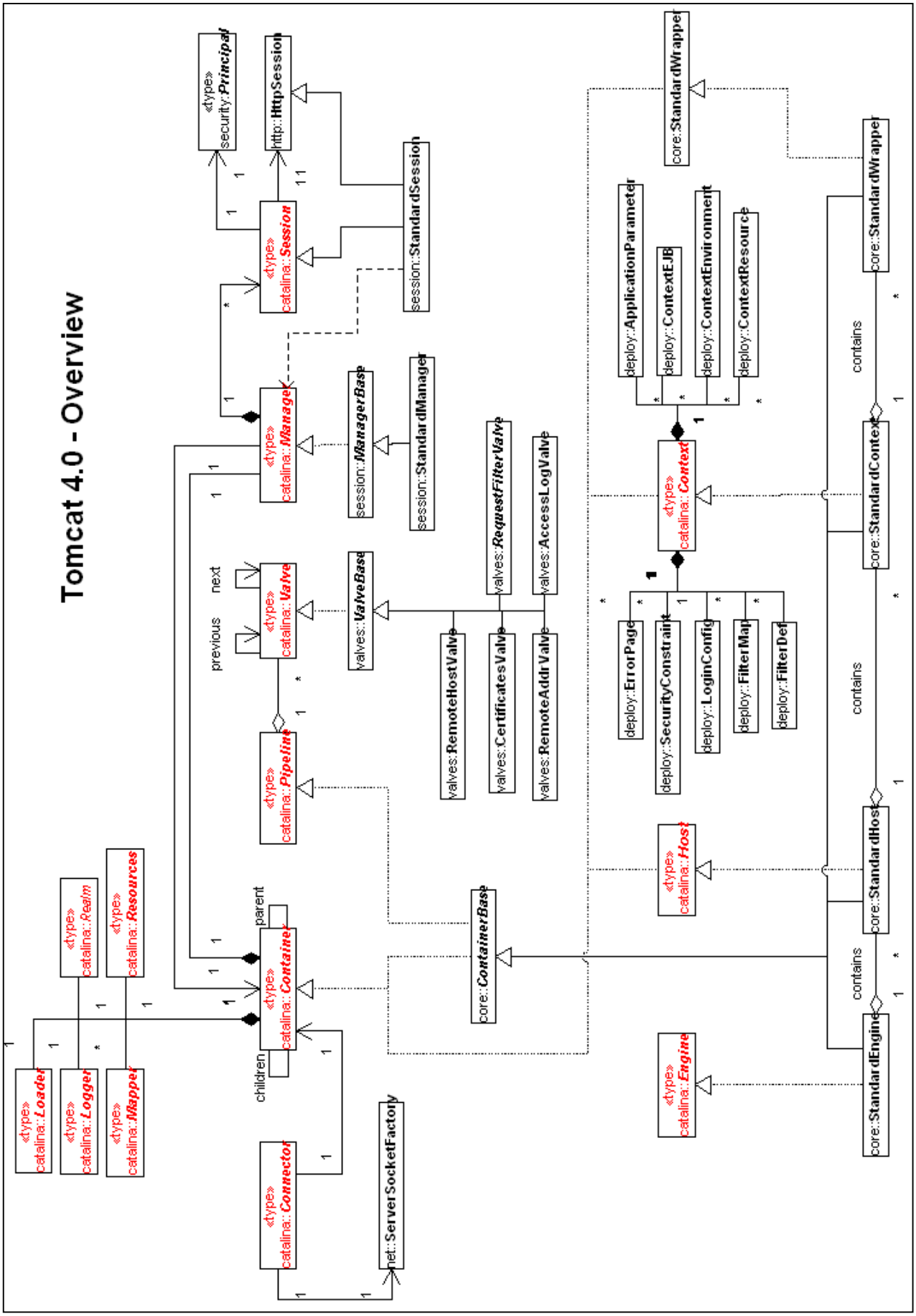
Το ***Resources interface*** επιτρέπει την πρόσβαση σε πηγές δεδομένων μιας δικτυακής εφαρμογής με χρήση των διευκολύνσεων του εξυπηρετητή του ΠΙ (π.χ. Apache) και όχι μέσω κάποιου *Container* και μεθόδων όπως οι *getResource(), getResourceAsStream()*.

Session Management Components: Η διαχείριση συνόδων (session management) παρέχεται από συνιστώσες που υλοποιούν τα παρακάτω *interfaces*.

Manager: αναπαριστά ένα διαχειριστή ενός συνόλου ενεργών S*essions* που σχετίζονται με ένα *Container*. Αυτό το *interface* επεκτείνεται για να υποστηρίξει προχωρημένα χαρακτηριστικά όπως session persistence σε επανεκκινήσεις του εξυπηρετητή, session swapping για τις αδρανείς sessions και session relocation σε ένα κατανεμημένο περιβάλλον.

Session: αναπαριστά ένα *HttpSession* που χρησιμοποιείται κατά κόρο στον Tomcat Servlet Container και αναπαριστά μια σύνοδο χρήστη και όλα τα σχετικά με αυτή αντικείμενα δεδομένων του χρήστη.

Όλες οι παραπάνω συνιστώσες συνθέτουν την αρθρωτή αρχιτεκτονική αγωγού (pipelined) του Jacarta Apache Tomcat που φαίνεται στο UML διάγραμμα του σχήματος 2.4.



Σχήμα 2.4 - UML διάγραμμα [37] Αρχιτεκτονικής του Jacarta Apache Tomcat 4.0

2.3.2 Servlets

Τα Servlets [38] είναι συνιστώσες δικτύου (web components) βασισμένες αποκλειστικά στην τεχνολογία της Java. Ως Java κλάσεις τα Servlets είναι ανεξάρτητα πλατφόρμας και μπορούν να φορτωθούν δυναμικά και να εκτελεστούν σε ένα Servlet Container που αποτελεί τμήμα ενός εξυπηρετητή εφαρμογής και συγχρόνως επέκταση ενός εξυπηρετητή του ΠΙ.

Ένας Servlet Container παρέχει τις δικτυακές εκείνες υπηρεσίες μέσω των οποίων αποστέλλονται αιτήσεις και απαντήσεις (requests, responses) μεταξύ πελατών και εξυπηρετητών σύμφωνα με το πρωτόκολλο HTTP. Επίσης περιέχει και διαχειρίζεται τα Servlets δικτυακών εφαρμογών καθόλη τη διάρκεια του κύκλου ζωής τους (lifecycle).

Όσον αφορά τη λειτουργικότητα τα Servlets τοποθετούνται κάπου ανάμεσα στα προγράμματα CGI και στις proprietary server επεκτάσεις όπως το NSAPI ή τα Apache modules. Τα Servlets έχουν τα ακόλουθα προτερήματα σχετικά με τους υπόλοιπους μηχανισμούς επεκτάσεων εξυπηρετητών:

- Είναι πολύ γρηγορότερα από τα CGI scripts λόγω του διαφορετικού μοντέλου επεξεργασίας
- Χρησιμοποιούν τυποποιημένο API που υποστηρίζεται από πολλούς εξυπηρετητές του ΠΙ
- Έχουν όλα τα προτερήματα της γλώσσας προγραμματισμού Java, συμπεριλαμβανομένης της ευκολίας ανάπτυξης και της ανεξαρτησίας πλατφόρμας
- Μπορούν να έχουν πρόσβαση σε ένα μεγάλο σύνολο από APIs που είναι διαθέσιμα για την προγραμματιστική πλατφόρμα J2EE

Το Servlet interface είναι η κεντρική αφαίρεση του Servlet API. Η συντριπτική πλειοψηφία των Servlets μιας δικτυακής εφαρμογής επεκτείνουν την κλάση *HttpServlet* άμεσα ή έμμεσα η οποία και υλοποιεί (implements) το συγκεκριμένο interface. Στο interface είναι ορισμένη μια αφηρημένη μέθοδος *service()* η οποία και υπερκαλύπτεται (overriden) από τον ορισμό των βασικών μεθόδων *doGet()* και *doPost()* από την κλάση *HttpServlet*. Οι μέθοδοι αυτές εκτελούνται για το χειρισμό HTTP GET και HTTP POST αιτήσεων αντίστοιχα.

Η δήλωση των Servlets που γίνεται στον περιγραφέα ανάπτυξης – deployment descriptor (αρχείο web.xml) του εξυπηρετητή εφαρμογής, ελέγχει το πώς ο Servlet Container δημιουργεί στιγμιότυπα ενός Servlet. Στην πιο απλή των περιπτώσεων ο Servlet Container μπορεί να δημιουργήσει μόνο ένα στιγμιότυπο ανά δήλωση Servlet. Στην περίπτωση ωστόσο που ένα Servlet υλοποιεί το *SingleThreadModel interface*, ο Servlet Container μπορεί να δημιουργήσει γι' αυτό πολλαπλά στιγμιότυπα (pool of Servlets) για το χειρισμό μεγάλο φόρτου αιτήσεων. Η χρήση του SingleThreadModel interface

εγγυάται ότι μόνο ένα νήμα εκτέλεσης (thread of execution) θα εκτελεί τη μέθοδο *service()* σε ένα στιγμιότυπο ενός συγκεκριμένου Servlet.

Κύκλος ζωής Servlet: Ο Servlet Container είναι υπεύθυνος για την φόρτωση (loading) και την υποστασιοποίηση (instantiation) των Servlets. Αυτές οι εργασίες μπορεί να γίνουν για κάθε Servlet είτε κατά την εκκίνηση του Servlet Container, είτε όταν έρθει η πρώτη αίτηση για το συγκεκριμένο Servlet. Η φόρτωση γίνεται με τη χρήση των συνηθισμένων ευκολιών φόρτωσης κλάσεων Java από τοπικά ή απομακρυσμένα συστήματα αρχείων ή και μέσω δικτυακών υπηρεσιών. Ακολουθεί η φάση της αρχικοποίησης όπου γίνεται για μια και μόνο φορά το διάβασμα μόνιμων παραμέτρων, η εγκατάσταση συνδέσεων με βάσεις δεδομένων και άλλες χρονοβόρες εργασίες. Στη συνέχεια το στιγμιότυπο Servlet που έχει φορτωθεί και αρχικοποιηθεί μπορεί να δεχτεί αιτήσεις από τον Servlet Container για εξυπηρέτηση και να παράξει απαντήσεις. Αυτό γίνεται με τη χρήση των αντικειμένων *HttpServletRequest* και *HttpServletResponse* που είναι τα αντικείμενα-παράμετροι που περνιούνται από τον Servlet container στις μεθόδους *doGet()* και *doPost()*. Όταν τέλος αποφασίζει ο Servlet Container καλεί τη μέθοδο *destroy()*, επιτρέποντας σε ένα στιγμιότυπο Servlet να ελευθερώσει τους πόρους που έχει καταλάβει και να αποθηκεύσει τις λεγόμενες persistent sessions, διότι στη συνέχεια το ίδιο θα απελευθερωθεί από τον container. Η μέθοδος *destroy()* πριν κάνει τα προαναφερθέντα περιμένει να ολοκληρωθεί η εκτέλεση νημάτων για το συγκεκριμένο στιγμιότυπο Servlet.

Θέματα multithreading: Οι Servlet Containers υλοποιούν την επεξεργασία ταυτόχρονων αιτήσεων προς το ίδιο Servlet με τη χρήση νημάτων εκτέλεσης. Για κάθε δηλωμένο Servlet στην απλούστερη περίπτωση υπάρχει μοναδικό Servlet στιγμιότυπο, την μέθοδο *service()* του οποίου μπορούν να εκτελούν ταυτόχρονα τα νήματα που αντιστοιχούν σε διαφορετικές αιτήσεις χρηστών. Το γεγονός αυτό δημιουργεί προβλήματα ασφάλειας νημάτων (thread safety), τα οποία πρέπει να λαμβάνονται σοβαρά υπόψη από τον προγραμματιστή μιας δικτυακής εφαρμογής. Το πρόβλημα εστιάζεται στην προσπέλαση για διάβασμα και γράψιμο είτε διαμοιραζόμενων πόρων, είτε ακόμη και των ίδιων των μεταβλητών ενός Servlet που «ζουν» όσο ζει το ίδιο (instance/class variables) ή ολόκληρη η δικτυακή εφαρμογή που το περιέχει (static variables). Στην περίπτωση διαμοιραζόμενων πόρων ο προγραμματιστής πρέπει είτε να προγραμματίσει το Servlet να υλοποιεί το *SingleThreadModel* (όταν ο διαμοιραζόμενος πόρος δεν προσφέρει κλείδωμα δεδομένων) είτε να υλοποιήσει τα επίμαχα σημεία με επαναληπτικές διαδικασίες μέχρι άρσης του κλειδώματος δεδομένων (όταν ο διαμοιραζόμενος πόρος προσφέρει κλείδωμα δεδομένων). Στην περίπτωση μεταβλητών ο προγραμματιστής πρέπει να αποφεύγει τη χρήση static και instance/class μεταβλητών. Σε περιπτώσεις ανάγκης μπορεί να χρησιμοποιηθεί η δεσμευμένη λέξη *synchronized* στη δήλωση τους. Επίσης λύση αποτελεί η χρήση τοπικών μεταβλητών π.χ. μέσα σε βρόγχους ή συναρτήσεις που καλούνται από τη μέθοδο *service()* κλπ., που είναι αποκλειστικά για ένα νήμα εκτέλεσης.

2.4 Επίπεδο Παρουσίασης

Αναμφισβήτητα από την αρχή εμφάνισης του Διαδικτύου ως ακόμη και σήμερα η μεταφορά πληροφοριών στο Διαδίκτυο γίνεται κατά κόρο σε σελίδες—αρχεία HTML (HyperText Markup Language). Οι σελίδες HTML αρχικά ήταν στατικές, αλλά με τον καιρό έγιναν μερικώς ή πλήρως δυναμικές για την αύξηση της λειτουργικότητάς τους και την ανταπόκρισή τους στον κατακλυσμό των πληροφοριών με περιορισμό του χρόνου συγγραφής των (δυναμική παραγωγή HTML κώδικα).

Σήμερα ολοένα εμφανίζονται νέα θέματα προς επίλυση σχετικά με την παρουσίαση πληροφοριών στον τομέα ανάπτυξης δικτυακών εφαρμογών. Αυτά τα θέματα-προβλήματα έχουν σαν σκοπό τη βελτίωση του τρόπου ανάπτυξης τέτοιων εφαρμογών, την ελαχιστοποίηση του χρόνου συντήρησης, την εξατομίκευση της παρουσιαζόμενης πληροφορίας κλπ.

Όπως αναφέρθηκε στην παράγραφο §1.3 τα σχετικά με την παρουσίαση θέματα που είχαμε να επιλύσουμε στην παρούσα εργασία ήταν η αποδέσμευση δεδομένων-παρουσίασης, η πολυγλωσσία και υποστήριξη άμεσης εναλλαγής γλώσσας, η γραφική αναπαράσταση ιεραρχιών όρων και η άμεση εναλλαγή του στιλ παρουσίασης. Για την επίλυσή τους χρησιμοποιήσαμε τις πιο σύγχρονες τεχνολογίες ανάπτυξης δικτυακών εφαρμογών που αναφέρουμε παρακάτω.

2.4.1 Η Γλώσσα XML

Η XML (eXtensible Markup Language) είναι μια επεκτάσιμη γλώσσα περιγραφής δεδομένων. Όπως και τη γλώσσα HTML είναι βασισμένη σε ετικέτες (tag-based), ωστόσο οι ετικέτες δεν είναι ορισμένες εκ των προτέρων. Ο συγγραφέας ενός XML εγγράφου ορίζει τις δικές του ετικέτες οι οποίες μπορεί να είναι είτε αυθαίρετες, είτε να ακολουθούν κάποιον ορισμό τύπου εγγράφου-DTD (Document Type Definition). Η τελευταία περίπτωση είναι η πιο ενδιαφέρουσα αφού ένα XML αρχείο που ακολουθεί ένα συγκεκριμένο DTD μοιάζει με μια βάση δεδομένων (σχήμα και δεδομένα) σε μορφή κειμένου.

Η XML είναι προσανατολισμένη στην περιγραφή δεδομένων σε αντίθεση με την HTML που χρησιμοποιείται για τη μορφοποίηση τους. Υπό αυτή την έννοια η XML δεν σχεδιάστηκε για να αντικαταστήσει την HTML αλλά ως συμπλήρωμά της. Ένας άλλος επιτυχής ορισμός για την XML είναι ότι αποτελεί ένα ανεξάρτητο πλατφόρμας, λογισμικού και υλικού εργαλείο για τη μετάδοση πληροφοριών [39].

Οι χρήσεις της XML είναι πάρα πολλές. Αποδεσμεύει τα δεδομένα από την παρουσίαση ώστε αλλαγές στα δεδομένα ή στη λογική παρουσίασης εγγράφων να μην επηρεάζουν τον τρόπο παρουσίασης και τα δεδομένα αντίστοιχα. Μειώνει την πολυπλοκότητα της ανταλλαγής δεδομένων μεταξύ συστημάτων και εφαρμογών πάνω

από το Διαδίκτυο. Διευκολύνει την ανταλλαγή οικονομικών δεδομένων μεταξύ επιχειρήσεων πάνω από το Διαδίκτυο. Επιτρέπει την αποθήκευση και ανάκτηση δεδομένων από αρχεία ή και βάσεις δεδομένων καθώς και τη μορφοποίησή τους με τη χρήση γενικών εφαρμογών. Κάνει τη διαθεσιμότητα των δεδομένων απλούστερη σε μηχανές αναζήτησης και πράκτορες. Μπορεί να ορίζει γλώσσες για διάφορα πεδία εφαρμογής όπως τα κινητά τηλέφωνα (Wireless Markup Language) η απεικόνιση διανυσματικών γραφικών (Scalable Vector Graphics) κλπ.

Στην παρούσα εργασία η XML χρησιμοποιήθηκε αποκλειστικά για τον ορισμό των διαφόρων δομών (templates) με τις οποίες θα ανακτώνται τα δεδομένα από το υποκείμενο ΣΔΒΔ και για την ανεξαρτητοποίησή τους από τη λογική παρουσίασης. Πρόκειται επίσης να χρησιμοποιηθεί για τη διευκόλυνση της εισαγωγής δεδομένων (data entry) στο σύστημα η σχεδίαση της οποίας παρουσιάζεται σε επόμενο κεφάλαιο.

2.4.2 Η Γλώσσα SVG

Η SVG (Scalable Vector Graphics) [40] είναι μια γλώσσα για την περιγραφή διδιάστατων διανυσματικών γραφικών με τη χρήση XML και ταυτόχρονα ένας μορφότυπος αρχείων γραφικών (graphics file format). Η SVG επιτρέπει τη δημιουργία υψηλής ποιότητας δυναμικά παραγόμενων γραφικών από δεδομένα πραγματικού χρόνου [41]. Με την ισχυρή αυτή τεχνολογία έχει ήδη αρχίσει να δημιουργείται μια νέα γενιά δικτυακών εφαρμογών βασισμένων σε οδηγούμενα από δεδομένα, αλληλεπιδραστικά, εξατομικευμένα γραφικά. Η εμφάνιση τους μέσα σε σελίδες γίνεται με τη χρήση ειδικού "plug-in" λογισμικού [42] που επεκτείνει τους φυλλομετρητές του ΠΙ.

Στην παρούσα εργασία η SVG χρησιμοποιήθηκε για την παρουσίαση ιεραρχιών όρων του μοντελοποιούμενου πεδίου εφαρμογής για τη διευκόλυνση της πλοήγησης στο υλοποιημένο ΠΣΒΠΙ.

2.4.3 Η Γλώσσα XSL

Το χαρακτηριστικό που προσφέρει η XML στον καθένα να ορίζει όποια ετικέτα (tag) επιθυμεί, καθιστά αδύνατο στους φυλλομετρητές να μορφοποιήσουν ένα XML έγγραφο αφού δεν γνωρίζουν τη σημασία των διαφόρων ετικετών. Η XSL (eXtensible Stylesheet Language) είναι μια πρότυπη γλώσσα, τυποποιημένη από το World Wide Web Consortium (W3C), για τον καθορισμό της μορφής παρουσίασης XML εγγράφων [43]. Αποτελείται από 3 μέρη :

- XSLT, γλώσσα για το μετασχηματισμό XML εγγράφων σε άλλα XML έγγραφα ή έγγραφα άλλου τύπου που μπορούν να παρουσιασθούν από ένα φυλλομετρητή (όπως XHTML έγγραφα [44]). Με την XSLT μπορούν να προστεθούν στοιχεία (elements) στο τελικό αρχείο εξόδου ή να αφαιρεθούν

στοιχεία από το πηγαίο XML έγγραφο, να αναδιαταχθούν ή να ταξινομηθούν στοιχεία και να παρθούν αποφάσεις για το ποια στοιχεία θα εμφανιστούν και ποια όχι.

- XPath, γλώσσα για τον καθορισμό τμημάτων σε ένα XML έγγραφο
- XSL Formatting Objects, ένα λεξικό για τη μορφοποίηση XML εγγράφων βάσει π.χ. των τιμών των XML δεδομένων ή βάσει της συσκευής εξόδου που προορίζονται όπως οθόνη, εκτυπωτής, ηχεία κ.α.

Στη διαδικασία μετασχηματισμού η γλώσσα XSLT χρησιμοποιεί την XPath για να εντοπίσει τμήματα του XML εγγράφου που ταιριάζουν με ένα ή περισσότερα προκαθορισμένα templates. Όταν εντοπισθούν τέτοια η XSLT τα μετασχηματίζει στα αντίστοιχα τμήματα του εγγράφου-αποτελέσματος. Τμήματα που δεν ταιριάζουν με κανένα template καταλήγουν ως έχουν στο έγγραφο-αποτέλεσμα.

Στην παρούσα εργασία η XSL χρησιμοποιήθηκε για την παρουσίαση ενός συνόλου από templates δεδομένων που ανακτώνται από το SIS και αποστέλλονται στον εξυπηρετητή του ΠΙ ως XML streams. Η αποδέσμευση δεδομένων-παρουσίασης που προσφέρει ο συνδυασμός XML-XSL αποδείχθηκε πολύ λειτουργική για την εμφάνιση των ανακτόμενων δεδομένων με διάφορους τρόπους και σε διάφορες γλώσσες.

Ο μοναδικός φυλλομετρητής που υποστηρίζει 100% την επίσημη πρόταση του W3C για την XSL είναι ο Microsoft Internet Explorer 6.0, γι' αυτό και είναι απαραίτητο να είναι εγκατεστημένος στο μηχάνημα κάθε πελάτη (client) του παρόντος ΠΣΒΠΙ.

2.4.4 DHTML - Javascript

Η Dynamic HTML [45] είναι ένας όρος που χρησιμοποιείται για να περιγράψει το συνδυασμό των τεχνολογιών HTML, CSS (Cascading Style Sheet) και scripts (Javascript ή Vbscript) που «δίνουν ζωή» σε HTML έγγραφα. Στο παρόν ΠΣΒΠΙ οι τεχνολογίες αυτές ενσωματώνονται με τη χρήση του εργαλείου **mapbMenu** που έχει αναπτυχθεί με αυτές και το οποίο παρουσιάζουμε αμέσως.

2.4.4.1 Το εργαλείο εμφάνισης μενού mapbMenu

Το εργαλείο mapbMenu [46] είναι ένα πλήρως προσαρμοζόμενο και παραμετρικό εργαλείο κατασκευής μενού για δικτυακές εφαρμογές. Επιτρέπει την δημιουργία ιεραρχίας μενού-υπομενού οποιουδήποτε βάθους και την τοποθέτησή του συνολικού μενού σε κατακόρυφη ή οριζόντια διάταξη σε οποιοδήποτε σημείο της επιφάνειας του εκάστοτε φυλλομετρητή. Επίσης επιτρέπει παραμετρικό ορισμό γραμματοσειρών, χρωμάτων (foreground και background) για τους τίτλους των μενού, πλάτους των μενού κλπ. Για την κατασκευή του μενού μιας δικτυακής εφαρμογής απαιτούνται 3 αρχεία, το βασικό αρχείο **mapb_menu.js**, όπου υπάρχει το script για την εμφάνιση του μενού, το αρχείο **menu.js** όπου υπάρχει ο ορισμός του μενού και το αρχείο **menu.css** για την

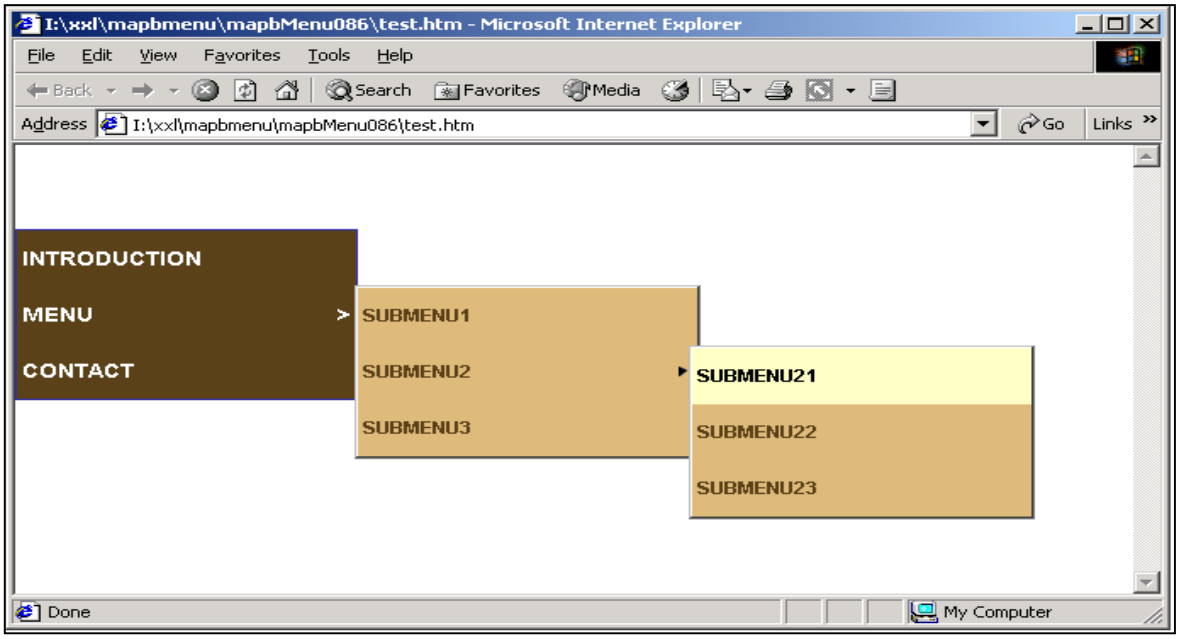
μορφοποίηση των τίτλων των μενού με παραμετρικό ορισμό γραμματοσειράς, μεγέθους χαρακτήρων, πάχους πλαισίων κλπ.

Το εργαλείο αυτό χρησιμοποιήθηκε στο επίπεδο παρουσίασης του παρόντος ΠΣΒΠΙ για την παροχή ευκολιών στην προσθήκη νέων λειτουργιών ανάκτησης δεδομένων από το υποκείμενο ΣΔΒΔ. Οι επιλογές αυτές ορίζονται στο αρχείο **menu.js** με πολύ απλό τρόπο.

Για παράδειγμα απαιτείται ο παρακάτω κώδικας από πλευράς προγραμματιστή :

```
mainMenu = new Menu ('mainMenu',0,60);
mainMenu.m_vertical = true;            // If true menu is vertical
mainMenu.sm_width = 200;               // Width of submenu items
mainMenu.sm_height = 40;               // Height of submenu items
mainMenu.m_color  = '#5A4118';         // Color of initial menu
mainMenu.m_a_color  = '#FFFFC6';       //Initial Menu when mouseOver
mainMenu.addItem ('INTRODUCTION','URL_INTRO',false);
mainMenu.addItem ('MENU','',false);
   mainMenu.addItem('SUBMENU1', 'URL1', false);
   mainMenu.addItem('SUBMENU2', '', false);
         mainMenu.addItem('SUBMENU21', 'URL21', false);
         mainMenu.addItem('SUBMENU22', 'URL22', false);
         mainMenu.addItem('SUBMENU23', ' URL23', true);
   mainMenu.addItem('SUBMENU3', ' URL3', true);
mainMenu.addItem ('CONTACT','URL_CONTACT',true);
mainMenu.doMenu();
```

για να παραχθεί το μενού του σχήματος 2.5:



Σχήμα 2.5 - Παράδειγμα μενού υλοποιημένου με το εργαλείο mapbMenu

2.5 Ολοκλήρωση Τεχνολογιών – Αρχιτεκτονική Συστήματος

Η συνολική προτεινόμενη αρχιτεκτονική για το παρόν ΠΣΒΠΙ εικονίζεται στο σχήμα 2.6. Την κεντρική θέση του συστήματος καταλαμβάνουν ένας εξυπηρετητής ΠΙ Apache και δύο ή περισσότεροι εξυπηρετητές εφαρμογής Apache Tomcat 4.0 (όλοι ιδανικά βρίσκονται σε διαφορετικά μηχανήματα ενός τοπικού δικτύου). Οι λειτουργίες ανάκτησης δεδομένων έχουν ήδη υλοποιηθεί με χρήση της τεχνολογίας των Java Servlets. Όλα τα Servlets μαζί αποτελούν τη δικτυακή εφαρμογή του ΠΣΒΠΙ η οποία πρέπει να είναι εγκατεστημένη σε κάθε μηχάνημα όπου υπάρχει εξυπηρετητής εφαρμογής Tomcat. Κάθε Servlet αντιστοιχεί σε μία ή περισσότερες λειτουργίες του ΠΣΒΠΙ και συντίθεται από δυο είδη συναρτήσεων, τις συναρτήσεις επερωτήσεων προς το χρησιμοποιούμενο ΣΔΒΔ και τις συναρτήσεις παραγωγής XML. Τα XML δεδομένα που παράγονται από τα διάφορα Servlets συμμορφώνονται με αντίστοιχα DTD, που βρίσκονται σε συγκεκριμένο φάκελο της δενδρικής δομής της δικτυακής εφαρμογής. Αυτό συμβαίνει για να υπάρχει κάποιος τυποποιημένος τρόπος επικοινωνίας μεταξύ των προγραμματιστών που ασχολούνται με τη λογική επεξεργασίας του ΠΣΒΠΙ και μεταξύ αυτών που ασχολούνται με το επίπεδο παρουσίασης. Το επίπεδο παρουσίασης κατά κύριο λόγο αποτελείται από XSL αρχεία. Σε κάθε Servlet του ΠΣΒΠΙ αντιστοιχεί ένα[2] ανεξάρτητο αρχείο XSL που μορφοποιεί τα XML δεδομένα που εκείνο παράγει (βασισμένο στο αντίστοιχο DTD).

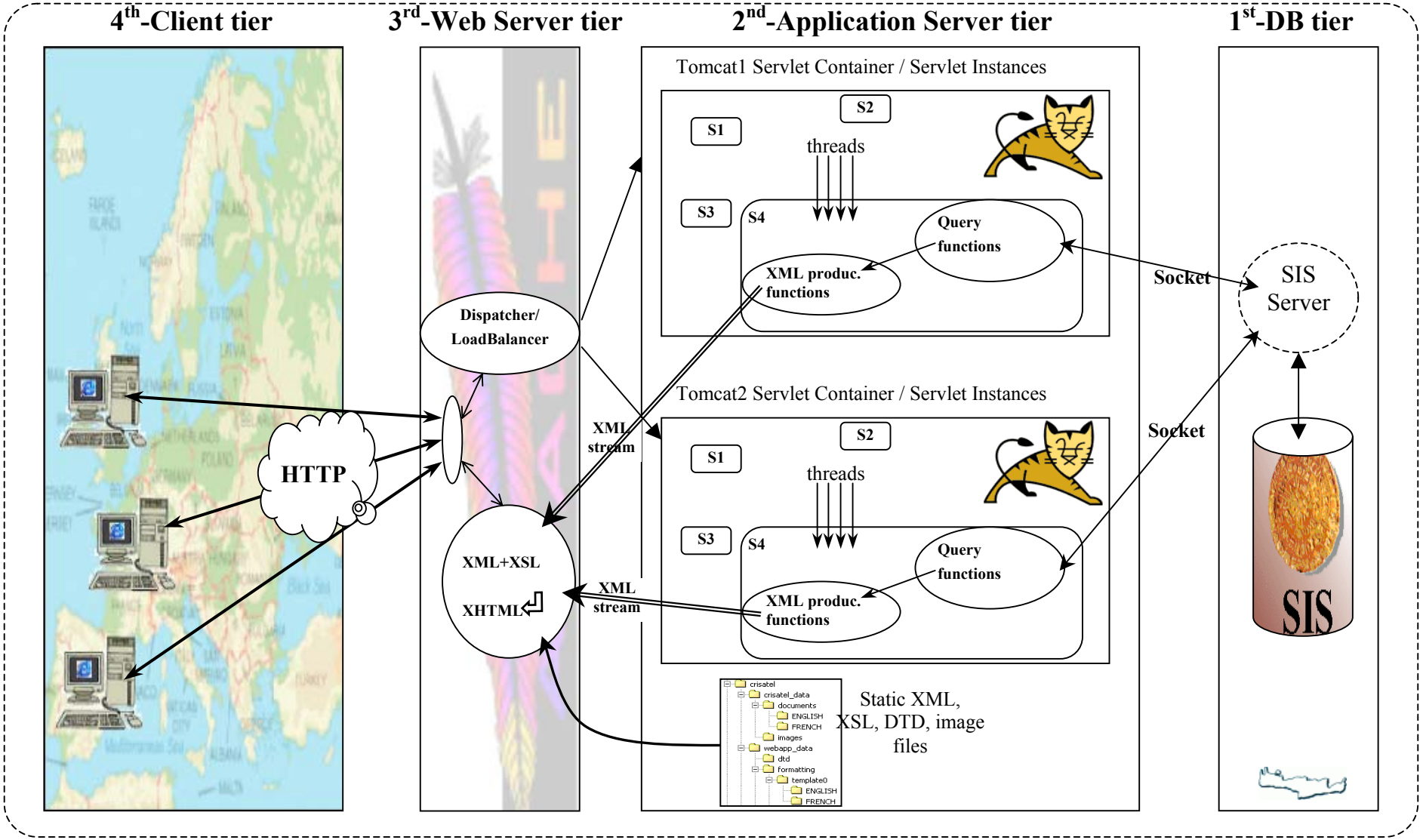
Λειτουργία αρχιτεκτονικής:

Πριν το όλο ΠΣΒΠΙ αρχίσει να λειτουργεί πρέπει να οριστούν σε κατάλληλα αρχεία του εξυπηρετητή του ΠΙ Apache οι «εργάτες»-workers Tomcat (μηχάνημα, διεύθυνση IP, port, παράγοντας φόρτου), που θα εξυπηρετούν τις αιτήσεις για Java Servlets. Αντίστοιχες ρυθμίσεις πρέπει να γίνουν και σε αρχεία των Tomcat εξυπηρετητών (βλ. [47]). Αμέσως μετά μπορεί να γίνει η εκκίνηση πρώτα όλων των Tomcat εξυπηρετητών και στη συνέχεια του εξυπηρετητή Apache.

Οι αιτήσεις των πελατών, που μπορεί να βρίσκονται οπουδήποτε (αρκεί να έχουν πρόσβαση στο Διαδίκτυο), λαμβάνονται από τον εξυπηρετητή του ΠΙ Apache μέσω του πρωτοκόλλου HTTP. Εφόσον η αίτηση ανήκει σε μια νέα σύνοδο (session) χρήστη ο Apache εξετάζει τη διαθεσιμότητα και το φόρτο των εξυπηρετητών Tomcat και η αίτηση διαβιβάζεται στον καταλληλότερο απ' αυτούς. Αν η αποστελλόμενη αίτηση ανήκει σε σύνοδο που είναι ήδη ενεργή σε κάποιον εξυπηρετητή Tomcat, τότε ο Apache την διαβιβάζει σ' αυτόν ("load balancing with sticky sessions").

Η αίτηση που παραλαμβάνεται από κάποιον από τους εξυπηρετητές Tomcat του συστήματος μετατρέπεται ακόλουθα σε αντικείμενο τύπου *HttpServletRequest* το οποίο και διαχειρίζεται ο Servlet Container.

[2] για την ακρίβεια 1 XSL / (παρουσίαση & γλώσσα), εξηγούμε σε επόμενο κεφάλαιο



Σχήμα 2.6 - Η 4-επίπεδη αρχιτεκτονική του συστήματος

Αν η αίτηση αντιστοιχεί σε Servlet που έχει ήδη φορτωθεί στον *Servlet container* (από προηγούμενη αίτηση) τότε απλά δημιουργείται ένα νέο νήμα εκτέλεσης (thread) το οποίο εκτελεί τη συνάρτηση *doGet()* του Servlet στιγμιότυπου. Η συνάρτηση αυτή για όλα τα Servlets του παρόντος ΠΣΒΠΙ κάνει διαδοχικά τις εξής λειτουργίες:

1. Διαβάζει τις παραμέτρους της δικτυακής εφαρμογής εφόσον δεν έχουν διαβαστεί από κανένα άλλο Servlet αυτής και τις φορτώνει σε στατικό αντικείμενο που ζει όσο ζει και η εφαρμογή.
2. Εγκαθιστά σύνδεση με τη βάση του SIS εφόσον δεν έχει εγκατασταθεί σύνδεση για τη σύνοδο που ανήκει η συγκεκριμένη αίτηση.
3. Καλεί διαδοχικά συναρτήσεις που περιέχουν επερωτήσεις του SIS (Query functions), οι οποίες αποστέλλονται στον εξυπηρετητή αυτού (SISServer) με τη χρήση sockets, μέσω των οποίων λαμβάνονται και τα αποτελέσματα. Οι συναρτήσεις αυτές γεμίζουν κατάλληλες δομές δεδομένων (πίνακες από *Vectors*, *Hashtables* κλπ) ορισμένες στη συνάρτηση *doGet()*.
4. Οι δομές δεδομένων περνιούνται σαν παράμετροι σε συναρτήσεις παραγωγής ρεύματος XML προς τον εξυπηρετητή του ΠΙ.

Αφού φτάσει το ρεύμα XML στον εξυπηρετητή του ΠΙ ο τελευταίος εντοπίζει τους απαιτούμενους στατικούς πόρους (XSL, DTD, εικόνες) και τους ζητάει από το μηχάνημα του εξυπηρετητή εφαρμογής που εξυπηρέτησε την αίτηση. Όλα τα παραπάνω δεδομένα αποστέλλονται στον πελάτη-φυλλομετρητή βάσει του πρωτοκόλλου HTTP και εκείνος αναλαμβάνει τη μορφοποίηση και την εμφάνισή τους στο χρήστη.

Κεφάλαιο 3

Σημασιολογικό μοντέλο δεδομένων

Το κεφάλαιο αυτό πραγματεύεται την κατασκευή του σημασιολογικού μοντέλου δεδομένων για το ΠΣΒΠΙ της παρούσας εργασίας. Ο σχεδιασμός του έγινε με γνώμονα την προσπάθεια ανάδειξης των σχέσεων που διέπουν τα έργα τέχνης και τις έννοιες συντήρησής τους, καθώς και των γνώσεων που απορρέουν από αυτές τις σχέσεις. Ο στόχος που τέθηκε εξ' αρχής, για την ανάπτυξη ενός εκπαιδευτικού ηλεκτρονικού εγχειριδίου για συντηρητές έργων τέχνης, προσεγγίζεται αρκετά καλά με την συγκεκριμένη μοντελοποίηση.

3.1 Τα τμήματα του μοντέλου

Το σχεδιασμένο μοντέλο αποτελείται από τρία βασικά τμήματα, το τμήμα θησαυρού, το τμήμα του πεδίου συντήρησης/αποκατάστασης έργων τέχνης και το τμήμα της γνώσης του συγκεκριμένου πεδίου. Τα τμήματα αυτά συνδέονται μεταξύ τους με διαφόρων ειδών συσχετίσεις μεταξύ κλάσεών τους. Η ολοκλήρωση των τμημάτων είναι ουσιαστική στην προσπάθεια να αποδοθεί εκπαιδευτικός χαρακτήρας στο παρόν ΠΣΒΠΙ.

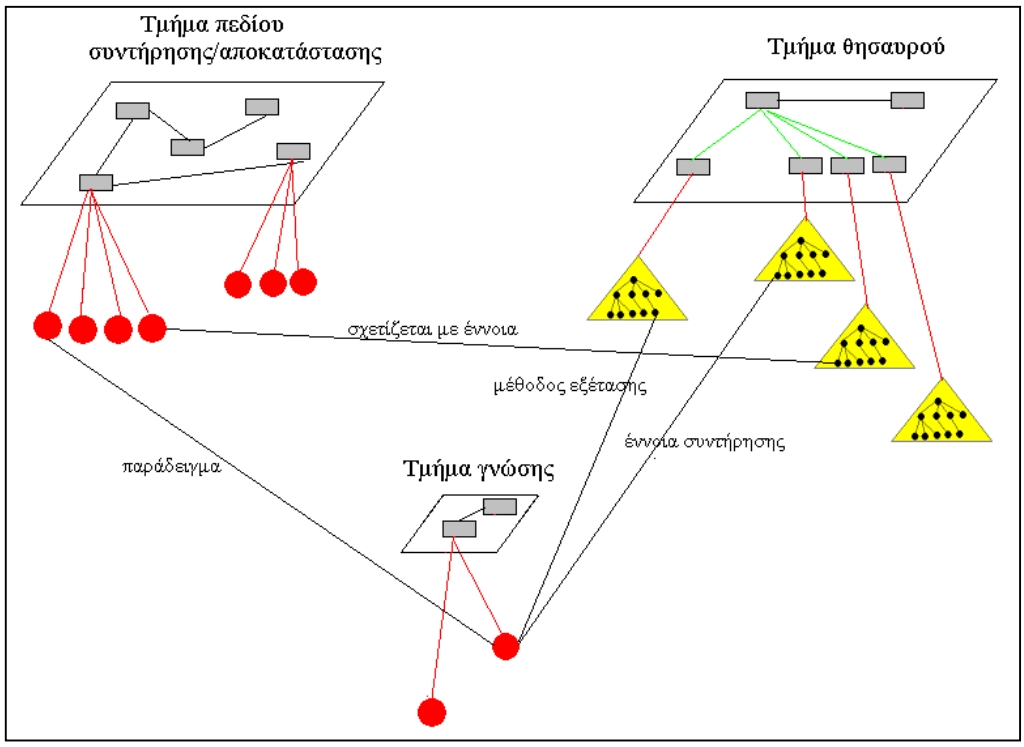
Το τμήμα του θησαυρού είναι απαραίτητο στην προσπάθεια συμφωνίας μεταξύ των συντηρητών παγκοσμίως για την χρήση συγκεκριμένων όρων που θα αποδίδουν δεδομένες έννοιες. Επίσης η μετάφραση όρων σε πολλές διαφορετικές γλώσσες είναι πολύ σημαντική για τη συνεννόηση συντηρητών διαφορετικών εθνικοτήτων για τη συντήρηση έργων τέχνης που βρίσκονται σε κατάσταση δανεισμού ή μετακίνησης.

Το τμήμα του πεδίου συντήρησης/αποκατάστασης είναι ο χώρος των «απτών» παραδειγμάτων που θα βοηθήσουν τον νέο συντηρητή να κατανοήσει με τη βοήθεια πολυφασματικών εικόνων πώς εμφανίζονται οι διάφορες διαδικασίες, φαινόμενα και έννοιες σε συγκεκριμένα έργα.

Το τμήμα της γνώσης του πεδίου είναι εκείνο που προσπαθεί να παράσχει υπάρχουσα γνώση στους εκπαιδευόμενους συντηρητές. Η γνώση δεν είναι τίποτα άλλο από μια συσχέτιση μεθόδων εξέτασης πολυφασματικών εικόνων με έννοιες του πεδίου συντήρησης/αποκατάστασης (τεχνικές κατασκευής, υλικά κατασκευής, τεχνικές αποκατάστασης, φαινόμενα αλλοιώσεων). Η γνώση μπορεί επίσης να συσχετιστεί και με

τα πιο χαρακτηριστικά από τα παραδείγματα του πεδίου που αναδεικνύουν τη συσχέτιση που η ίδια εκφράζει.

Η συσχέτιση των τμημάτων του μοντέλου φαίνονται στο σχήμα 3.1.



Σχήμα 3.1 - Τα τμήματα του μοντέλου

Οι συσχετίσεις μεταξύ των τμημάτων του μοντέλου ισχύουν πρωταρχικά για το επίπεδο των κλάσεων πράγμα που δε φαίνεται στο συγκεκριμένο σχήμα (φαίνονται μόνο οι σχέσεις στο Token level) για να μην είναι πολύπλοκο.

3.2 Μοντελοποίηση θησαυρού

Για την μοντελοποίηση του θησαυρού του πεδίου χρησιμοποιείται ένα πολύ απλοποιημένο μοντέλο σε σύγκριση με το μοντέλο του TMS [48]. Ο λόγος είναι ότι στην προκείμενη περίπτωση χρειάζονται μόνο οι βασικότερες από τις συσχετίσεις (*BT*, *RT*, *UF*) μεταξύ όρων για να επιτύχουμε το επιθυμητό αποτέλεσμα, απλοποιώντας παράλληλα τη σύνταξη των επερωτήσεων ανάκτησης δεδομένων[3]. Επίσης ακολουθείται διαφορετική φιλοσοφία όσον αφορά την πολυγλωσσία, όπου υπάρχει μια βασική γλώσσα και μόνο οι όροι που είναι καταχωρισμένοι υπό αυτήν συνδέονται με τα στιγμιότυπα άλλων κλάσεων του μοντέλου (έργα τέχνης, πολυφασματικές εικόνες κλπ). Οι όροι-μεταφράσεις παρόλο

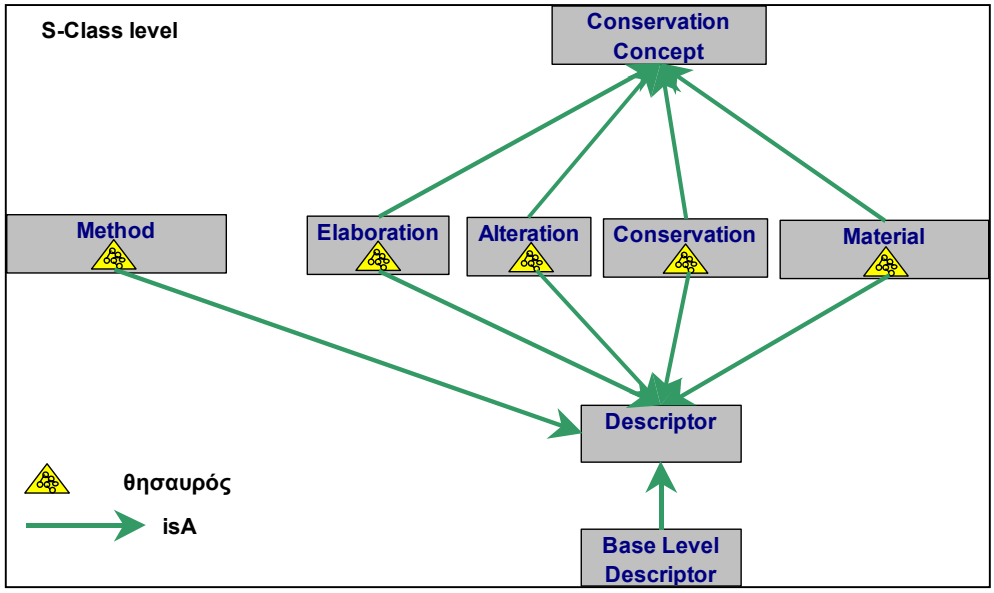
[3] Αυτό δυσκολεύει ωστόσο την εισαγωγή δεδομένων όπως αναφέρεται σε επόμενο κεφάλαιο

που είναι και οι ίδιοι άτομα (tokens), ωστόσο δε συνδέονται παρά μόνο με τους όρους τις βασικής γλώσσας.

3.2.1 Εντοπισμός βασικών κλάσεων-συσχετίσεων

Οι βασικές διαστάσεις του προβλήματος της συντήρησης/αποκατάστασης έργων τέχνης είναι τέσσερις : οι τεχνικές κατασκευής (elaboration techniques), τα φαινόμενα αλλοιώσεων (alterations), οι διαδικασίες συντήρησης (conservation processes) και τέλος τα υλικά κατασκευής (materials), που επηρεάζουν σημαντικά τις προηγούμενες τρεις. Με τη χρήση μεθόδων εξέτασης (examination methods) με πολυφασματικές εικόνες μπορούν να εντοπισθούν διάφορες εκφάνσεις των παραπάνω διαστάσεων πάνω σε ένα συγκεκριμένο έργο.

Σε κάθε μια από τις παραπάνω έννοιες περιλαμβάνεται ένα σύνολο όρων που μπορούν να οργανωθούν σε ιεραρχία γενικών-ειδικών όρων. Για όλους τους όρους στη βασική γλώσσα (εν προκειμένω Αγγλική) που θα περιλαμβάνονται στο σύστημα ορίζουμε την κλάση Descriptor. Ως εξειδικεύσεις αυτής ορίζουμε τις κλάσεις Elaboration, Alteration, Conservation, Material, Method. Οι τέσσερις πρώτες είναι έννοιες συντήρησης και ως εκ τούτου τις ορίζουμε ως εξειδικεύσεις της κλάσης Conservation Concept (σχήμα 3.2).



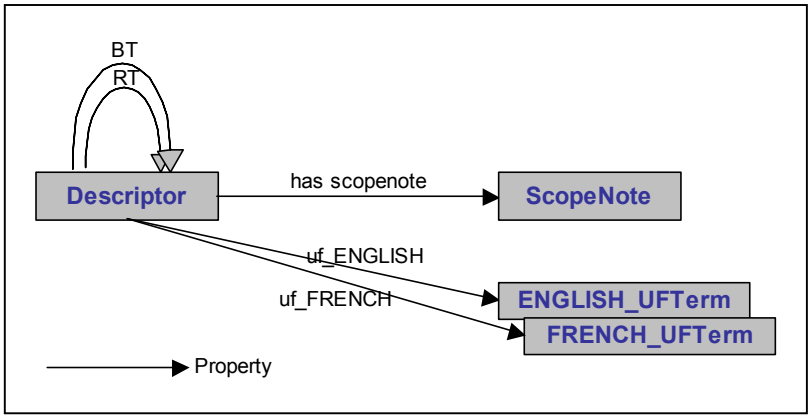
Σχήμα 3.2 - Οι βασικές κλάσεις του θησαυρού σε ιεραρχία γενίκευσης-εξειδίκευσης

Στο σχήμα 3.2 φαίνεται επιπλέον η κλάση Base Level Descriptor. Η χρησιμότητά της είναι να διαμερίσει τις ιεραρχίες όρων «κρατώντας» όρους-κλειδιά, γεγονός που θα βοηθήσει στην ευκολότερη επισκόπηση των περιεχομένων της βάσης που σχετίζονται με τους όρους των ιεραρχιών. Για παράδειγμα ένας τέτοιος όρος κλειδί για την ιεραρχία των υλικών θα μπορούσε να είναι ο όρος wooden materials, που σ' αυτήν την περίπτωση θα πρέπει να τοποθετηθεί ως στιγμιότυπο της κλάσης Base Level Descriptor.

Για κάθε όρο του θησαυρού απαιτείται η ύπαρξη επεξηγηματικού κειμένου το οποίο θα διασαφηνίζει επακριβώς την έννοια που περιγράφει ο όρος. Η κλάση ScopeNote έχει εισαχθεί για το σκοπό αυτό και με αυτήν συνδέεται η κλάση Descriptor με τη συσχέτιση *has_scopenote* . Στην κλάση ScopeNote ταξινομούνται επεξηγηματικά κείμενα γραμμένα στη βασική γλώσσα.

Η δόμηση των ιεραρχιών όρων στηρίζεται στις ανακλαστικές ιδιότητες *BT*, *RT* οι οποίες υφίστανται μόνο μεταξύ Descriptors, δηλαδή όρων στη βασική γλώσσας.

Στο σύστημα μπορούν να υπάρχουν και μη καθιερωμένοι όροι - UsedFor terms - που χρησιμοποιούνται κάποιες φορές στη θέση των καθιερωμένων με τους οποίους πρέπει προφανώς να σχετίζονται. Μη καθιερωμένοι όροι για έναν όρο X της βασικής γλώσσας μπορούν να υπάρχουν σε διάφορες γλώσσες. Ωστόσο είναι σημαντικό να ειπωθεί ότι οι μη καθιερωμένοι όροι δεν μεταφράζονται. Άρα για ένα μη καθιερωμένο όρο EN_uf_X1 δεν συνεπάγεται ότι υπάρχει η μετάφρασή του FR_uf_X1. Για τους μη καθιερωμένους όρους ορίζουμε κλάσεις ανά γλώσσα όπως ENGLISH_UFTerm, FRENCH_UFTerm και συσχετίσεις *uf_ENGLISH*, *uf_FRENCH* της κλάσης Descriptor με αυτές.



Σχήμα 3.3 - Δόμηση όρων θησαυρού

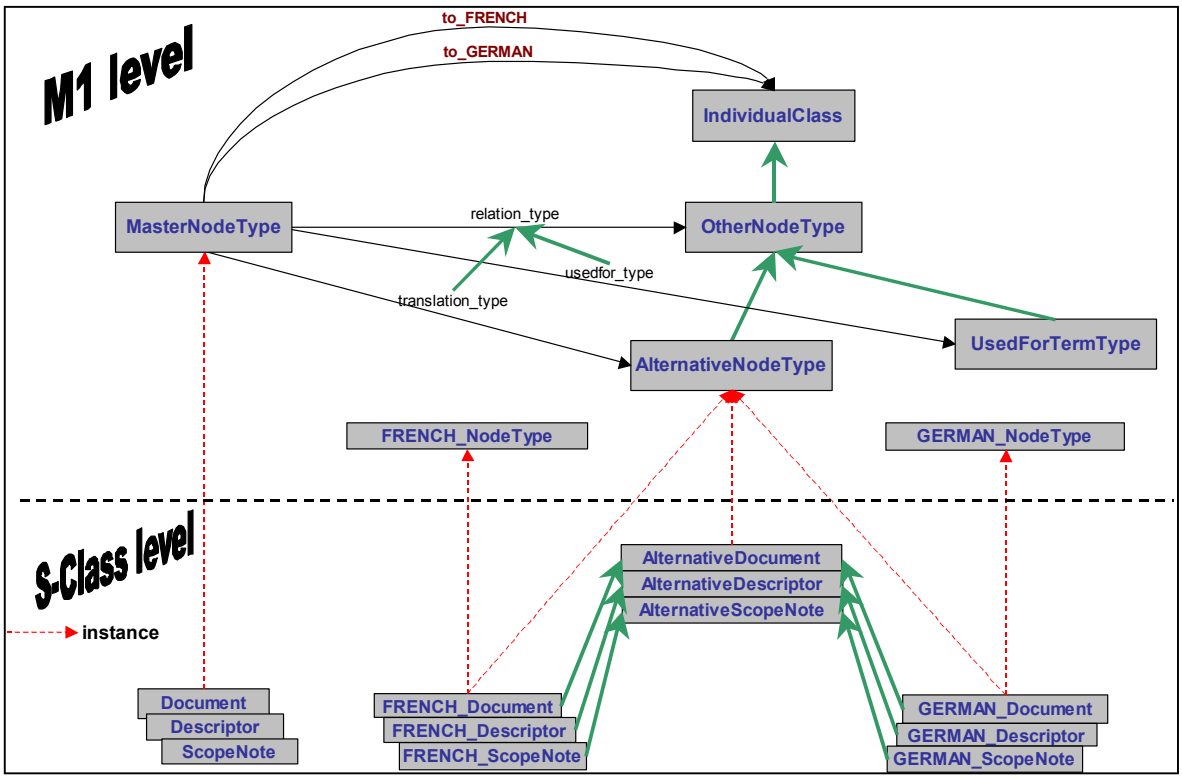
3.2.2 Πολυγλωσσία

Η απαίτηση για πολυγλωσσία στο παρόν ΠΣΒΠΙ εντοπίζεται στις κλάσεις εκείνες που είτε το όνομα είτε οι primitive τιμές ιδιοτήτων των στιγμιοτύπων τους είναι αντικείμενο μετάφρασης. Όλες αυτές οι κλάσεις ταξινομούνται κάτω από τη μετα-κλάση MasterNodeType από την οποία ξεκινάνε οι μετα-ιδιότητες που επιτελούν τη μεταγωγή στις διάφορες γλώσσες όπως *to_FRENCH*, *to_GERMAN* κλπ. Οι μετα-ιδιότητες καταλήγουν στην primitive κλάση IndividualClass για να μπορεί η μεταγωγή να καταλήγει είτε σε άτομα (tokens) είτε σε primitive τιμές.

Βασικές κλάσεις που υπόκεινται μεταγωγή γλώσσας είναι οι Descriptor, ScopeNote και Document. Η κλάση Document χρησιμοποιείται για την ταξινόμηση εγγράφων που αντιπροσωπεύουν τόσο δελτία συντήρησης έργων τέχνης, όσο και κείμενα περιγραφής

γνώσεων του πεδίου (συγκεκριμένες διαπιστώσεις βασισμένες σε συγκεκριμένες μεθόδους εξέτασης).

Για κάθε κλάση που υπόκειται σε μετάφραση υπάρχουν αντίστοιχες κλάσεις για κάθε γλώσσα πλην της βασικής. Έτσι για την κλάση Descriptor υπάρχουν οι κλάσεις FRENCH_Descriptor, GERMAN_Descriptor, τις οποίες ορίζουμε ως εξειδικεύσεις της κλάσης AlternativeDescriptor (σχήμα 3.4).



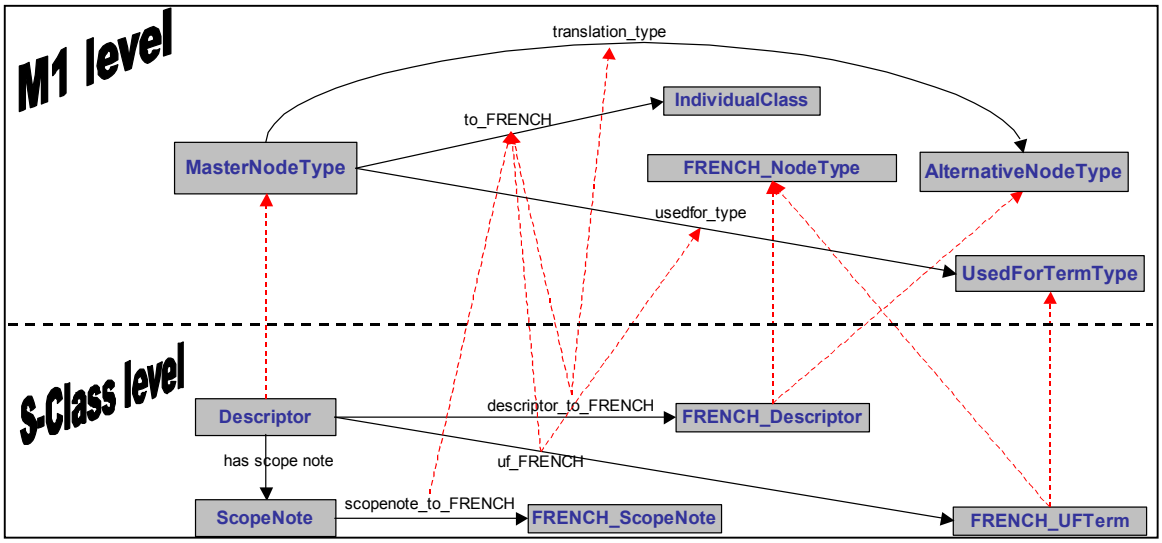
Σχήμα 3.4 - Μοντελοποίηση εναλλαγής γλώσσας

Όλες οι κλάσεις που περιέχουν στιγμιότυπα σε συγκεκριμένη γλώσσα εκτός της βασικής ταξινομούνται σε αντίστοιχη μετα-κλάση. Έτσι οι κλάσεις FRENCH_Descriptor, FRENCH_ScopeNote, FRENCH_Document ταξινομούνται κάτω από την μετα-κλάση FRENCH_NodeType. Επίσης ταξινομούνται κάτω από την μετα-κλάση AlternativeNodeType, πράγμα που δείχνει ότι τα στιγμιότυπά τους είναι μεταφράσεις αντίστοιχων στιγμιοτύπων της βασικής γλώσσας.

Η μεταγωγή που επιτυγχάνεται στο μετα-επίπεδο από τη βασική γλώσσα σε κάθε άλλη, μέσω των μετα-ιδιοτήτων *to_LANG* [4] (*to_FRENCH* κλπ), δεν διακρίνει την περίπτωση μεταγωγής μέσω μετάφρασης, από την περίπτωση μεταγωγής μέσω χρήσης μη καθιερωμένων αλλόγλωσσων όρων. Γι' αυτό στο μετα-επίπεδο έχουν εισαχθεί οι μετα-ιδιότητες *translation_type*, *usedfor_type*, οι οποίες συνδέουν την μετα-κλάση MasterNodeType με τις μετα-κλάσεις AlternativeNodeType, UsedForTermType αντίστοιχα. Η ιδιότητα *descriptor_to_FRENCH* που συνδέει ένα καθιερωμένο όρο

[4] Στο εξής όπου χρησιμοποιείται LANG θα μπορεί να λάβει τιμές FRENCH, GERMAN κλπ.

Descriptor (βασική γλώσσα) με ένα FRENCH_Descriptor (στο επίπεδο S-Class) πρέπει να ταξινομείται κάτω από τις δυο μετα-ιδιότητες *translation_type*, *to_FRENCH*.



Σχήμα 3.5 - Υποστασιοποίηση ιδιοτήτων για την εναλλαγή γλώσσας

Παρόμοια η ιδιότητα *uf_FRENCH* που συνδέει ένα καθιερωμένο όρο Descriptor (βασική γλώσσα) με ένα FRENCH_UFTerm (στο επίπεδο S-Class) πρέπει να ταξινομείται κάτω από τις δυο μετα-ιδιότητες *usedfor_type*, *to_FRENCH*.

Η πολυγλωσσία στο παρόν σύστημα εμφανίζεται και στην περίπτωση ιδιοτήτων με primitive τιμές. Για παράδειγμα η κλάση Assessment, που θα αναφέρουμε στο επόμενο κεφάλαιο έχει μια ιδιότητα *plainText_ENGLISH* που η τιμή της είναι ένα string. Η μετάφραση σε αυτήν την περίπτωση επιτυγχάνεται με τη δημιουργία νέας ιδιότητας *plainText_FRENCH* η οποία ταξινομείται κάτω από την μετα-ιδιότητα *to_FRENCH*. Για τον συγκεκριμένο αυτό λόγο οι κλάσεις Man-Made Object, Assessment και Actor που αναφέρονται στις επόμενες παραγράφους ταξινομούνται κάτω από τη μετα-κλάση MasterNodeType.

3.3 Μοντελοποίηση πεδίου εφαρμογής

Για τη μοντελοποίηση του πεδίου εφαρμογής στο συγκεκριμένο ΠΣΒΠΙ αρκεί να εντοπισθούν τα βασικά σημεία που θα μπορούσαν να είναι χρήσιμα σε ένα εκπαιδευόμενο συντηρητή.

Κεντρικό ρόλο στη δουλειά ενός συντηρητή κατέχει το λεγόμενο δελτίο συντήρησης. Στο δελτίο αυτό περιέχονται γενικές πληροφορίες για ένα έργο τέχνης καθώς και επίσημες (official) εκτιμήσεις ειδικών για ολόκληρη την πορεία συντήρησής του από της κατασκευής του και στο εξής. Οι εκτιμήσεις τις περισσότερες φορές στηρίζονται στη χρήση διαφόρων μεθόδων εξέτασης με προτίμηση αυτών που αφήνουν το έργο

αναλλοίωτο (μη καταστρεπτικές), όπως είναι οι μέθοδοι εξέτασης με τη χρήση πολυφασματικών εικόνων.

Από την παραπάνω συνοπτική περιγραφή μπορεί εύκολα να συμπεράνει κανείς ότι οι βασικές κλάσεις που πρέπει να εισαχθούν στη μοντελοποίηση του πεδίου είναι οι Man-Made Object, Document, Assessment και Multispectral Image**.**

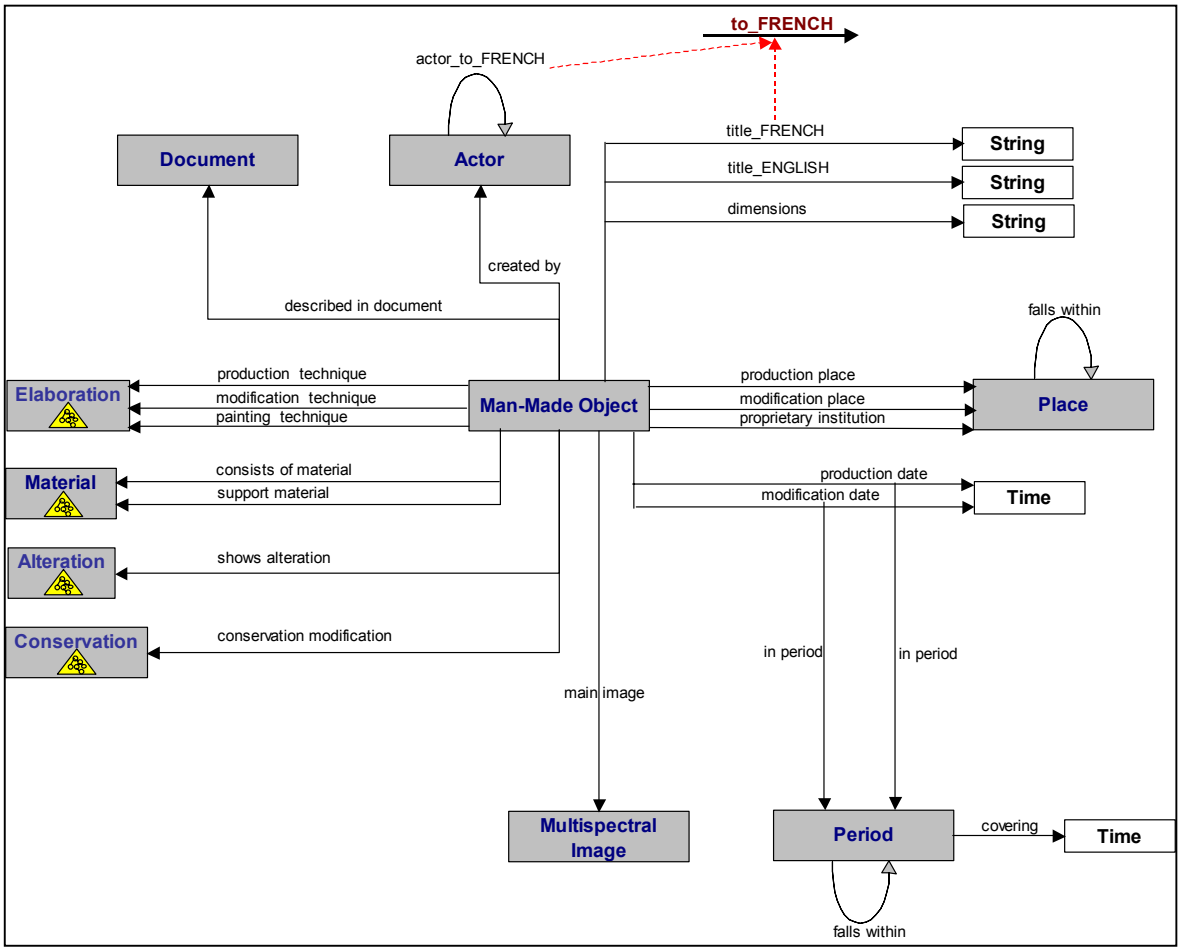
3.3.1 Η κλάση Man-Made Object

Κάθε στιγμιότυπο που ταξινομείται κάτω από την κλάση Man-Made Object αντιπροσωπεύει ένα έργο τέχνης. Οι πληροφορίες που μας ενδιαφέρουν να αποθηκεύονται για ένα έργο είναι:

1. τίτλος (στις διάφορες γλώσσες) (String)
2. δημιουργός (Actor)
3. διαστάσεις (String)
4. ο τόπος δημιουργίας (Place)
5. ημερομηνία δημιουργίας (Time)
6. τα μέρη όπου συνέβησαν τροποποιήσεις -modifications- (Place)
7. ημερομηνίες τροποποιήσεων (Time)
8. υπεύθυνο ίδρυμα (Place)
9. έγγραφο που αντιπροσωπεύει το δελτίο συντήρησης (Document)
10. φωτογραφίες που αντιστοιχούν σε αυτό και ποια είναι η κύρια απ'αυτές (λήψη στο ορατό) (Multispectral Image)
11. Κύρια τεχνική δημιουργίας (π.χ. για ζωγραφικούς πίνακες painting technique) (Elaboration)
12. Κύριο υλικό κατασκευής (π.χ. για ζωγραφικούς πίνακες support material) (Material)
13. Τεχνικές κατασκευής/τροποποιήσεων (production/modification techniques) (Elaboration)
14. Φαινόμενα αλλοιώσεων (Alteration)
15. Υλικά κατασκευής (Material)
16. Διαδικασίες συντήρησης που έχει υποστεί (Conservation)

Δίπλα στην κάθε πληροφορία βλέπουμε τον τύπο που χρησιμοποιείται για την παράστασή της. Στις περιπτώσεις primitive τύπων (String, Time) η αντίστοιχη πληροφορία δεν αποτελεί σημαντικό κριτήριο βάσει του οποίου να γίνει αναζήτηση στα πλαίσια ενός εκπαιδευτικού ηλεκτρονικού εγχειριδίου για συντηρητές.

Στο σχήμα 3.6 φαίνεται παραστατικά ένα κομμάτι του μοντέλου του πεδίου με κεντρική κλάση την Man-Made Object. Από τις κλάσεις που εικονίζονται δεν έχουμε αναφερθεί καθόλου στις Actor, Place και Period ενώ στις κλάσεις Document και Multispectral Image θα αναφερθούμε σε επόμενες παραγράφους.



Σχήμα 3.6 – «Γύρω» από την κλάση Man-Made Object

Οι κλάσεις Place και Period χρησιμοποιούνται για την οργάνωση τόπων και χρονικών περιόδων. Η ιεραρχική οργάνωση επιτυγχάνεται για τα άτομα και των δυο κλάσεων με τη χρήση της ανακλαστικής ιδιότητας *falls within*.

Έτσι για παράδειγμα κάτω από την κλάση Place μπορεί να ταξινομηθεί το άτομο WholeWorld που να συνδέεται με τη σχέση *falls within* (με κατεύθυνση προς τα πίσω) με τα άτομα Europe, Asia, Africa, America, Australia. Το άτομο America μπορεί να συνδέεται με τη σχέση *falls within* με τα άτομα NorthAmerica και SouthAmerica κ.ο.κ.

Η οργάνωση τόπων και χρονικών περιόδων κρίνεται απαραίτητη διότι είναι σημαντικό να μπορούν οι εκπαιδευόμενοι συντηρητές να μελετήσουν τις διάφορες τεχνοτροπίες, τεχνικές, υλικά κ.λ.π. που επικρατούσαν σε διάφορες περιοχές και χρονικές περιόδους όσον αφορά την κατασκευή έργων τέχνης.

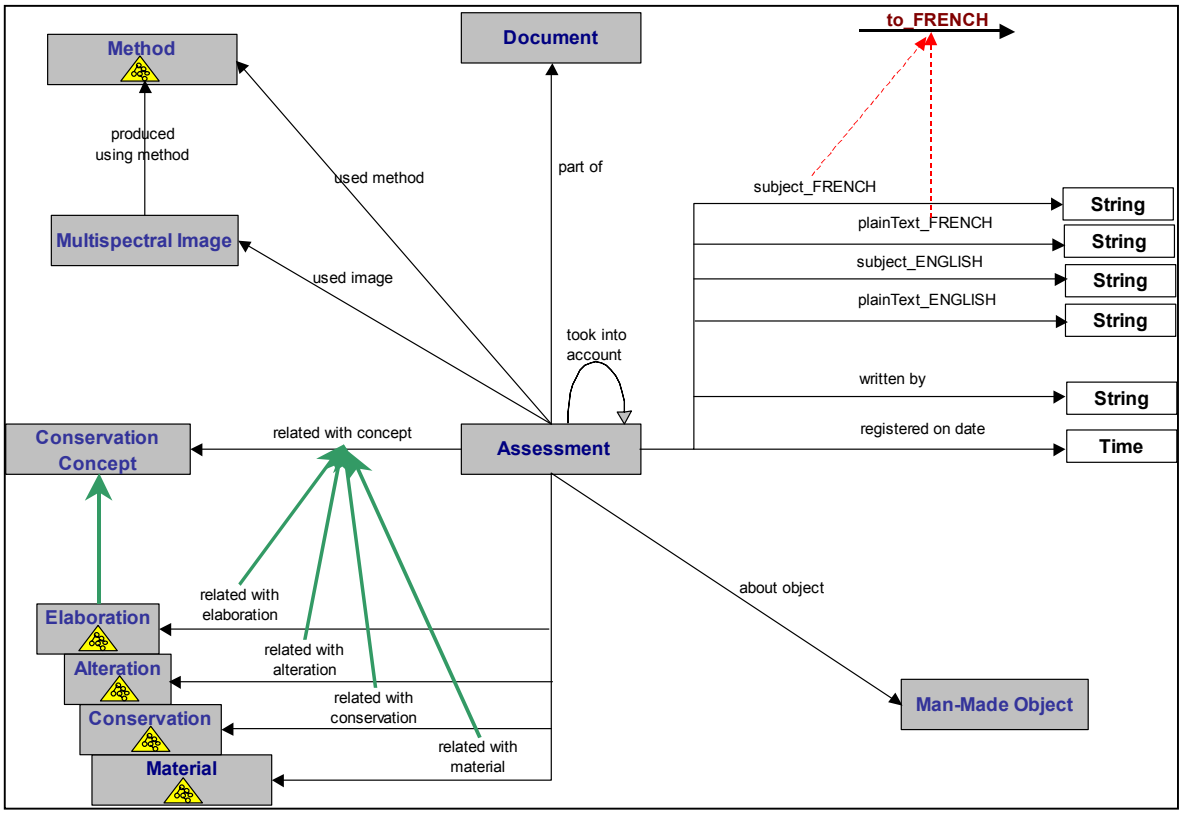
Τέλος όσον αφορά την κλάση Actor πρέπει να πούμε ότι η ανακλαστική ιδιότητα *actor_to_FRENCH* που έχει, χρησιμεύει στις επερωτήσεις για έργα βάσει του ονόματος

του δημιουργού όπως αυτό μεταφράζεται εν προκειμένω στα γαλλικά. Τα γαλλικά ονόματα δημιουργών είναι άτομα που δε συνδέονται απευθείας με άτομα τύπου Man-Made Object, παρά μόνο μέσω των αγγλικών ονομάτων (η αγγλική είναι η βασική γλώσσα). Η αναζήτηση για έργα δημιουργών και πώς αυτή υλοποιείται στο παρόν ΠΣΒΠΙ είναι θέμα που θα μας απασχολήσει στο επόμενο κεφάλαιο.

3.3.2 Η κλάση Assessment

Κάθε στιγμιότυπο της κλάσης Assessment αντιπροσωπεύει μια εκτίμηση ειδικού για ένα συγκεκριμένο έργο. Το κείμενο που περιγράφει την εκτίμηση είναι ένα απλό string που προσπελάζεται μέσω των ιδιοτήτων του τύπου *plainText_LANG*.

Οι εκτιμήσεις συνήθως βασίζονται σε πολυφασματικές εικόνες οι οποίες έχουν ληφθεί με τη χρήση κάποιας μεθόδου εξέτασης. Ωστόσο υπάρχει περίπτωση να στηρίζονται ή να λαμβάνουν υπόψη (ανακλαστική ιδιότητα *took into account*) άλλες εκτιμήσεις. Επίσης στην πλειονότητα των περιπτώσεων σχετίζονται με κάποια έννοια συντήρησης (Conservation Concept) (σχήμα 3.7).



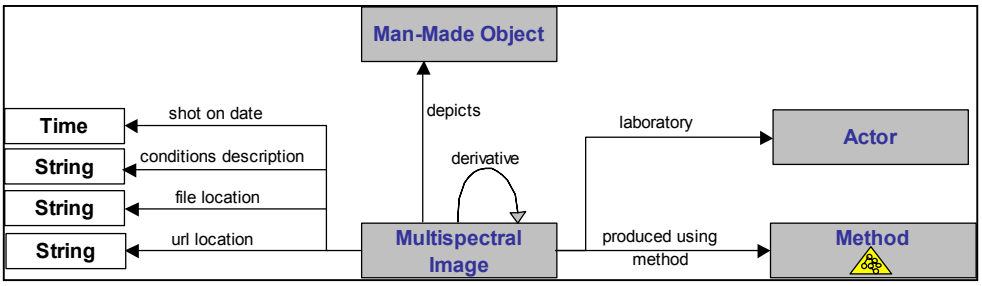
Σχήμα 3.7 - Η κλάση Assessment

Το σύνολο (aggregation) όλων των στιγμιοτύπων της κλάσης Assessment που είναι εκφρασμένες από το επίσημο ίδρυμα-κάτοχο του έργου αποτελεί το δελτίο συντήρησής του. Όλα τα στιγμιότυπα αυτά συνδέονται μέσω της ιδιότητας *part of* με το στιγμιότυπο της κλάσης Document που αντιπροσωπεύει το δελτίο συντήρησης.

Εκτιμήσεις μπορούν να διατυπωθούν και από επισκέπτες-συντηρητές του παρόντος ΠΣΒΠΙ, σχολιάζοντας μια πολυφασματική εικόνα ή μια άλλη εκτίμηση. Στην περίπτωση αυτή τα συγκεκριμένα στιγμιότυπα της κλάσης Assessment δε συνδέονται με κανένα στιγμιότυπο-Document με τη σχέση *part of*.

3.3.3 Η κλάση Multispectral Image

Κάθε στιγμιότυπο της κλάσης Multispectral Image αντιπροσωπεύει μια ψηφιακή εικόνα. Για το παρόν σύστημα οι πληροφορίες που μας ενδιαφέρουν να αποθηκεύονται για μια πολυφασματική εικόνα είναι το έργο τέχνης που απεικονίζει, η μέθοδος με την οποία παρήχθηκε, τυχόν παράγωγες εικόνες, η ημερομηνία και οι συνθήκες λήψης, το εργαστήριο λήψης, το όνομα αρχείου ή το URL όπου είναι αποθηκευμένη. Στο σχήμα 3.8 φαίνονται πώς μοντελοποιούνται αυτές οι πληροφορίες.



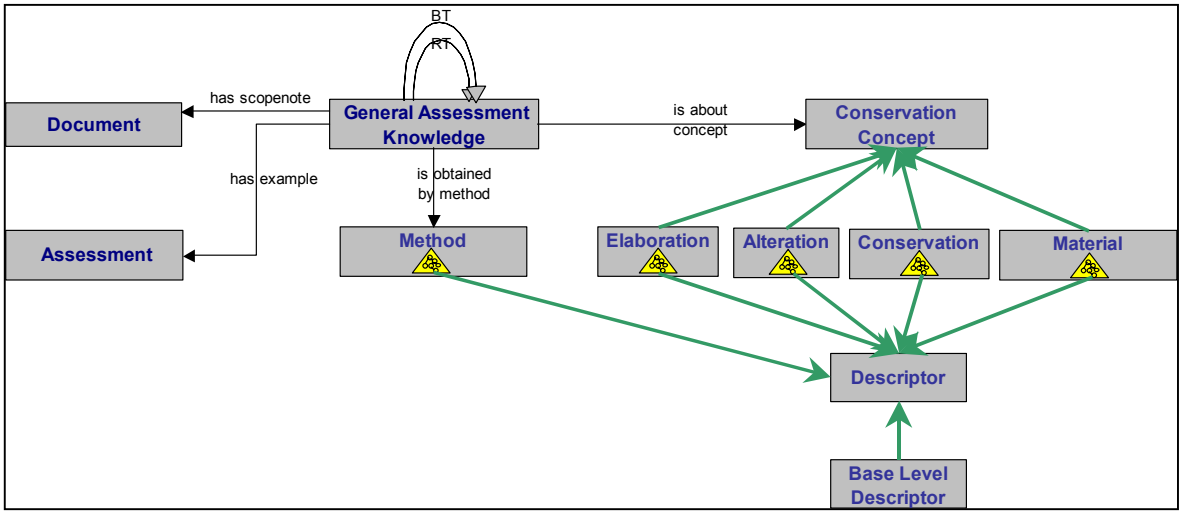
Σχήμα 3.8 - Η κλάση Multispectral Image

Ειδικά για την πληροφορία συνθηκών λήψης πρέπει να πούμε ότι τουλάχιστο για το παρόν σύστημα δεν απαιτείται να έχουμε περισσότερη δόμηση. Επειδή ωστόσο οι πολυφασματικές λήψεις εξαρτώνται σημαντικά από συγκεκριμένες παραμέτρους, προβλέπεται ότι στον primitive τύπο String που θα κρατάει την πληροφορία, θα αποθηκεύεται ένα XML string το οποίο και θα τις περιγράφει.

3.3.4 Η κλάση General Assessment Knowledge

Για τη μοντελοποίηση της γνώσης του πεδίου χρησιμοποιείται η κλάση General Assessment Knowledge. Κάθε στιγμιότυπο της κλάσης είναι ένας συνδετικός κρίκος μεταξύ μιας μεθόδου (Method) και μιας έννοιας συντήρησης (Elaboration, Alteration, Conservation, Material) (βλ. σχήμα 3.9). Το έγγραφο που περιγράφει τη γνώση που κρύβεται πίσω από αυτή τη σύνδεση, αντιπροσωπεύεται από ένα στιγμιότυπο Document το οποίο «δείχνει» σε κάποιο εξωτερικό (εκτός του ΣΔΒΔ) αρχείο κειμένου.

Οι γνώσεις του πεδίου εμφανίζουν την ιδιότητα να μπορούν να οργανωθούν ιεραρχικά. Αυτή η ιδιότητα είναι ιδιαίτερα σημαντική για τους εκπαιδευόμενους συντηρητές και τους επιτρέπει να μεταβαίνουν από τη γενική στην ειδική γνώση και το αντίστροφο. Για το σκοπό αυτό έχουν εισαχθεί στην κλάση General Assessment Knowledge οι ανακλαστικές ιδιότητες *BT*, *RT* ώστε να οργανωθούν τα στιγμιότυπά της ως να ήταν όροι θησαυρού.



Σχήμα 3.9 - Μοντελοποίηση γνώσης του πεδίου - General Assessment Knowledge

Για ένα στιγμιότυπο γνώσης προβλέπεται και η σύνδεσή του με ένα ή περισσότερα στιγμιότυπα Assessment τα οποία θα είναι χαρακτηριστικά παραδείγματα που επαληθεύουν τη συγκεκριμένη γνώση.

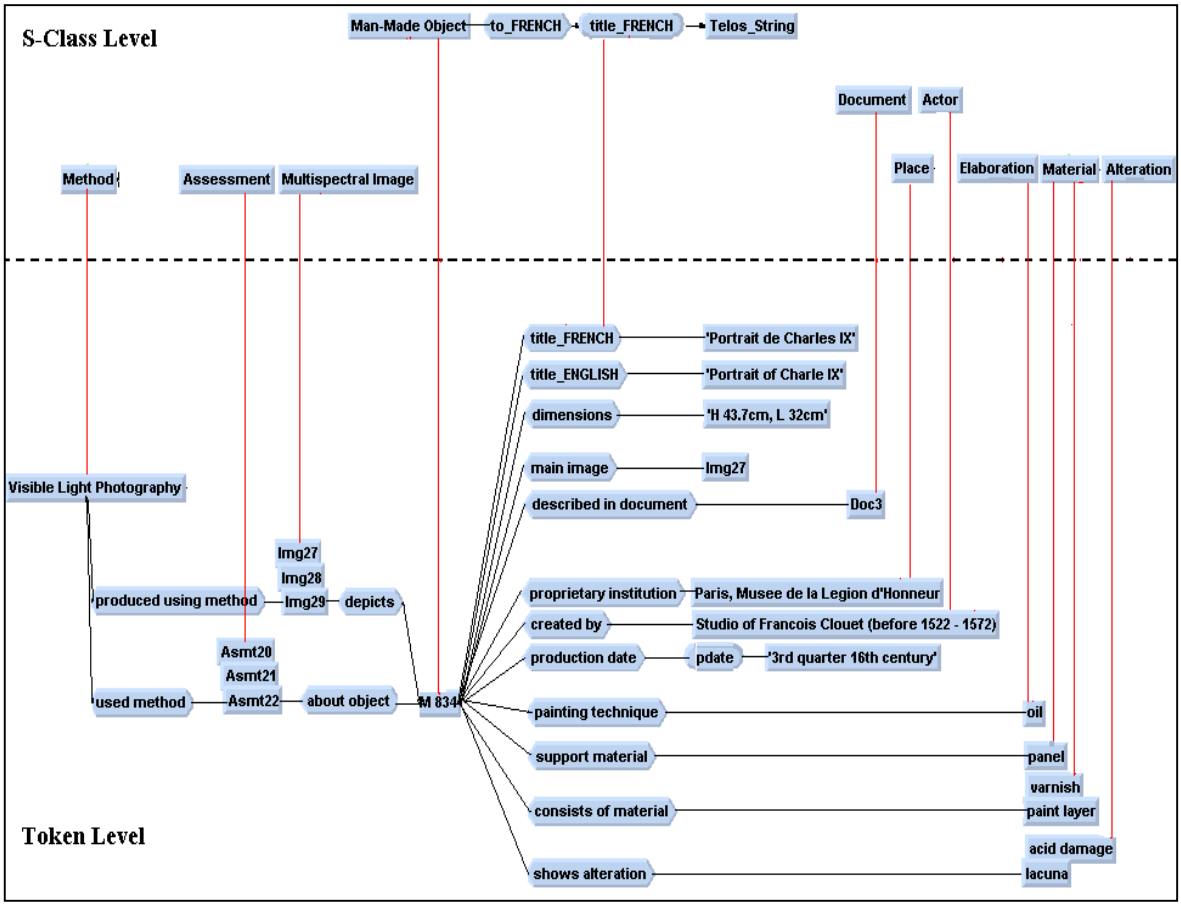
3.4 Παράδειγμα καταχωρισμένου πίνακα ζωγραφικής

Έστω ότι θέλουμε να δούμε ένα παράδειγμα καταχωρισμένου πίνακα ζωγραφικής και όλα τα σχετικά μ' αυτόν αντικείμενα στη βάση πληροφοριών (Token Level) που υλοποιείται με το μοντέλο δεδομένων της Telos. Στο συγκεκριμένο σημείο δε θα απασχοληθούμε καθόλου με τους όρους του θησαυρού ή κόμβους γνώσης.

Έστω λοιπόν ο ζωγραφικός πίνακα με κωδικό Legiond'Honneur`M 834 και τίτλο στα αγγλικά Portrait of Charles IX και στα γαλλικά Portrait de Charles IX. Ο πίνακας χρονολογείται στο 3rd quarter 16th century, έχει διαστάσεις 43.7cm ύψος και 32cm μήκος. και κατασκευάστηκε στο Studio of Francois Clouet (γαλλιστί Atelier de Francois Clouet). Η τεχνικής ζωγραφικής είναι λαδομπογιά – oil και το υλικό υποστήριξης panel. Το υπεύθυνο ίδρυμα που έχει την κυριότητα του έργου είναι το Paris, museum de la Legion d'Honneur.

Το δελτίο συντήρησης για τον πίνακα αντιπροσωπεύεται από τον κόμβο Doc3 και μπορεί να χωριστεί σε τρία λογικά μέρη τα οποία αποτελούν και τις επίσημες εκτιμήσεις

Asmt20, Asmt21, Asmt22 συντηρητών του υπεύθυνου ιδρύματος. Η Asmt20 εμπεριέχει γενικές πληροφορίες για τον πίνακα και σχετίζεται με τη βασική εικόνα Img27, η οποία είναι η φωτογραφία ολόκληρου του πίνακα στο ορατό (Visible Light Photography). Η Asmt21 περιγράφει τις αλλοιώσεις στο βερνίκι–varnish και στο ζωγραφικό στρώμα–paint layer που προκλήθηκαν από πρόσπτωση οξέος (acid damage). Η φωτογραφία που εικονίζει αυτή την αλλοίωση στα υλικά του πίνακα είναι η Img28. Τέλος η Asmt22 περιγράφει ένα κενό–lacuna που δημιουργήθηκε από την βαθιά διείσδυση του οξέος στο υλικό υπόστρωμα και εικονίζεται στην εικόνα Img29. Οι Img28, Img29 είναι φωτογραφίες ληφθείσες επίσης στο ορατό και παράγωγες της βασικής εικόνας–*main image* Img27.



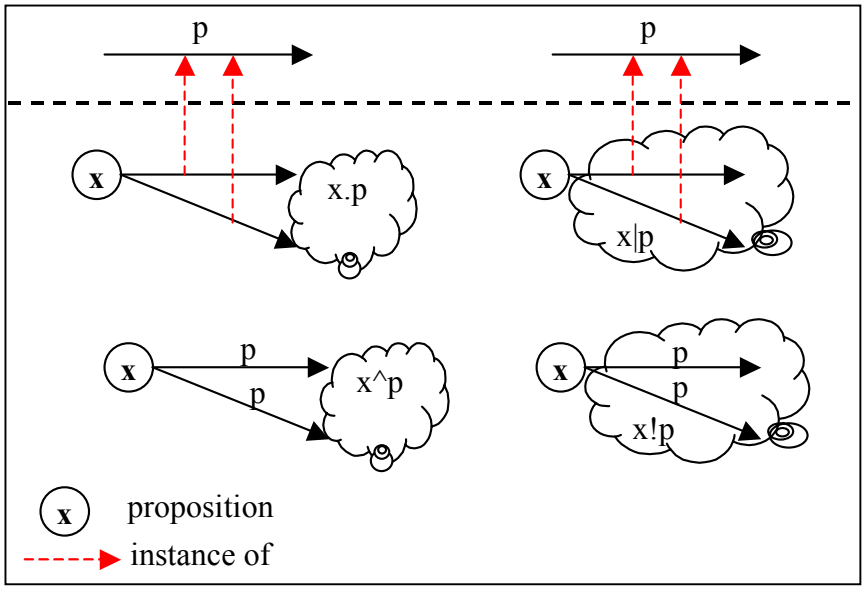
Σχήμα 3.10 - Παράδειγμα καταχωρισμένου πίνακα ζωγραφικής

3.5 Περιορισμοί ακεραιότητας

Για το μοντέλο που παρουσιάστηκε προηγουμένως πρέπει να ισχύουν ένα σύνολο από επαγωγικούς κανόνες και περιορισμούς ακεραιότητας ώστε τα δεδομένα που θα είναι καταχωρισμένα στο σύστημα να είναι συνεπή με ό,τι ισχύει στην πραγματικότητα.

3.5.1 Η Assertional Language της Telos

Η Telos παρέχει μια δηλωτική υπο-γλώσσα (Assertional Language – AL [18], [49], [50]), για να εκφράσει κανόνες και περιορισμούς με τη βοήθεια λογικής $1^{ης}$ τάξεως. Η AL περιλαμβάνει ειδικές δηλώσεις για τις σχέσεις isA και instanceOf (isa και in αντίστοιχα), καθώς και ειδικούς επιλογείς (x.p , x^p, x|p, x!p βλ.σχήμα 3.11) που επιτρέπουν την «πλοήγηση» στο γράφο του εκάστοτε μοντέλου.



Σχήμα 3.11 - Οι επιλογείς της Assertional Language της Telos

Σ' αυτή τη γλώσσα ορίζονται και τα αξιώματα για το μοντέλο δεδομένων που εισάγει η Telos:

Αξίωμα των προτάσεων (propositions) :

$$(\forall p,x,y,z/Proposition)(\forall t/Time)$$
$$(prop(p,x,y,z,t) \Rightarrow from(p)=x \wedge label(p)=y \wedge to(p)=z \wedge when(p)=t)$$

Μεταβατικότητα της σχέσης isA :

$$(\forall p1,p2,p3/Proposition)(\forall t1,t2,t3/Time)$$
$$(isa(p1,p2,t1) \wedge isa(p2,p3,t2) \wedge t3=t1*t2 \Rightarrow isa(p1,p3,t3)$$

Αξίωμα της εξειδίκευσης : *«Η έκταση μιας κλάσης είναι υπερσύνολο της έκτασης οποιασδήποτε από τις υποκλάσεις της»*

$$(\forall p1, p2, p3 / Proposition)(\forall t1, t2, t3 / Time)$$
$$(in(p1, p2, t1) \wedge (isa(p2, p3, t2) \wedge t3 = t1 * t2 \Rightarrow in(p1, p3, t3)$$

Ο περιορισμός της υποστασιοποίησης (instantiation constraint) : *«Αν το αντικείμενο p1 είναι στιγμιότυπο του αντικειμένου p2 τότε το from(p1) πρέπει να είναι στιγμιότυπο του from(p2), το to(p1) πρέπει να είναι στιγμιότυπο του to(p2) και when(p1) πρέπει να επικαλύπτει το when(p2)»*

$$(\forall p1, p2 / Proposition)(\forall t1 / Time)$$
$$(InstanceOf(p1, p2, t1) \Rightarrow (\exists t2, t3)(in(from(p1), from(p2), t2) \wedge (in(to(p1), to(p2), t3) \wedge$$
$$during(t1, t2) \wedge during(t1, t3) \wedge overlaps(when(p1), when(p2)))$$

Οι περιορισμοί ακεραιότητας και οι επαγωγικοί κανόνες ενός μοντέλου εκφράζονται ως κλειστοί καλά ορισμένοι τύποι (well formed formulas) της AL (αναδρομικά ορισμένοι από τους ατομικούς τύπος in, isa).

Έτσι ένας περιορισμός ακεραιότητας μπορεί να έχει μια από τις μορφές:

$I \equiv \forall x_1 / C_1 ... \forall x_k / C_k$ F

$I \equiv \exists x_1 / C_1 ... \exists x_k / C_k$ F όπου F είναι μια well formed formula.

Και ένας επαγωγικός κανόνας μπορεί να έχει τη μορφή:

$DR \equiv \forall x_1 / C_1 ... \forall x_k / C_k$ $(\mathrm{F} \Rightarrow \mathrm{A})$ όπου F,A είναι well formed formulas.

3.5.2 Περιγραφή-τυποποίηση περιορισμών και κανόνων μοντέλου

Κατά κύριο λόγο το σημασιολογικό μοντέλο που περιγράψαμε απαιτεί περιορισμούς μονότιμων γνωρισμάτων. Ωστόσο υπάρχουν και άλλου είδους περιορισμοί σχετικά με την ιεραρχική οργάνωση θησαυρού, περιοχών (Places), περιόδων (Periods) και την ταξινόμηση όρων. Επίσης πρέπει να οριστούν και επαγωγικοί κανόνες που να πυροδοτούν ενέργειες απόδοσης τιμών σε γνωρίσματα όταν πάρουν τιμή κάποια άλλα. Ακολουθεί η περιγραφή και τυποποίηση κανόνων και περιορισμών για το μοντέλο.

1. Κάθε στιγμιότυπο της κλάσης Descriptor έχει το πολύ μία μετάφραση στη γλώσσα LANG[5], η οποία είναι στιγμιότυπο της κλάσης LANG_Descriptor.

$$(\forall x / Descriptor)(\forall z, z' / LANG\_Descriptor)$$
$$(descriptor\_to\_LANG(x, z) \wedge descriptor\_to\_LANG(x, z') \Rightarrow z = z'$$

[5] το LANG συμβολίζει μια γλώσσα και μπορεί να πάρει τιμές FRENCH, GERMAN κλπ...

Ο ίδιος περιορισμός ισχύει και για τις κλάσεις Document και ScopeNote, έχουν δηλαδή το πολύ μία μετάφραση από τη βασική γλώσσα στη γλώσσα LANG.

2. Κάθε όρος-στιγμιότυπο της κλάσης Descriptor έχει το πολύ ένα στιγμιότυπο της κλάσης ScopeNote που τον περιγράφει.

$$(\forall x / Descriptor)(\forall z, z' / ScopeNote)$$
$$(has\ scopenote(x, z) \wedge has\ scopenote(x, z') \Rightarrow z = z'$$

3. Κάθε όρος του θησαυρού ταξινομείται αποκλειστικά σε μία εκ των υπο-κλάσεων της κλάσης Descriptor (Elaboration, Alteration, Conservation, Material, Method).

$$(\forall C, C', x / Proposition)$$
$$[[isa(C, Descriptor) \wedge isa(C', Descriptor) \wedge C \neq C'] \Rightarrow \neg\exists\, x\ [in(x, C) \wedge in(x, C')]]$$

4. Στην ιεραρχία που ορίζουν τα στιγμιότυπα κάθε υπο-κλάσης της κλάσης Descriptor με χρήση της σχέσης BT δεν επιτρέπονται κύκλοι. Για να ορίσουμε τον περιορισμό αρχικά εκφράζουμε τη μεταβατικότητα της σχέσης BT.

$$(\forall C, x, y, z / Proposition)(isa(C, Descriptor))(in(x, C))(in(y, C))(in(z, C)) \Rightarrow$$
$$[BT(x, y) \wedge BT(y, z) \Rightarrow BT(x, z)]$$

$$\neg\exists \mathrm{x}\ [\mathrm{BT}(\mathrm{x}, \mathrm{x})]$$

Το ίδιο ισχύει για τα στιγμιότυπα της κλάσης General Assessment Knowledge πάλι για τη σχέση *BT*. Επίσης ισχύει και για τα στιγμιότυπα των κλάσεων Place και Period και τη σχέση *falls within*.

5. Κάθε στιγμιότυπο γνώσης (General Assessment Knowledge) συνδέει απαραίτητα ακριβώς μία μέθοδο (Method) και ακριβώς μία έννοια συντήρησης (Conservation Concept). Υπάρχει το πολύ ένα στιγμιότυπο γνώσης που να συνδέει ένα συγκεκριμένο συνδυασμό στιγμιοτύπων (Method, Conservation Concept). Επίσης υπάρχει ακριβώς ένα στιγμιότυπο Document που περιγράφει μια συγκεκριμένη γνώση (στιγμιότυπο General Assessment Knowledge).

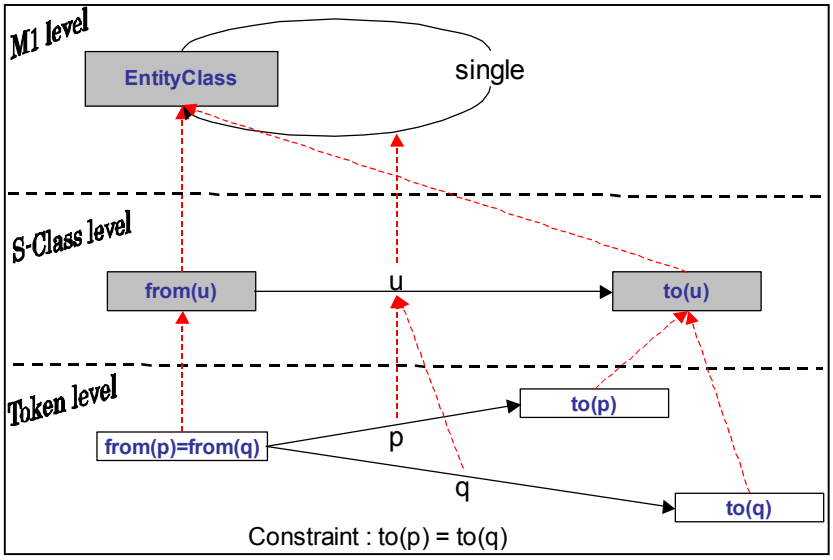
$$(\forall x / GeneralAssessmentKnowledge)(\exists y / Method, \exists z / ConservationConcept)$$
$$[is\ obtained\ by\ method(\mathrm{x}, \mathrm{y}) \wedge is\ about\ concept(\mathrm{x}, \mathrm{z})]$$

$$(\forall x / GeneralAssessmentKnowledge, \forall y, y' / Method, \forall z, z' / ConservationConcept)$$
$$((is\ obtained\ by\ method(\mathrm{x}, \mathrm{y}) \wedge is\ about\ concept(\mathrm{x}, \mathrm{z})) \wedge$$
$$(is\ obtained\ by\ method(\mathrm{x}, \mathrm{y}') \wedge is\ about\ concept(\mathrm{x}, \mathrm{z}')) \Rightarrow y = y' \wedge z = z'$$

$$(\forall x / GeneralAssessmentKnowledge)(\exists y / Document\ [has\ scopenote(x,y)])$$

$$(\forall x / GeneralAssessmentKnowledge, \forall y, y' / Document)$$
$$(has\ scopenote(x,y) \wedge has\ scopenote(x, y')) \Rightarrow y = y'$$

6. Για κάθε στιγμιότυπο Man-Made Object υπάρχει ακριβώς ένα έγγραφο Document (που έχει το ρόλο του δελτίου συντήρησης), ένας δημιουργός Actor, ένας τόπος δημιουργίας Place, μία ημερομηνία δημιουργίας production date η οποία ανήκει σε μία αποκλειστικά περίοδο Period, μία βασική τεχνική κατασκευής και ένα βασικό υλικό κατασκευής (για πίνακες ζωγραφικής painting technique και support material αντίστοιχα) και μία βασική ψηφιακή εικόνα main image στο ορατό.
   Για την τυποποίηση των παραπάνω περιορισμών («ακριβώς ένα») ορίζουμε τη μετακλάση EntityClass [50] (βλ. σχήμα 3.12) και περιορισμό ακεραιότητας atMostOne για την ανακλαστική μετα-ιδιότητα αυτής single.



Σχήμα 3.12 - Μοντελοποίηση περιορισμού ακεραιότητας μονότιμων γνωρισμάτων

Σύμφωνα με τον περιορισμό atMostOne για οποιαδήποτε στιγμιότυπα p και q ενός στιγμιοτύπου u της μετα-ιδιότητας single, αν τα p και q έχουν την ίδια πηγή τότε θα έχουν και τον ίδιο προορισμό. Με άλλα λόγια το u είναι μια μονότιμη ιδιότητα, αφού είναι στιγμιότυπο της EntityClass!single και γι' αυτό κανένα στιγμιότυπο της πηγής του δεν θα πρέπει να συνδέει 2 από τα στιγμιότυπά του ως ιδιότητες.
Οι περιορισμοί ακεραιότητας τύπου «ακριβώς ένα» που εμείς χρειαζόμαστε μπορούν εύκολα να εκφραστούν φορμαλιστικά με τη βοήθεια της single. Έτσι επιλεκτικά έχουμε:

$$in(ManMadeObject!described\ in\ document, single)$$
$$(\forall x / ManMadeObject)(\exists y / ManMadeObject!described\ in\ document\ [from(y) = x])$$

$$in(ManMadeObject!\,production\ date, single)$$
$$in((ManMadeObject!\,production\ date)!in\ \ period, single)$$
$$(\forall x / ManMadeObject)(\exists y /(ManMadeObject!\,production\ date\,[\,from(y) = x]) \wedge$$
$$(\exists z /(ManMadeObject!\,production\ date)!in\ \ period\,[\,from(z) = y])$$

7. Ένας χρονικός περιορισμός:

$$(\forall t1, t2 / Time)$$
$$(\forall x / ManMadeObject!\,production\ date)(\forall y / ManMadeObject!modification\ date)$$
$$[t1 = to(x) \wedge t2 = to(y) \Rightarrow t1 <= t2]$$

8. Το βασικό υλικό κατασκευής – support material είναι υλικό από το οποίο αποτελείται ένα έργο τέχνης. Επίσης η βασική τεχνική κατασκευής – painting technique είναι τεχνική παραγωγής για το έργο. Αυτές οι προτάσεις οδηγούν στην διατύπωση των παρακάτω επαγωγικών κανόνων (deduction rules).

$$(\forall x / ManMadeObject)(\forall y / Elaboration)$$
$$[\,painting\ technique(x,y) \Rightarrow\ production\ \ technique(x, y)$$

$$(\forall x / ManMadeObject)(\forall y / Material)$$
$$[support\ material(x,y) \Rightarrow consists\ of\ \ material(x, y)$$

9. Κάθε στιγμιότυπο Multispectral Image απεικονίζει ακριβώς ένα έργο τέχνης και έχει παραχθεί με τη χρήση ακριβώς μιας μεθόδου.

$$(\forall x / Multispectral\ Image)(\exists y / Method, \exists z / Man - Made\ Object)$$
$$[\,produced\ using\ method(x,y) \wedge depicts(x,z)]$$

$$(\forall x / Multispectral\ Image, \forall y, y' / Method, \forall z, z' / Man - Made\ Object)$$
$$((produced\ using\ method(x,y)\ \ \wedge\ \ depicts(x,z)) \wedge$$
$$(produced\ using\ method(x,y')\ \ \wedge\ \ depicts(x,z')) \Rightarrow y = y' \wedge z = z'$$

10. Επαγωγικοί κανόνες που προκύπτουν σχετικά με ένα στιγμιότυπο Multispectral Image βάσει του υπάρχοντος μοντέλου:

$$(\forall x / Multispectral\ Image, \forall y / Method, \forall z / Man - Made\ Object, \forall a / Assessment)$$
$$[\,produced\ using\ method(x,y) \wedge depicts(x,z) \wedge used\ image(a, x) \Rightarrow$$
$$used\ method(a, y) \wedge about\ object(a, z)]$$

3.6 Παρατηρήσεις – Συμπεράσματα

Το σημασιολογικό μοντέλο που περιγράφηκε, σχεδιάστηκε εξ' αρχής με σκοπό να υποστηρίξει ένα εκπαιδευτικό ηλεκτρονικό εγχειρίδιο για συντηρητές. Για το λόγο αυτό δεν αποτελεί σε καμία περίπτωση μια πλήρη μοντελοποίηση του πεδίου συντήρησης/αποκατάστασης έργων τέχνης. Αποτελεί ωστόσο μια μοντελοποίηση προσανατολισμένη να υποστηρίξει εκπαιδευτικές λειτουργίες :

- μέσα από την οργάνωση πολύγλωσσου θησαυρού όρων με τις απλούστερες από τις σχέσεις (*BT, RT, UF*)
- μέσω της οργάνωσης γνώσεων που έχουν αποκτηθεί και που συσχετίζουν μεθόδους εξέτασης με έννοιες συντήρησης
- μέσω παραδειγμάτων-πινάκων που κάνουν χειροπιαστές τις γνώσεις

Οι εκπαιδευτικές λειτουργίες είναι δυο ειδών, οι λειτουργίες ανάκτησης/ παρουσίασης δεδομένων σύμφωνα με προκαθορισμένα υποδείγματα (templates) και οι λειτουργίες σχολιασμού εικόνων ή εκτιμήσεων συντηρητών από επισκέπτες του δικτυακού τόπου. Το πρώτο είδος έχει πλήρως υλοποιηθεί με τη χρήση των τεχνολογιών που αναφέρθηκαν στο $2^{ο}$ κεφάλαιο και παρουσιάζεται διεξοδικά στο επόμενο κεφάλαιο. Το δεύτερο είδος παρουσιάζεται στο $5^{ο}$ κεφάλαιο όπου αναφέρεται αναλυτικά η σχεδίαση της εισαγωγής δεδομένων στο σύστημα.

Η παρούσα μοντελοποίηση υποστηρίζει πλήρως τις λειτουργίες του συστήματος αρκεί στο υποσύστημα εισαγωγής δεδομένων να υλοποιηθούν οι περιορισμοί ακεραιότητας που περιγράφονται στην παράγραφο §3.5.

Κεφάλαιο 4

Σχεδίαση και Υλοποίηση Λειτουργιών Συστήματος

Στο κεφάλαιο αυτό αφού πρώτα παρουσιάζεται η σύλληψη που έγινε για τη δόμηση της δικτυακής εφαρμογής στη συνέχεια περιγράφονται αναλυτικά οι λειτουργίες που υλοποιήθηκαν για το παρόν ΠΣΒΠΙ. Για κάθε λειτουργία αναφέρονται οι παράμετροι κλήσης, η δομή των δεδομένων-εξόδου, η υλοποίηση των επερωτήσεων, η μορφοποίηση του παραγόμενου ρεύματος XML κ.α. Ακολουθεί ένας μικρός σχολιασμός για τη σχεδίαση της διεπαφής χρήστη και τέλος παρουσιάζεται και σχολιάζεται ένας πίνακας επιδόσεων λειτουργιών με τη χρήση εικονικών δεδομένων.

4.1 Σχεδίαση εφαρμογής

Η σχεδίαση της εφαρμογής έγινε με πρωταρχικό στόχο την ικανοποίηση των αναγκών για πλήρη παραμετροποίηση, υποστήριξη πολυγλωσσίας και υποστήριξη πολλαπλών παρουσιάσεων. Η πλήρης παραμετροποίηση κρίθηκε απαραίτητη για την απρόσκοπτη εγκατάσταση της δικτυακής εφαρμογής σε οποιοδήποτε υπολογιστή χωρίς την ανάγκη επεμβάσεων στον πηγαίο κώδικα και επαναμεταγλώττισής του. Η πολυγλωσσία ήταν απαιτούμενο εξ' αρχής χαρακτηριστικό για το παρόν ΠΣΒΠΙ, ενώ η υποστήριξη πολλαπλών παρουσιάσεων είναι χαρακτηριστικό που επιπρόσθετα υλοποιήθηκε και δίνει αρκετές δυνατότητες για επεκτάσεις από την πλευρά της διεπαφής χρήστη.

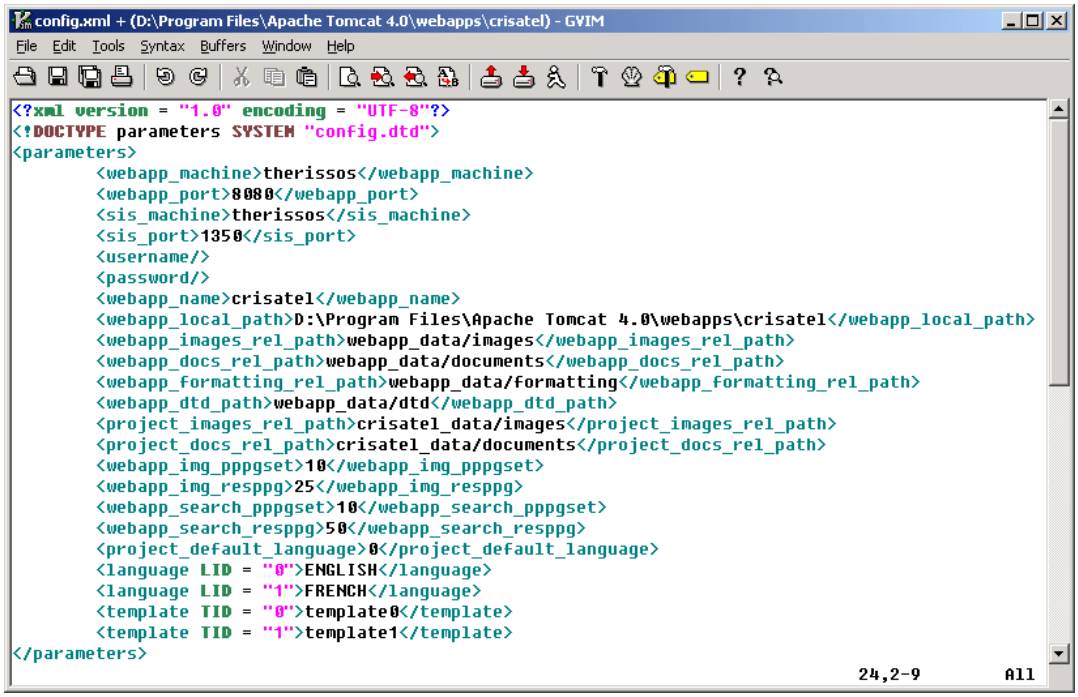
4.1.1 Παραμετροποίηση

Σε μια δικτυακή εφαρμογή που επικοινωνεί με ένα ΣΔΒΔ και πρόκειται να χρησιμοποιεί και να διαχειρίζεται ένα σύνολο πόρων όπως αρχεία κειμένου, εικόνες, αρχεία μορφοποίησης XML εγγράφων κ.α. είναι απολύτως απαραίτητος ο ορισμός εξωτερικών παραμέτρων. Οι παράμετροι αυτοί θα πρέπει να είναι εύκολα προσβάσιμοι ώστε να μεταβάλλονται κατά βούληση από εγκατάσταση σε εγκατάσταση.

Για το παρόν σύστημα διακρίναμε ως απαραίτητες τις εξής παραμέτρους: όνομα ή διεύθυνση IP και port εξυπηρετητή του ΠΙ, όνομα ή διεύθυνση IP και port εξυπηρετητή ΣΔΒΔ, όνομα εφαρμογής, πλήρη διαδρομή για τον φάκελο-ρίζα της δικτυακής εφαρμογής, φάκελο εικόνων δικτυακής εφαρμογής, φάκελο-ρίζα αρχείων μορφοποίησης,

φάκελο αρχείων Ορισμού Τύπων Εγγράφων (DTD), φάκελο εικόνων πεδίου εφαρμογής, φάκελο εγγράφων πεδίου εφαρμογής, πλήθος αποτελεσμάτων ανά σελίδα και σελίδων ανά σύνολο σελίδων (pageset) στις περιπτώσεις αναζήτησης ή εμφάνισης εικόνων και τέλος χαρακτηριστικές ονομασίες για κάθε γλώσσα (ENGLISH, FRENCH κλπ) και για κάθε στιλ παρουσίασης (template0, template1 κλπ).

Όλες οι παράμετροι ορίστηκαν στο αρχείο config.xml που εικονίζεται στο σχήμα 4.1 και το οποίο ακολουθεί τον τύπο config.dtd.



Σχήμα 4.1 - Το αρχείο παραμέτρων config.xml

Πώς όμως χρησιμοποιούνται οι παράμετροι αυτοί από τη δικτυακή εφαρμογή; Με κάποιο τρόπο θα πρέπει να διαβάζονται κατά την έναρξη εκτέλεσης της κύριας συνάρτησης *doGet()* του κάθε Servlet της εφαρμογής. Το διάβασμα αυτό το εκτελεί η κλάση **ReadConfig** που κατασκευάσαμε για το σκοπό αυτό. Η κλάση αυτή χρησιμοποιεί τον αναλυτή XML (XML parser) με όνομα crimson, διαβάζει το αρχείο **config.xml** και αποδίδει σε αντίστοιχες μεταβλητές της τις τιμές που είναι ορισμένες σ' αυτό. Δεν επιβαρύνει αυτό ωστόσο την εκτέλεση κάθε Servlet; Στην πραγματικότητα όχι διότι το αντικείμενο της κλάσης ReadConfig είναι στατικό και «ζει» καθ' όλη τη διάρκεια ζωής της δικτυακής εφαρμογής.

Έτσι με χρήση του κώδικα:

```
private static ReadConfig I;
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
  if (I==null) { I = new ReadConfig(); }
```

εξασφαλίζεται ότι το διάβασμα των παραμέτρων γίνεται μόνο κατά την εκτέλεση του πρώτου Servlet της εφαρμογής και ποτέ άλλοτε σε όλη τη διάρκεια ζωής της! (μέχρι δηλαδή την επόμενη επανεκκίνηση του εξυπηρετητή εφαρμογής). Ο παραπάνω κώδικας ενσωματώνεται σε όλα τα Servlets της εφαρμογής.

Το αρχείο config.xml μπορεί να εμπλουτιστεί με νέα παράμετρο έστω *prm* εφόσον μελλοντικά προκύψει κάποια νέα ανάγκη. Αυτό προαπαιτεί τα εξής βήματα:

- Κατάλληλη προσθήκη της *prm* στο αρχείο config.dtd
- Ορισμός μεταβλητής *prm* στην κλάση ReadConfig καθώς και αρχικοποίηση και ανάθεση τιμής σε αυτή (συνολικά περίπου 5 γραμμές κώδικα)
- Επαναμεταγλώττιση της δικτυακής εφαρμογής

Η νέα παράμετρος θα μπορεί πλέον να προσπελάζεται από ένα Servlet μέσω της μεταβλητής *I.prm* .

Το μόνο που δεν παραμετροποιείται στην παρούσα εφαρμογή είναι η θέση του αρχείου config.xml. Ανάλογα με το λειτουργικό σύστημα και τον τρόπο εκτέλεσης του εξυπηρετητή εφαρμογής (ως υπηρεσία ή χειρωνακτικά), το αρχείο αναζητείται από την κλάση ReadConfig ή στον φάκελο συστήματος (π.χ. \WINNT\System32 κλπ.) ή στον κατάλογο εκτέλεσης της δικτυακής εφαρμογής (...Apache Tomcat 4.0\webapps\crisatel).

4.1.2 Πολυγλωσσία και άμεση εναλλαγή γλώσσας

Η επίτευξη του χαρακτηριστικού της πολυγλωσσίας γίνεται σε δυο επίπεδα.

Το πρώτο επίπεδο είναι το επίπεδο του ΣΔΒΔ. Σ' αυτό το επίπεδο εφόσον απαιτείται γίνεται η μετάφραση όρων, περιγραφών όρων (scopenotes), ονομάτων δημιουργών, τίτλων έργων, εγγράφων (εύρεση μόνο ονόματος και όχι όλης της διαδρομής όπου αυτό βρίσκεται) κλπ. Η μετάφραση επιτυγχάνεται με την εκτέλεση κατάλληλων επιπλέον επερωτήσεων στο SIS (επιπλέον των επερωτήσεων που απαιτούνται για τη βασική γλώσσα) βάσει του μοντέλου που περιγράφηκε στο κεφάλαιο 3.

Τα αποτελέσματα που εμφανίζονται στον επισκέπτη του παρόντος ΠΣΒΠΙ με το πέρας εκτέλεσης μιας λειτουργίας δεν περιορίζονται στα δυναμικά δεδομένα που ανακτώνται από το υποκείμενο ΣΔΒΔ. Απλά παραδείγματα είναι οι διάφορες ετικέτες που εμφανίζονται και οι οποίοι προσδιορίζουν τα εμφανιζόμενα δεδομένα, οι λεζάντες-τίτλοι των λειτουργιών, τα μενού κ.α. Η μετάφραση αυτών των στατικών δεδομένων γίνονται στο δεύτερο επίπεδο, το επίπεδο των αρχείων μορφοποίησης XSL. Κάθε λειτουργία έχει για κάθε γλώσσα[6] ξεχωριστό αρχείο XSL το οποίο βρίσκεται σε κατάλληλα ορισμένο φάκελο για το σκοπό αυτό. Στα αρχεία αυτά υπάρχει ο κατάλληλος κώδικας (XHTML) για την εμφάνιση στο φυλλομετρητή των διαφόρων ετικετών (στην κατάλληλη κάθε φορά

[6] και για κάθε στιλ παρουσίασης

γλώσσα) και του «πλαισίου» όπου θα εμφανιστούν τα δυναμικά δεδομένα με την βοήθεια των λεγόμενων XSL tags.

Οι αρμοδιότητες των δυο επιπέδων για την πλήρη επίτευξη μετάφρασης συνδυάζονται ως εξής για μια λειτουργία:

1. Διάβασμα της παραμέτρου LID της λειτουργίας που προσδιορίζει τον κωδικό της γλώσσας στην οποία θέλουμε να εμφανιστούν τα αποτελέσματα της λειτουργίας
2. Αν η παράμετρος LID έχει τιμή 0 τότε η γλώσσα εμφάνισης αποτελεσμάτων είναι η βασική οπότε δε χρειάζεται η εκτέλεση επιπλέον επερωτήσεων (μετάφρασης) απ' αυτές που απαιτούνται για την ίδια την εκτέλεση της λειτουργίας. Σε αντίθετη περίπτωση εκτελούνται οι απαραίτητες επερωτήσεις μετάφρασης στη γλώσσα που προσδιορίζει η τιμή του LID (η γλώσσα είναι η *I.LANG[LID]* βλ. §4.1.1)
3. Όταν έχουν συλλεχθεί όλα τα απαραίτητα δεδομένα, γίνεται η μορφοποίηση τους σε ρεύμα XML (XML stream) σύμφωνα με το DTD που προσδιορίζεται στην αρχή του ρεύματος αυτού. Στην αρχή του ρεύματος προσδιορίζεται και το αρχείο XSL που θα μορφοποιήσει το ρεύμα. Αυτό θα βρίσκεται σε φάκελο του οποίου η διαδρομή θα εξαρτάται από την παράμετρο LID (δηλ. τη γλώσσα). Τα DTD για τις λειτουργίες είναι συνταγμένα κατά τέτοιο τρόπο ώστε ό,τι έχει μετάφραση και μπορεί να αποτελέσει παράμετρο για άλλη λειτουργία (κατά κύριο λόγο αυτό ισχύει για τους Descriptors), θα πρέπει να συνοδεύεται από το ισοδύναμό του στη βασική γλώσσα. Μ' αυτόν τον τρόπο όλες οι σχετικές με όρους λειτουργίες καλούνται με παραμέτρους-όρους στη βασική γλώσσα. Αυτό απαλάσσει από τη μετάφραση στη βασική γλώσσα των διαφόρων παραμέτρων κατά την αρχή εκτέλεση ενός Servlet όταν η επιλεγμένη γλώσσα είναι δευτερεύουσα.
4. Το ρεύμα XML που παράγεται στο βήμα 3 είναι αυτοδύναμο και προσδιορίζει πλήρως το τι θα εμφανιστεί στον τελικό χρήστη και βάσει ποιου XSL. Έτσι στο τελευταίο αυτό βήμα απλά ο εξυπηρετητής του ΠΙ «καταλαβαίνει» το τι επιπλέον πόρους απαιτεί το ρεύμα XML και τους αποστέλλει μαζί με αυτό στον φυλλομετρητή του χρήστη όπου και γίνεται η κατάλληλη μορφοποίηση.

Η οθόνη-αποτέλεσμα που εμφανίζει μια οποιαδήποτε λειτουργία Function στον επισκέπτη της εφαρμογής περιέχει συνδέσμους που οδηγούν σε άλλες λειτουργίες. Αν η λειτουργία εμφανίζει αποτελέσματα στη δευτερεύουσα γλώσσα, έστω FRENCH (LID=1), τότε όλοι οι σύνδεσμοι που οδηγούν σε άλλες λειτουργίες θα είναι της μορφής URL?LID=1. Άρα όλοι οι σύνδεσμοι που προσδιορίζονται στα «γαλλικά» XSL θα έχουν αυτή τη μορφή και έτσι εξασφαλίζεται ότι και η λειτουργία στην οποία θα μεταβεί ο χρήστης με το πάτημα του συνδέσμου θα είναι στα γαλλικά. Οι μόνοι σύνδεσμοι ενός

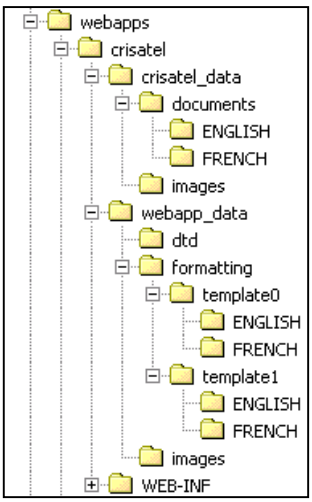
«γαλλικού» XSL που οδηγούν σε άλλες γλώσσες είναι εικόνες σημαιών χωρών. Πάτημα μιας τέτοιας σημαίας π.χ. της αγγλικής, οδηγεί στην ίδια λειτουργία, από την οποία προέκυψε η συγκεκριμένη οθόνη-αποτέλεσμα, αλλά στην αγγλική γλώσσα (Function?LID=0). Έτσι εξασφαλίζεται η άμεση εναλλαγή γλώσσας η οποία είναι σημαντική για τη συνεννόηση συντηρητών διαφορετικών εθνικοτήτων.

4.1.3 Πολλαπλές παρουσιάσεις

Το θέμα της εμφάνισης των αποτελεσμάτων λειτουργιών με διαφορετικά στιλ παρουσίασης είναι συγγενικό με το θέμα της πολυγλωσσίας. Στην πραγματικότητα είναι ακόμα απλούστερο αφού επιτυγχάνεται αποκλειστικά με τη χρήση αρχείων μορφοποίησης XSL. Σε συνδυασμό με το θέμα της πολυγλωσσίας μπορούμε να πούμε ότι αν στο παρόν ΠΣΒΠΙ υποστηρίζονται **n** γλώσσες και **m** στιλ παρουσίασης τότε χρειαζόμαστε **nxm** διαφορετικά XSL αρχεία για κάθε λειτουργία.

Η εναλλαγή μεταξύ των στιλ παρουσιάσεων επιτυγχάνεται με τη βοήθεια της παραμέτρου TID. Η τιμή της παραμέτρου *I.TEMPL[TID]* είναι η ονομασία κάποιου στιλ παρουσίασης.

Η οργάνωση των αρχείων XSL γίνεται ιεραρχικά σε φακέλους κάτω από τον παραμετρικά οριζόμενο φάκελο webapp_formatting_rel_path (βλ. σχήμα 4.1), ο οποίος βρίσκεται κάτω από τον κατάλογο της δικτυακής εφαρμογής. Στο σχήμα 4.2 φαίνεται αυτή η ιεραρχική οργάνωση.



Όπως βλέπουμε για 2 γλώσσες και 2 στιλ παρουσίασης υπάρχουν 4 φάκελοι που περιέχουν XSL αρχεία οι template0/ENGLISH, template0/FRENCH, template1/ENGLISH, template1/FRENCH.

Για τα έγγραφα που περιέχουν περιγραφές γνώσεων ή δελτία συντήρησης έργων τέχνης (σε μορφή κειμένου) απαιτούνται μόνο φάκελοι ανά γλώσσα (κάτω από το φάκελο project_rel_docs_path) αφού τα έγγραφα δεν αλλάζουν όταν αλλάζει το στιλ παρουσίασης...

Σχήμα 4.2 - Οργάνωση φακέλων εφαρμογής

Το αρχείο XSL που μορφοποιεί το ρεύμα XML που παράγεται από τη λειτουργία Function όταν η επιλεγμένη γλώσσα είναι η LID και επιλεγμένη παρουσίαση η TID προσδιορίζεται από τον εξής κώδικα στην αρχή του ρεύματος XML:

<?xml-stylesheet type=\"text/xsl\" href=\"" + I.webapp_formatting_rel_path + "/" + I.TEMPL[TID] + "/" + I.LANG[LID] + "/" + "Function.xsl\"?>

4.2 Υλοποίηση λειτουργιών ανάκτησης δεδομένων

Οι λειτουργίες που υλοποιήθηκαν μπορούν να καταταχθούν σε 4 κατηγορίες βάσει των στιγμιοτύπων των κλάσεων με τα οποία σχετίζονται. Έτσι έχουμε τις λειτουργίες του θησαυρού όπου εστιάζουμε σε όρους του θησαυρού (έννοιες συντήρησης ή μεθόδους), τις λειτουργίες σχετικά με έργα τέχνης και εκτιμήσεις συντηρητών, τις λειτουργίες σχετικά με πολυφασματικές εικόνες και τέλος τις λειτουργίες αναζήτησης. Η κατηγοριοποίηση αυτή δεν βάζει ωστόσο διαχωριστικές γραμμές μεταξύ των διαφόρων λειτουργιών. Στην ουσία η μετάβαση από την μια κατηγορία λειτουργιών στην άλλη είναι ομαλή κατά την πλοήγηση στο σύστημα γεγονός στο οποίο συνδράμει η μοντελοποίηση.

4.2.1 Λειτουργίες θησαυρού

Για τις λειτουργίες θησαυρού χρησιμοποιήθηκε το εξής σενάριο:

Ο χρήστης θα πρέπει να μπορεί να δει γραφικά τις ιεραρχίες του θησαυρού που βρίσκονται κάτω από τις κλάσεις Elaboration, Alteration, Conservation, Material και Method ώστε να έχει μια γενική εικόνα του θησαυρού και μια ιδέα της δόμησης των καθιερωμένων όρων του πεδίου εφαρμογής. Επιλογή συγκεκριμένου όρου θα τον οδηγεί σε φόρμα εμφάνισης πληροφοριών γι' αυτόν. Σ' αυτή τη φόρμα θα υπάρχουν σύνδεσμοι προς σχετικούς κόμβους γνώσης, προς λίστα με σχετικά έργα τέχνης, εκτιμήσεις και πολυφασματικές εικόνες. Στη φόρμα εμφάνισης πληροφοριών για κάποιον κόμβο γνώσης ο χρήστης θα βλέπει πώς ο όρος σχετίζεται με κάποια μέθοδο εξέτασης. Στη λίστα με σχετικά έργα τέχνης θα μπορεί να δει λίστα με όλα τα έργα που σχετίζονται με τον όρο και να εστιάσει στις εκτιμήσεις εκείνες για τα έργα που δείχνουν αυτή τη συσχέτιση.

4.2.1.1 Γραφική απεικόνιση ιεραρχίας όρων

Η γραφική απεικόνιση ιεραρχίας όρων σε φυλλομετρητή του ΠΙ είναι ένα σχετικά καινούριο πρόβλημα. Το πρόβλημα έχει μελετηθεί σε βάθος κυρίως στα πλαίσια δημιουργίας πρόσθετων εργαλείων για απεικόνιση γράφων π.χ. για εφαρμογές πλοήγησης σε θησαυρούς όρων. Εξαιρετική δουλειά αποτελεί το [51] για τον σχεδιασμό σε ιεραρχική μορφή προσανατολισμένων γράφων, η οποία και ενσωματώθηκε στον αναδιφητή του SIS (GAIN [52]). Η δυσκολία αποσύνδεσης του πηγαίου κώδικα από την περιβάλλουσα εφαρμογή μας απέτρεψε από την χρήση του στην παρούσα εργασία.

Η εμπλοκή του φυλλομετρητή του ΠΙ στο πρόβλημα προσθέτει και τη δυσκολία εύρεσης του τρόπου που η ιεραρχία θα εμφανιστεί μέσα σ' αυτόν. Υπήρξαν τέσσερις διαφορετικές σκέψεις για την εμφάνιση της ιεραρχίας:

- ως "indented list"
- με τη χρήση Java applet και στιλ "windows explorer"

- με τη χρήση εξωτερικής εφαρμογής π.χ. GAIN
- με χρήση ειδικού plugin για τον φυλλομετρητή του ΠΙ

Η πρώτη σκέψη απορρίφθηκε διότι απλά δε συνιστά γραφική απεικόνιση και δύσκολα θα προσελκύσει τον χρήστη. Η δεύτερη σκέψη είναι ελκυστική από πλευράς διεπαφής χρήστη αλλά έχει το τεχνικό μειονέκτημα ότι απαιτεί να υλοποιηθεί και να «τρέχει» σε μόνιμη βάση ένας πολυνηματικός εξυπηρετητής RMI στο server tier. Η τρίτη σκέψη είναι ελκυστική από πλευράς του τελικού αποτελέσματος εμφάνισης ιεραρχίας. Ωστόσο μπερδεύει το χρήστη στην πλοήγηση, αφού θα πρέπει να χρησιμοποιήσει τη συγκεκριμένη διεπαφή της εφαρμογής και επιπλέον δεν θα μπορεί πλέον να επιστρέψει «ομαλά» στις σημασιολογικά σχετιζόμενες λειτουργίες του συστήματος κλικάροντας πάνω στην ιεραρχία.

Για τους παραπάνω λόγους καταλήξαμε στην τελευταία σκέψη έχοντας ακούσει για την προτεινόμενη από το W3C γλώσσα παράστασης διανυσματικών γραφικών SVG και την υποστήριξη της από ειδικά plugin (όπως το AdobeSVGView).

Το μοναδικό μειονέκτημα στην περίπτωση αυτή είναι ότι ο φυλλομετρητής του ΠΙ δεν μπορεί να παρουσιάσει το SVG script ως ένα τμήμα ιστοσελίδας παρά μόνο αν αυτό είναι αποθηκευμένο σε αρχείο (embedded script). Αυτό σημαίνει ότι το SVG script με την ιεραρχία δεν μπορεί να παράγεται δυναμικά ως ρεύμα SVG κάθε φορά που ένας χρήστης το ζητήσει. Επίσης αποθήκευση σε προσωρινό αρχείο του SVG script για κάθε αίτηση χρήστη θα οδηγούσε σίγουρα σε πολλά άχρηστα αρχεία στον εξυπηρετητή του ΠΙ.

Με δεδομένο ότι ο θησαυρός των καθιερωμένων όρων ενός πεδίου δεν αλλάζει συχνά θεωρήσαμε ότι δεν είναι μειονέκτημα η διαδικασία παραγωγής της ιεραρχίας γραφικά να γίνεται "offline".

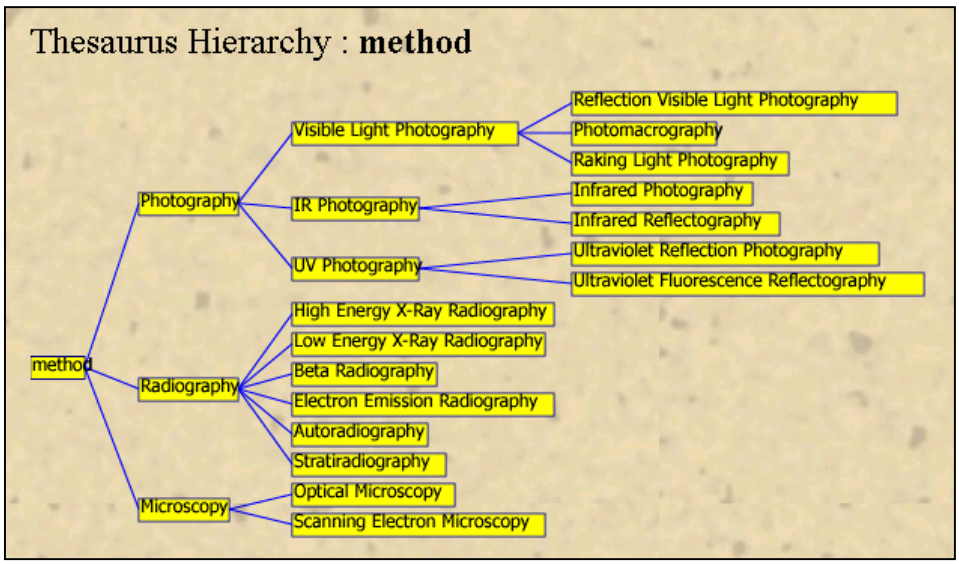
Παραγωγή Ιεραρχίας γραφικά: Η παραγωγή της ιεραρχίας γίνεται με κλήση του Servlet **ProduceHierarchy?LID=[lid]&TID=[tid]&hierarchy=[topterm]** μέσα από φυλλομετρητή. Μετά το "?" είναι προσδιορισμένοι οι παράμετροι που απαιτούνται γι' αυτήν την κλήση. Οι παράμετροι LID, TID απαιτούνται ώστε η κόμβοι-όροι του δέντρου που θα εμφανιστεί να είναι στην κατάλληλη γλώσσα και ώστε να τοποθετηθεί το τελικό SVG script στον κατάλληλο φάκελο του εξυπηρετητή εφαρμογής (βλ. σχήμα 4.2). Η παράμετρος hierarchy προσδιορίζει τον όρο – ρίζα του οποίου το δένδρο στενότερων όρων (NT) θέλουμε να απεικονίσουμε. Προφανώς για να δείξουμε ολόκληρη την ιεραρχία π.χ. των αλλοιώσεων θα πρέπει να έχει τοποθετηθεί ως στιγμιότυπο της κλάσης Alteration ένας κορυφαίος όρος (topterm) π.χ. alteration από τον οποίο δε θα ξεκινάει καμία ιδιότητα *BT*.

Ο αλγόριθμος έχει 3 βασικά στάδια:

- ανάκτηση του δένδρου με κορυφή τον topterm με αναδρομική επερώτηση στο SIS και γέμισμα δυο παράλληλων διανυσμάτων V1, V2 με όρους. Η σχέση που συνδέει τους όρους είναι ότι ο όρος V1[i] έχει BT των V2[i].

- Αναδρομή πάνω στα διανύσματα V1, V2. Σε κάθε βήμα της αναδρομής για τον τρέχοντα κορυφαίο όρο topterm που θα βρίσκεται σίγουρα στο διάνυσμα V2, βρίσκονται όλα τα παιδιά του στο V1(άμεσα στενότεροι όροι). Ο topterm τοποθετείται σε συντεταγμένη y ίση με τη μέση τιμή των συντεταγμένων y_k των παιδιών του ($y = (y_{k1}+ y_{k2}+\ldots+ y_{kn}) / n$) και συντεταγμένη x κατά τι μεγαλύτερη από το άθροισμα της συντεταγμένης x_p του πατέρα του και του μήκους του μεγαλύτερου από τα αδέρφια του πατέρα του ($x = x_p + \max(L_{p\text{-siblings}}) + c$). Κατά την εκτέλεση ενός βήματος της αναδρομής, για τον εκάστοτε όρο topterm, δημιουργούνται αντικείμενα της ειδικά κατασκευασμένης κλάσης SVGOBJ που παριστούν τον κόμβο topterm στην τελική γραφική αναπαράσταση και τις γραμμές από τον κόμβο αυτό προς τα παιδιά του. Τα αντικείμενα αυτού του τύπου τοποθετούνται στο καθολικό διάνυσμα svgV. Συνθήκη τερματισμού της αναδρομής είναι ο όρος να μην έχει παιδιά οπότε η συντεταγμένη αυτού y είναι κατά τι μεγαλύτερη από τη συντεταγμένη $y_{last\text{-}leaf}$ του πιο πρόσφατου φύλλου της ιεραρχίας που συνάντησε ο αλγόριθμος κατά την εκτέλεσή του (υπάρχει καθολική μεταβλητή για το σκοπό αυτό).
- Με διάβασμα του διανύσματος svgV προσπελάζονται όλα τα αντικείμενα SVGOBJ που περιέχουν ό,τι πληροφορίες χρειάζονται για την παράσταση κόμβων –ορθογώνια παραλληλόγραμμα– και γραμμών μεταξύ κόμβων (συντεταγμένες, χρώμα γεμίσματος, γραμματοσειρά, μέγεθος και χρώμα γραμματοσειράς, URL σύνδεσμος κ.α.).

Το Servlet **Thesaurus?LID=[lid]&TID=[tid]&hierarchy=[topterm]** είναι εκείνο που τελικά δείχνει στον χρήστη την ιεραρχία με ρίζα το topterm γραφικά. Αυτό που κάνει στην ουσία αυτό το στατικό Servlet είναι να προσδιορίζει με XML το SVG script (topterm.svg) το οποίο θα ενσωματωθεί στη σελίδα που θα εμφανιστεί στο χρήστη. Χαρακτηριστική είναι η παρακάτω εικόνα:



Σχήμα 4.3 - Η ιεραρχία Method γραφικά (SVG)

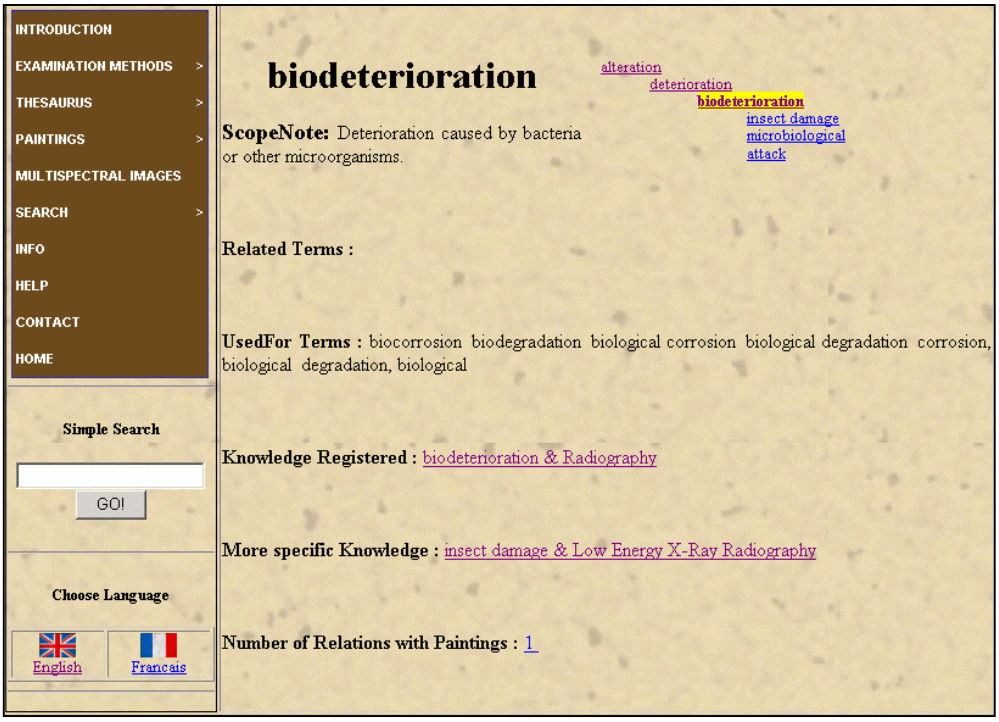
4.2.1.2 Φόρμα εμφάνισης πληροφοριών όρου

Κλικάροντας κάποιον όρο μιας από τις ιεραρχίες που εμφανίζονται από τη λειτουργία της προηγούμενης παραγράφου ο χρήστης μπορεί να δει τις εξής πληροφορίες για τον όρο: το scopenote, τους σχετικούς όρους (related terms), τους μη καθιερωμένους όρους (used for terms), το πλήθος των σχετιζόμενων έργων τέχνης, τους κόμβους γνώσης που σχετίζονται άμεσα μ' αυτόν ή με στενότερους όρους και τέλος όλη την ιεραρχία όρων (BTNTHier) που οδηγεί από τον κορυφαίο όρο της ιεραρχίας έως και τα παιδιά του όρου. Χαρακτηριστική η αρχή του DTD που ορίστηκε γι' αυτή τη λειτουργία:

```
<!ELEMENT DescriptorInfo (Descriptor , ScopeNote , RT* , UF* , BTNTHier , RelPaintings, GeneralKnowledge*, NarrowerKnowledge*)>
```

Το Servlet που υλοποιεί την συγκεκριμένη λειτουργία και μορφοποιεί τα ανακτώμενα από το SIS δεδομένα σε XML (σύμφωνα με το προαναφερθέν DTD) ονομάζεται DescriptorInfo και καλείται ως εξής:

DescriptorInfo?LID=[lid]&TID=[tid]&descriptor=[dscr]



Σχήμα 4.4 - Φόρμα εμφάνισης πληροφοριών όρου

Η λειτουργία ολοκληρώνεται σε τρεις φάσεις. Αρχικά γίνεται ανάκτηση με κατάλληλες επερωτήσεις προς το SIS όλων των πληροφοριών εκτός της BTNTHier. Στη συνέχεια ανακτάται η συγκεκριμένη ιεραρχία με 2 αναδρομικές επερωτήσεις πάνω στη σχέση BT με κατεύθυνση «εμπρός» και απεριόριστο βάθος και με κατεύθυνση «πίσω» (για τους NT) και βάθος 1 (θέλουμε μέχρι και τα παιδιά του όρου). Η πληροφορία που ονομάζουμε BTNTHier είναι πολύ σημαντική για τη συγκεκριμένη λειτουργία γιατί βοηθάει τον χρήστη του συστήματος να συνειδητοποιήσει τη θέση του όρου στην ιεραρχία καθώς και να επισκεφθεί εύκολα και γρήγορα ευρύτερους και άμεσα

στενότερους όρους. Τα δεδομένα και από τις δυο φάσεις αποθηκεύονται σε ένα πίνακα από διανύσματα. Χρειαζόμαστε πίνακα από διανύσματα γιατί πρέπει να ανακτηθεί ένα πλήθος πληροφοριών πολλές από τις οποίες μπορεί να είναι πλειότιμες (π.χ. RT*).

Τέλος πρέπει να πούμε ότι η λειτουργία διαφοροποιείται ελαφρά για την περίπτωση των όρων-μεθόδων και εκτελείται από το Servlet **MethodInfo** που στη θέση του πλήθους σχετιζόμενων έργων τέχνης RelPaintings υπάρχουν οι πληροφορίες πλήθους σχετιζόμενων εκτιμήσεων RelAssessments και πολυφασματικών εικόνων RelImages (είναι ορισμένες στο MethodInfo.dtd).

4.2.1.3 Εμφάνιση έργων τέχνης, εικόνων, εκτιμήσεων σχετικών με όρο

Στο συγκεκριμένο σύνολο λειτουργιών ο χρήστης μεταβαίνει μέσω των πληροφοριών RelPaintings, RelAssessments, RelImages της φόρμας πληροφοριών όρου. Οι πληροφορίες αυτές είναι αριθμητικές και εμφανίζονται ως σύνδεσμοι, για παράδειγμα *Related Paintings : **12*** .

Οι λειτουργίες έχουν υλοποιηθεί με την κατασκευή τριών Servlets που καλούνται ως κάτωθι:

DescriptorPaintings?LID=[lid]&TID=[tid]&descriptor=[descriptor]&page=[page]
MethodAssessments?LID=[lid]&TID=[tid]&method=[method]&page=[page]
MethodImages?LID=[lid]&TID=[tid]&method=[method]&page=[page]

Η πρώτη λειτουργία αφού ανακτήσει τα στιγμιότυπα Man-Made Object (τα λογικά αναγνωριστικά αυτών objID[7]) που σχετίζονται με τον όρο descriptor μέσω των σχέσεων *modification technique*, *consists of material*, *shows alteration*, *conservation modification*, βρίσκει επιπλέον τους τίτλους τους. Τελικά παρουσιάζεται στο χρήστη μια λίστα έργων, σε κάθε γραμμή της οποίας υπάρχουν σύνδεσμοι που οδηγούν στο δελτίο συντήρησης του αντίστοιχου έργου ή ακόμα και εστιάζουν στην συγκεκριμένη εκτίμηση Assessment για το έργο η οποία σχετίζεται με τον όρο descriptor (λειτουργία **DscrAsmtPntg** - δεν θα αναφερθούμε αναλυτικότερα).

Η δεύτερη λειτουργία αφού ανακτήσει όλα τα στιγμιότυπα Assessment (τους κωδικούς τους asmtID) που σχετίζονται με τον όρο method μέσω της σχέσης *used method*, βρίσκει επιπλέον ένα σύνολο πληροφοριών για κάθε Assessment όπως συγγραφέα, θέμα, έργο που αφορά (κωδικός objID), εικόνες που χρησιμοποίησε αλλά και το ίδιο το κείμενο. Ωστόσο μόνο οι τρεις πρώτες πληροφορίες φαίνονται στη λίστα των σχετικών με τη μέθοδο method εκτιμήσεων που εμφανίζει η λειτουργία. Ο χρήστης μπορεί να εστιάσει σε μια συγκεκριμένη εκτίμηση κλικάροντας για κάθε γραμμή της λίστας τον αντίστοιχο σύνδεσμο.

Η τρίτη λειτουργία αφού ανακτήσει όλα τα στιγμιότυπα Multispectral Image (τους κωδικούς τους imgID) που σχετίζονται με τον όρο method μέσω της σχέσης *used image*,

[7] αυτή είναι η περιγραφή της συγκεκριμένης πληροφορίας στο αντίστοιχο DTD

βρίσκει επιπλέον ένα σύνολο πληροφοριών για κάθε εικόνα όπως λεζάντα εικόνας, έργο που αφορά (objID), μέθοδο λήψης, ονομασία εξωτερικού αρχείου, καθώς και αναλυτικά μετα-δεδομένα για τις συνθήκες λήψης. Ο χρήστης μπορεί να εστιάσει σε μια συγκεκριμένη εικόνα κλικάροντας για κάθε γραμμή της λίστας την αντίστοιχη μικρή εικόνα (thumbnail).

Σε όλα τα παραπάνω Servlets εκτός από τις προφανείς παραμέτρους descriptor και method υπάρχει και η παράμετρος page για τη σελιδοποίηση των αποτελεσμάτων όπως θα δούμε σε επόμενη παράγραφο.

Είναι σημαντικό να πούμε σ' αυτό το σημείο ότι οι παράμετροι descriptor και method μπορεί να μην είναι απλοί όροι αλλά μπορεί να έχουν μια ευρύτερη έννοια και να σχετίζονται μ' αυτούς πολλοί άλλοι στενότεροι όροι (σχέση *BT*). Σ' αυτήν την περίπτωση οι λειτουργίες ανακτούν όλα τα έργα τέχνης, εκτιμήσεις, εικόνες και κόμβους γνώσης που σχετίζονται με οποιονδήποτε από τους όρους του δένδρου (BT-NT Tree) με ρίζα descriptor ή method αντίστοιχα. Αυτό είναι ένα πολύ δυνατό χαρακτηριστικό των λειτουργιών για την επίτευξη του οποίου το σημαντικότερο ρόλο παίζει το SIS. Με αυτό το χαρακτηριστικό αξιοποιείται και η πιθανή ταξινόμηση όρων κάτω από την κλάση Base Level Descriptor όπως θα δούμε παρακάτω.

4.2.1.4 Φόρμα εμφάνισης πληροφοριών κόμβων γνώσης

Στη συγκεκριμένη λειτουργία ο χρήστης μεταβαίνει μέσω των πληροφοριών GeneralKnowledge*, NarrowerKnowledge* της φόρμας πληροφοριών όρου. Και οι δυο αυτές πληροφορίες μπορεί να είναι πλειότιμες και κάθε τιμή τους είναι ένα μοναδικό λογικό αναγνωριστικό (gkID). Η λειτουργία εκτελείται με την εξής κλήση Servlet:

KnowledgeInfo?LID=[lid]&TID=[tid]&gkID=[gkID]

Κλικάροντας τον αντίστοιχο σύνδεσμο κόμβου γνώσης στη φόρμα πληροφοριών όρου ο χρήστης μπορεί να δει τις εξής πληροφορίες για το συγκεκριμένο στιγμιότυπο γνώσης: το scopenote (περιγραφή γνώσης), την έννοια συντήρησης (concept) και τη μέθοδο (method) που συσχετίζονται, τους σχετιζόμενους κόμβους γνώσης, παραδείγματα εκτιμήσεων που επιβεβαιώνουν τη γνώση και τέλος όλη την ιεραρχία κόμβων γνώσης που οδηγεί από την πιο γενική γνώση έως και τις αμέσως ειδικότερες εννοιολογικά γνώσεις. Χαρακτηριστική η αρχή του DTD που ορίστηκε γι' αυτή τη λειτουργία:

```
<!ELEMENT KnowledgeInfo (concept , method , ScopeNote , RT* , example*, BTNTHier)>
```

4.2.2 Λειτουργίες σχετικές με έργα τέχνης και εκτιμήσεις ειδικών

Για το συγκεκριμένο σύνολο λειτουργιών χρησιμοποιήθηκε το εξής σενάριο:

Ο χρήστης θα μπορεί να δει ένα χάρτη-πίνακα που θα κατανέμει το σύνολο του περιεχομένου της βάσης (έργα τέχνης) βάσει κάποιων κριτηρίων-κατηγοριών και βάσει

χρονολογίας δημιουργίας. Επιλέγοντας ένα κελί του πίνακα θα μεταβαίνει σε λίστα έργων που ικανοποιούν τα συγκεκριμένα κριτήρια. Από τη λίστα αυτή ακολουθώντας κατάλληλο σύνδεσμο θα βλέπει το δελτίο συντήρησης έργου και από το τελευταίο θα μπορεί να προσπελάσει και τις εκτιμήσεις των απλών χρηστών που έχουν γίνει πάνω σε πολυφασματικές εικόνες του έργου.

4.2.2.1 Χάρτης κατανομής συνόλου έργων

Στη λειτουργία αυτή το σύνολο των στιγμιοτύπων της κλάσης Man-Made Object κατανέμεται βάσει των συσχετίσεων που παρουσιάζουν με έννοιες συντήρησης και βάσει των περιόδων όπου ανήκει χρονικά το χρονικό διάστημα δημιουργίας τους. Σημαντικότερες από τις συσχετίσεις κατανομής είναι οι *support material* και *painting technique* διότι όπως είδαμε στους περιορισμούς ακεραιότητας στη μοντελοποίηση του συστήματος αυτές είναι μονότιμες. Ωστόσο μπορεί η κατανομή να γίνει και βάσει των συσχετίσεων *production technique*, *modification technique*, *shows alteration*, *conservation modification* και *consists of material*.

Το Servlet που υλοποιεί την συγκεκριμένη λειτουργία καλείται ως εξής:

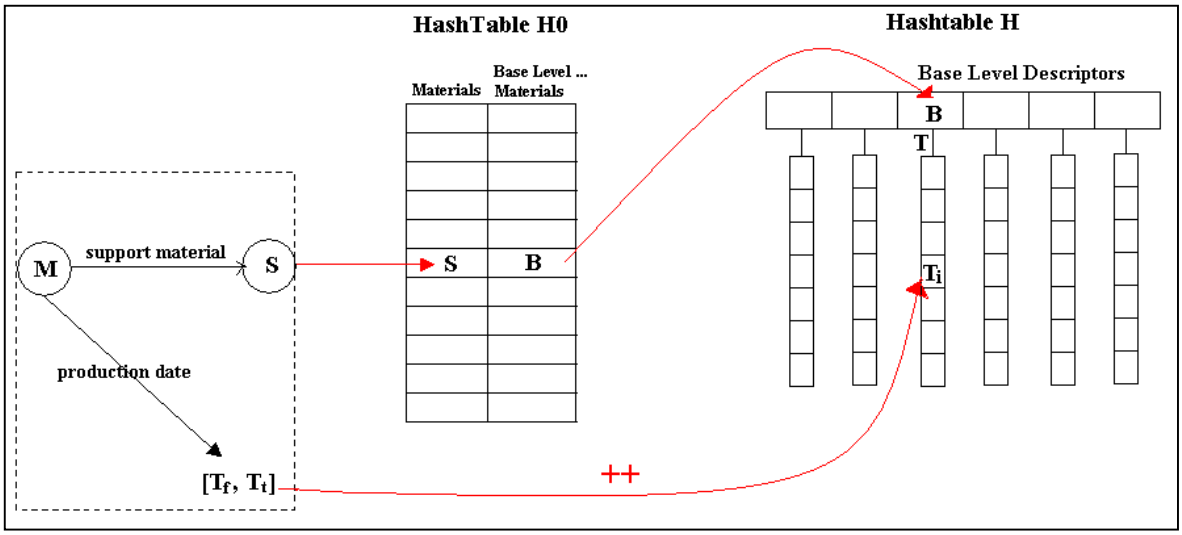
Paintings?LID=[lid]&TID=[tid]&classification=[classification]

Η λειτουργία ολοκληρώνεται σε τέσσερις φάσεις:

1. Αρχικά γίνεται ανάκτηση όλων των χρονικών περιόδων που είναι καταχωρισμένες στο σύστημα και δημιουργούνται αντικείμενα τύπου TPeriod (απλή βοηθητική κλάση) με χαρακτηριστικά έτος αρχής και έτος λήξης περιόδου και πλήθος έργων nobjects που ανήκουν χρονικά στην περίοδο. Όλα τα αντικείμενα αυτά τοποθετούνται σε διάνυσμα T όπου και ταξινομούνται χρονολογικά.
2. Στη συνέχεια ανάλογα με τη συσχέτιση με την οποία θέλουμε να γίνει η ταξινόμηση (classification) ανακτώνται οι κατάλληλοι όροι. Έτσι για την συσχέτιση *support material* ανακτάται το σύνολο M των Materials. Δεν σταματάμε όμως εκεί. Στη συνέχεια ανακτάται το σύνολο B των λεγόμενων βασικών όρων – Base Level Descriptors που διαμερίζουν τις ιεραρχίες των όρων. Η τομή M∩B δίνει τα Materials που συγχρόνως είναι και βασικοί όροι. Με αναδρομική επερώτηση στη συνέχεια στο SIS πάνω στη σχέση *BT* (φορά προς τα πίσω, άπειρο βάθος) ανακτώνται τα Materials που βρίσκονται κάτω από τους βασικούς όρους. Έτσι τελικά είμαστε σε θέση να κατασκευάσουμε ένα Hashtable **H0** με κλειδιά Materials και στοιχεία Base Level Descriptors∩ Materials.
3. Στη συνέχεια γίνεται η ταξινόμηση των έργων βάσει των συσχετίσεων τύπου classification που αυτά έχουν. Αυτό γίνεται ως εξής: δημιουργείται νέο Hashtable **H** με κλειδιά τα στοιχεία του συνόλου M∩B και τιμές διανύσματα T !

(βλ.παραπάνω). Αν κατά την σάρωση των στιγμιοτύπων π.χ. της κατηγορίας *support material* βρεθεί ότι ένα έργο έχει *support material* S και έχει φτιαχτεί κατά το χρονικό διάστημα $[T_f, T_t]$ τότε γίνεται η πράξη: ++H[H0[S]](Ti) όπου Ti η χρονική περίοδος που έχει τη μεγαλύτερη «χρονική επικάλυψη» με το διάστημα $[T_f, T_t]$ (βρίσκεται επαναληπτικά...)[8].

4. Το Hashtable H τελικά είναι εκείνο που περιέχει τα στοιχεία τα οποία θα μορφοποιηθούν τελικά σε ρεύμα XML σύμφωνα με το Paintings.dtd που έχει οριστεί για τη συγκεκριμένη λειτουργία.



Σχήμα 4.5 - Υλοποίηση της λειτουργίας Paintings

Όπως γίνεται κατανοητό η λειτουργία έχει σαν σκοπό να δώσει στο χρήστη την πληροφορία π.χ. ότι «υπάρχουν **10** καταχωρισμένα έργα τέχνης που έχουν σαν βασικό υλικό κατασκευής **wooden material** για την περίοδο **1600-1700**». Χαρακτηριστική είναι η εικόνα 4.6 από την τελική υλοποίηση.

support material / TimePeriod	2000 - 2009	1950 - 1999	1900 - 1949	1800 - 1899	1700 - 1799	1600 - 1699	1500 - 1599	1400 - 1499	1300 - 1399	1200 - 1299	1100 - 1199	1000 - 1099	900 - 999	800 - 899	700 - 799	600 - 699	500 - 599	400 - 499	300 - 399	200 - 299	100 - 199	0 - 99	99 - 1 BC	199 - 100 BC
Academy Board			1																					
panel							1																	
panel, poplar									1															

Σχήμα 4.6 - Αποτέλεσμα της λειτουργίας Paintings

4.2.2.2 Εστίαση σε επιλεγόμενη κατηγορία έργων

Κλικάροντας σε κάποιο κελί του πίνακα που παράγεται από την λειτουργία Paintings ο χρήστης μπορεί να εστιάσει και να δει μια λίστα με τα έργα της αντίστοιχης κατηγορίας και χρονικής περιόδου και συγκεκριμένα τις πληροφορίες κωδικό έργου, τίτλο και κύρια εικόνα (thumbnail). Ο κωδικός έργου και η εικόνα είναι σύνδεσμοι

[8] Βελτιώσεις που μπορούν να γίνουν αναφέρονται στο κεφάλαιο 6

αντίστοιχα για το δελτίο συντήρησης του έργου και την λειτουργία θέασης πληροφοριών εικόνας.

Το Servlet που υλοποιεί την συγκεκριμένη λειτουργία καλείται ως εξής:

ListPaintings?LID=[lid]&TID=[tid]&classification=[classification]&category=[category]&fromyear=[fromyear]&toyear=[toyear]&page=[page]

Η λειτουργία αυτή είναι ουσιαστικά ο συνδετικός κρίκος που οδηγεί από τη σύνοψη της βάσης – λειτουργία Paintings – στο δελτίο συντήρησης έργου, λειτουργία που αναφέρεται στην επόμενη παράγραφο.

4.2.2.3 Εμφάνιση δελτίου συντήρησης έργου

Το δελτίο συντήρησης έργου είναι η συνένωση σε μία οθόνη όλων των βασικών πληροφοριών για το έργο και όλων των επίσημων εκτιμήσεων γι' αυτό από το υπεύθυνο ίδρυμα. Υπάρχουν δυο ακόμα λειτουργίες του συστήματος (DscrAsmtPntg, PaintingUserAssessments) που προσομοιάζουν με το δελτίο συντήρησης (PaintingOfficialReport) όσον αφορά τις παρεχόμενες πληροφορίες. Για το λόγο αυτό έχει γραφτεί κοινό DTD, το PaintingAssessments.dtd, το οποίο εμπεριέχει τα Painting.dtd και Assessment.dtd (εμφωλευμένα-nested DTD), που ορίζουν το σύνολο των πληροφοριών που υπάρχουν για ένα έργο και μια εκτίμηση αντίστοιχα.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!ENTITY % Assessment PUBLIC "Assessment" "Assessment.dtd"> (nested)
%Assessment;
<!ENTITY % Painting PUBLIC "Painting" "Painting.dtd"> (nested)
%Painting;
<!ELEMENT PaintingAssessments (Descriptor? , Painting , Assessment+)>
```

Η λειτουργία καλείται ως εξής:

PaintingOfficialReport?LID=[lid]&TID=[tid]&painting=[pntg]

και ακολουθεί τα παρακάτω βήματα:

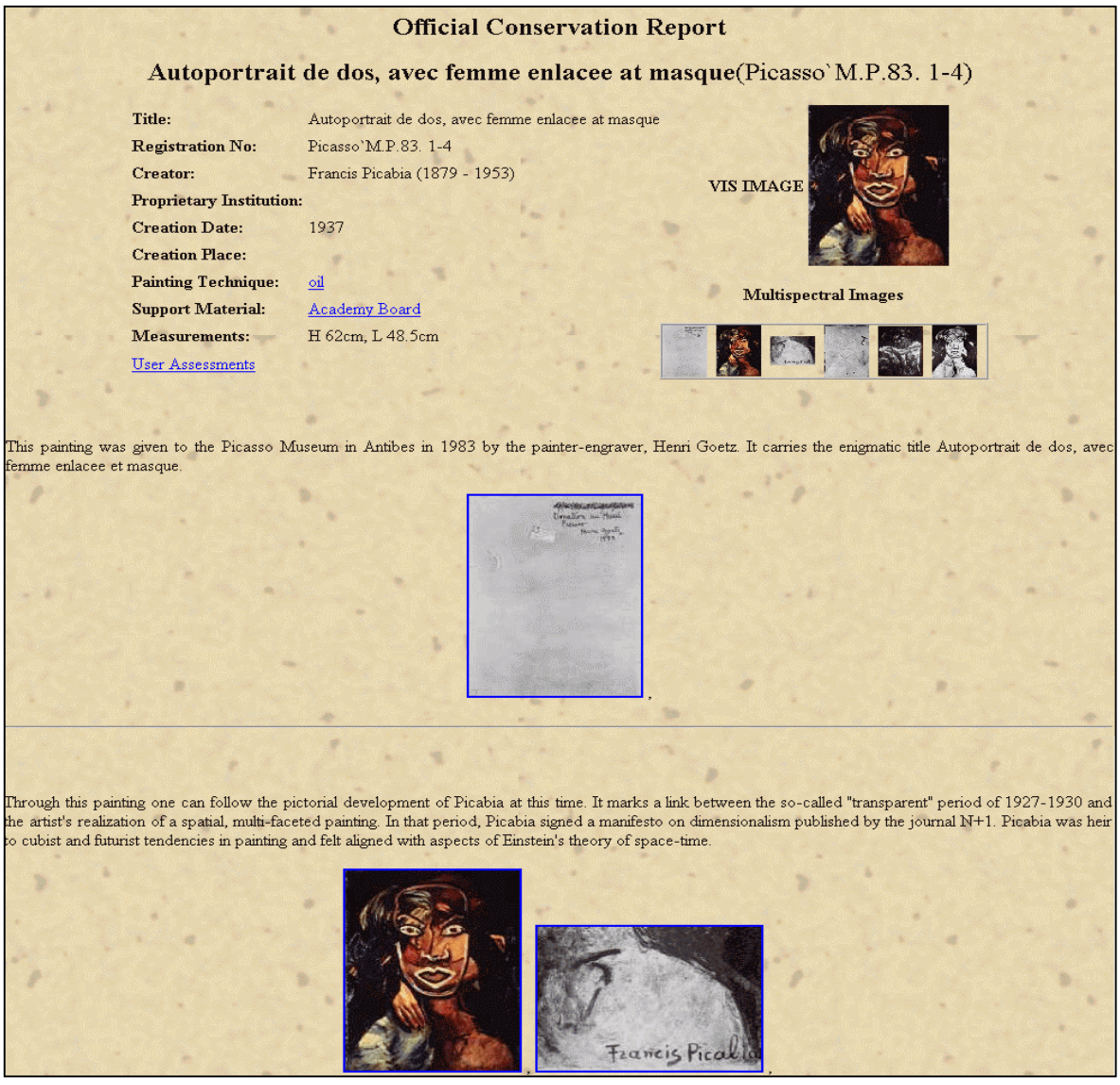
- Αρχικά με κατάλληλη επερώτηση προς το SIS φορτώνονται στον πίνακα από διανύσματα P όλες οι βασικές πληροφορίες για το έργο τέχνης
- Στη συνέχεια στο διάνυσμα Asmts μετά από κατάλληλη επερώτηση αποθηκεύονται οι κωδικοί των Assessments που σχετίζονται με το έργο
- Ακολουθεί αποστολή της επικεφαλίδας του ρεύματος XML και των βασικών πληροφοριών του έργου σε XML μορφή προς τον εξυπηρετητή του ΠΙ
- Για κάθε Assessment του διανύσματος Asmts ανακτώνται όλες οι σχετικές με αυτό πληροφορίες και φορτώνονται σε πίνακα από διανύσματα Α. Στη συνέχεια τα στοιχεία του Α μορφοποιούνται σε XML και στέλνονται προς τον εξυπηρετητή του ΠΙ.

Με κώδικα γίνεται περισσότερο κατανοητό:

```
P=GetPaintingInfo(pntg);
Asmts=GetAssessments(pntg);
```

```
WritePaintingXML(P);
for (int i=0; i<Asmts.size( ); i++) {
A = GetAssessmentInfo(painting,
Asmts.get(i));
WriteAssessmentXML(A);
}
WriteXMLFooter( );
```

Στο εμφανιζόμενο δελτίο συντήρησης υπάρχουν σύνδεσμοι που οδηγούν σε πληροφορίες όρων όπως το βασικό υλικό και η βασική τεχνική κατασκευής και οι μέθοδοι εξέτασης που χρησιμοποιούν τα διάφορα Assessments. Επίσης όλες οι εικόνες είναι σύνδεσμοι προς την αντίστοιχη λειτουργία εμφάνισης πληροφοριών πολυφασματικής εικόνας. Τέλος στην αρχή του δελτίου, και επειδή αυτό μπορεί να είναι αρκετά εκτενές, υπάρχουν thumbnails που δείχνουν μαζεμένες όλες τις πολυφασματικές εικόνες που υπάρχουν για το έργο. Χαρακτηριστική η παρακάτω εικόνα:



Σχήμα 4.7 - Παράδειγμα δελτίου συντήρησης

4.2.2.4 Εμφάνιση εκτιμήσεων χρηστών

Στη λειτουργία αυτή ο χρήστης αποκτά πρόσβαση μέσω του επίσημου δελτίου συντήρησης ενός έργου (σύνδεσμος User Assessments βλ. σχήμα 4.7). Το Servlet που υλοποιεί την συγκεκριμένη λειτουργία καλείται ως εξής:

PaintingUserAssessments?LID=[lid]&TID=[tid]&painting=[pntg]

Όπως προαναφέρθηκε και στην προηγούμενη παράγραφο η λειτουργία είναι παρόμοια με τη λειτουργία του δελτίου συντήρησης. Διαφέρει μόνο στο ότι από τα Assessments που υπάρχουν για το έργο με κωδικό pntg επιλέγει εκείνα που δεν είναι τμήμα (*part of*) εγγράφου (Document), τα οποία σίγουρα θα έχουν εκφρασθεί από επισκέπτες-χρήστες του συστήματος για εικόνες που απεικονίζουν το έργο. Ακριβώς το αντίθετο απ' αυτό που κάνει δηλαδή η λειτουργία του δελτίου συντήρησης.

4.2.3 Λειτουργίες σχετικές με πολυφασματικές εικόνες

Για τις σχετικές με πολυφασματικές εικόνες λειτουργίες χρησιμοποιήθηκε το εξής σενάριο:

Ο χρήστης θα μπορεί να δει ένα πίνακα όπου θα κατανέμεται το σύνολο των καταχωρισμένων εικόνων στους βασικούς όρους που έχουν οριστεί για την ιεραρχία των μεθόδων εξέτασης. Επιλέγοντας συγκεκριμένο βασικό όρο ο χρήστης θα μπορεί να δει λίστα με όλες τις εικόνες που σχετίζονται με αυτόν και τους στενότερους αυτού όρους. Σε κάθε γραμμή της λίστας θα υπάρχουν σύνδεσμοι και thumbnails προς δελτία συντήρησης και αναλυτικές πληροφορίες εικόνας αντίστοιχα.

4.2.3.1 Χάρτης κατανομής πολυφασματικών εικόνων

Στη λειτουργία αυτή το σύνολο των στιγμιοτύπων της κλάσης Multispectral Image κατανέμονται βάσει των μεθόδων λήψης τους. Η κατανομή ωστόσο δεν γίνεται στους ίδιους τους όρους-μεθόδους που συνδέονται άμεσα με τις εικόνες, αλλά σε βασικούς όρους (Base Level Descriptors) που βρίσκονται πιο «ψηλά» στην ιεραρχία μεθόδων. Για την επιτέλεση της λειτουργίας χρησιμοποιείται παρόμοια τεχνική υλοποίησης με αυτή του σχήματος 4.5 (χρήση δυο Hashtables H0, H), λίγο απλούστερη βέβαια αφού αυτή τη φορά δεν ασχολούμαστε με χρονικές περιόδους.

Το Servlet που υλοποιεί την συγκεκριμένη λειτουργία καλείται ως εξής:

MultispectralImages?LID=[lid]&TID=[tid]

Το αποτέλεσμα της λειτουργίας είναι ένας πίνακας στον οποίο δίπλα από κάθε βασικό όρο-μέθοδο υπάρχει ένας αριθμός-σύνδεσμος προς τη λίστα πολυφασματικών εικόνων που σχετίζονται με αυτόν τον όρο και τους στενότερους τους. Η λειτουργία που μας παρουσιάζει αυτή τη λίστα αντιστοιχεί στην εξής κλήση Servlet :

MethodImages?LID=[lid]&TID=[tid]&method=[method]&page=1

Κάθε γραμμή της λίστας έχει σύνδεσμο προς το δελτίο συντήρησης του αντίστοιχου έργου αλλά και εικόνα thumbnail προς τη λειτουργία εμφάνισης πληροφοριών πολυφασματικής εικόνας.

4.2.3.2 Εμφάνιση πληροφοριών για πολυφασματική εικόνα

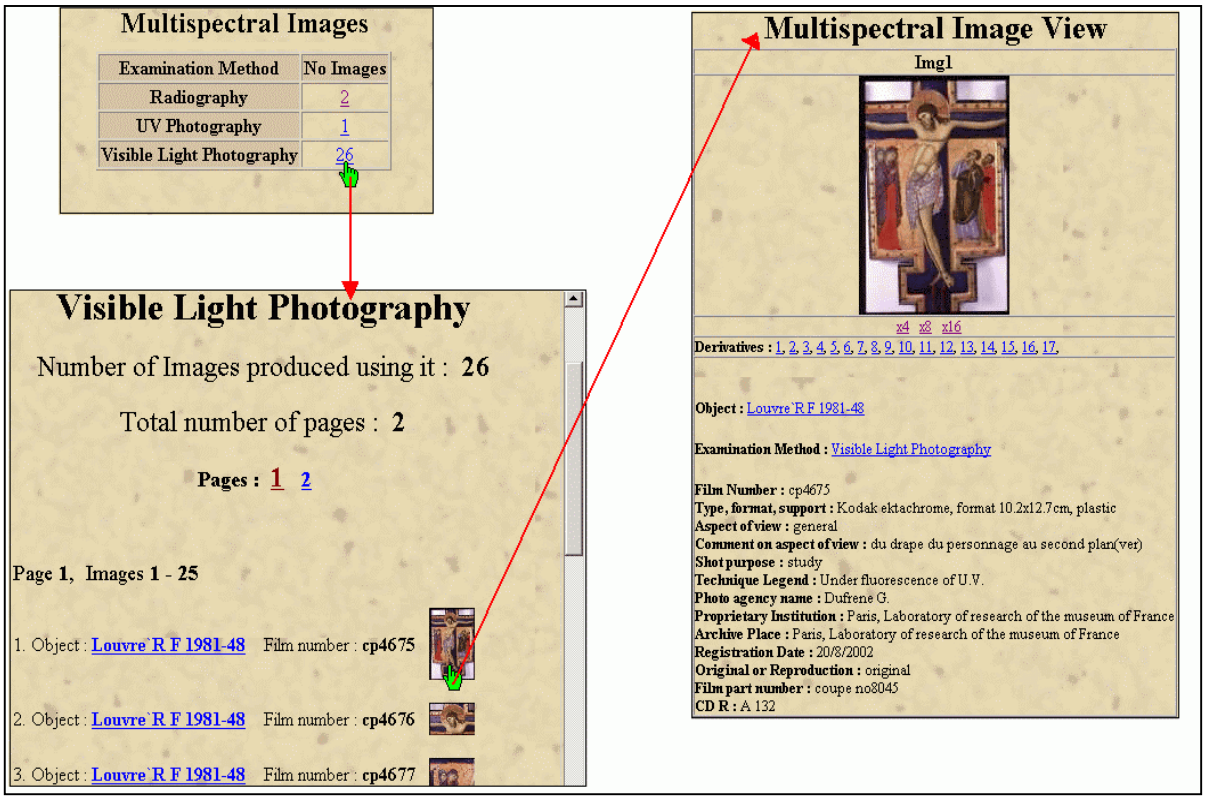
Στη λειτουργία αυτή παρουσιάζεται το σύνολο των πληροφοριών που υπάρχουν για μια πολυφασματική εικόνα: λεζάντα, μέθοδος λήψης, ημερομηνία λήψης, έργο που απεικονίζει, εξωτερικό όνομα αρχείου, παράγωγες εικόνες και αναλυτικές συνθήκες λήψης. Η τελευταία πληροφορία είναι προσβάσιμη μέσω του γνωρίσματος *conditions descriptrion* του μοντέλου και έχει ως τιμή ένα XML string. Οι πληροφορίες που περιέχονται σ' αυτό περιγράφονται από το ειδικά ορισμένο ImgMetadata.dtd.

Το Servlet που υλοποιεί την συγκεκριμένη λειτουργία καλείται ως εξής:

MultispectralImageView?LID=[lid]&TID=[tid]&imgID=[imgID]

Από τη φόρμα με πληροφορίες που εμφανίζει η λειτουργία ο χρήστης μπορεί να μεταβεί στο δελτίο συντήρησης του αντίστοιχου έργου, στη φόρμα εμφάνισης πληροφοριών για την αντίστοιχη μέθοδο εξέτασης, αλλά και στην ίδια τη λειτουργία MultispectralImageView για τις παράγωγες εικόνες (Derivatives, βλ. σχήμα 4.8).

Το πώς συνδέονται οι σχετικές με πολυφασματικές εικόνες λειτουργίες φαίνεται καθαρά στο επόμενο σχήμα 4.8.



Σχήμα 4.8 - Σύνδεση λειτουργιών πολυφασματικών εικόνων

4.2.4 Λειτουργίες αναζήτησης

Έχουν υλοποιηθεί τρεις λειτουργίες αναζήτησης για τα στιγμιότυπα των βασικών κλάσεων του συστήματος Man-Made Object, Assessment και Multispectral Image. Σε κάθε περίπτωση η αναζήτηση μπορεί να γίνει πάνω στις τιμές διαφόρων ιδιοτήτων των στιγμιοτύπων αυτών, ενώ ταυτόχρονα τα αποτελέσματα θα πρέπει να ικανοποιούν υποχρεωτικά όλα τα κριτήρια που έχουν τεθεί (cross-query αναζήτηση). Και στις τρεις λειτουργίες αναζήτησης βασική θέση έχουν οι έννοιες συντήρησης Elaboration, Alteration, Conservation και Material. Για διευκόλυνση των χρηστών οι όροι κάθε ιεραρχίας επιλέγονται από αλφαβητικά ταξινομημένη λίστα. Αυτό σημαίνει ότι για όλες τις λειτουργίες αναζήτησης απαιτείται να έχει γίνει πρώτα ανάκτηση όλων των όρων-εννοιών συντήρησης (και μόνο αυτών) ώστε να εμφανιστούν οι αντίστοιχες φόρμες αναζήτησης.

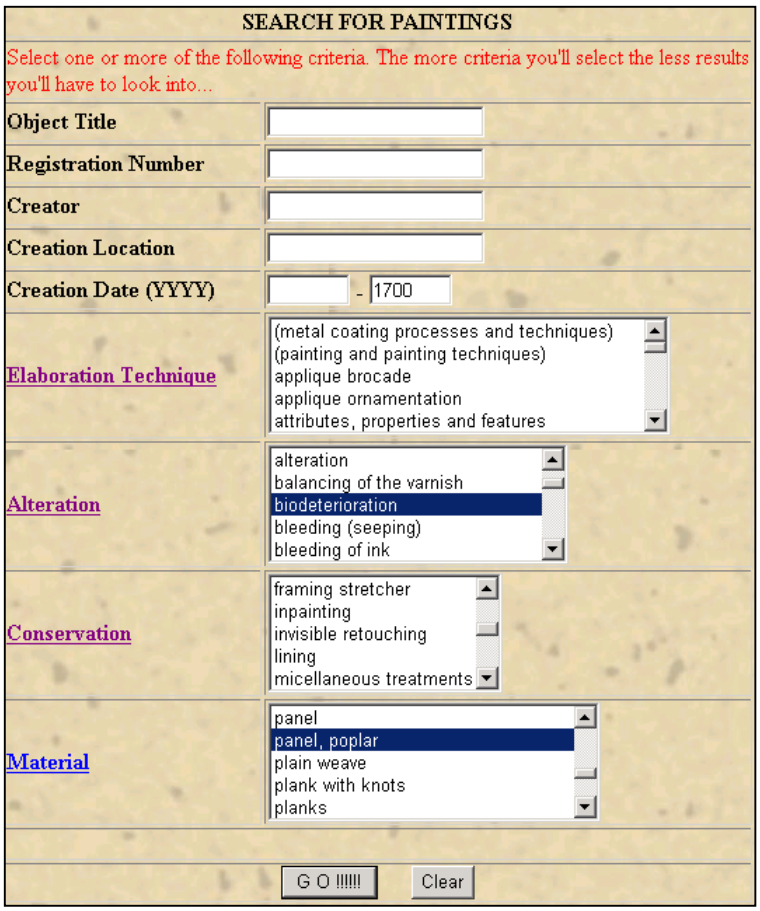
4.2.4.1 Αναζήτηση για έργα τέχνης

Η λειτουργία αυτή πραγματοποιεί αναζήτηση πάνω στις τιμές των εξής πληροφοριών που κρατώνται για τα έργα τέχνης:

- Τίτλο έργου (μερικό ταίριασμα κειμένου π.χ. *Charle*)
- Κωδικό έργου (ακριβές ταίριασμα κειμένου π.χ. Legiond'Honneur`M 834)
- Δημιουργό (μερικό ταίριασμα κειμένου)
- Τόπο δημιουργίας (μερικό ταίριασμα κειμένου)
- Χρονικό διάστημα δημιουργίας (από έτος F - έως έτος T, μπορεί να παραλειφθεί ένα από τα F, T)
- Elaboration, Alteration, Conservation, Material (επιλογή όρων από λίστες)

Η εμφάνιση της φόρμας αναζήτησης γίνεται με την εξής κλήση Servlet:
SearchPaintings?LID=[lid]&TID=[tid]
και χρήση του αντίστοιχου αρχείου SearchPaintings.xsl .Ο χρήστης στη συνέχεια μπορεί να συμπληρώσει όποια και όσα πεδία θέλει με τη γνώση ότι όσο περισσότερα κριτήρια θέσει τόσο λιγότερα αποτελέσματα θα λάβει. Για παράδειγμα μια αναζήτηση που θα μπορούσε να γίνει είναι : «Αναζήτηση των πινάκων ζωγραφικής που κατασκευάσθηκαν πριν από το 1700, έχουν υλικό υποστήριξης panel,poplar και έχουν υποστεί κάποιου είδους biodeterioration». Σ' αυτήν την περίπτωση ο χρήστης πρέπει να επιλέξει από τη λίστα των Materials το panel,poplar, από τη λίστα των Alterations το biodeterioration και τέλος να συμπληρώσει στο δεύτερο «κουτάκι» που προβλέπεται για το Creation Date τον αριθμό 1700 (βλ. σχήμα 4.9).

Αρκεί το πάτημα του κατάλληλου κουμπιού για να αποσταλούν οι επιλογές που έχουν γίνει στη λειτουργία που θα πραγματοποιήσει ουσιαστικά την αναζήτηση.



Σχήμα 4.9 - Η φόρμα αναζήτησης έργων τέχνης

Η αναζήτηση ουσιαστικά πραγματοποιείται με την εξής κλήση Servlet:

SearchPaintingsResults?LID=0&TID=1&Ttitle=[title]&Tregno=[regno]&Tcreator=[creator]&Tlocation=[place]&Tfromyear=[fromyear]&Ttoyear=[toyear]&DElaboration=[elaboration[]]&DAlteration=[alteration[]]&DConservation=[conservation[]]&DMaterial=[material[]]&page=[page]

Οι παράμετροι DElaboration, DAlteration κλπ. είναι πίνακες από παραμέτρους διότι μπορεί να απαιτηθεί σε μια αναζήτηση π.χ. να βρεθούν τα έργα που παρουσιάζουν δυο συγκεκριμένες διαφορετικές αλλοιώσεις. Φυσικά οι παραπάνω παράμετροι της κλήσης είναι όλες οι πιθανές οπότε μπορεί σε μια αναζήτηση να χρησιμοποιηθεί ακόμα και μόνο μία απ' αυτές. Πάντως είναι σαφώς ευκολότερο η λειτουργία αυτή να εκτελεστεί μέσω της φόρμας αναζήτησης και όχι από τη γραμμή διεύθυνσης κάποιου φυλλομετρητή.

Η διαδικασία που εκτελείται με αυτή την κλήση λαμβάνει υπόψη της τον τύπο των δεδομένων πάνω στα οποία γίνεται η αναζήτηση (στιγμιότυπα ορισμένων κλάσεων ή primitive types;), τον τρόπο αναζήτησης (ακριβές ή μερικό ταίριασμα κειμένου κλπ), καθώς και το γεγονός ότι η αναζήτηση είναι τύπου cross-query. Έτσι λέγοντας από την αρχή ότι τα τελικά αποτελέσματα θα είναι αποθηκευμένα στο σύνολο RS_F, έχουμε την εξής ακολουθία βημάτων,:

1. Αν η παράμετρος Tregno έχει τιμή τότε γίνεται αναζήτηση για το στιγμιότυπο Man-Made Object με αυτό τον κωδικό. Αν δε βρεθεί, η αναζήτηση έχει

τελειώσει χωρίς αποτέλεσμα, ενώ αν βρεθεί τότε στο RS_F θα τοποθετηθεί το έργο με κωδικό Tregno και η αναζήτηση θα έχει και πάλι τελειώσει.

2. Για κάθε όρο-παράμετρο D (επαναληπτικά) αρχικά βρίσκονται όλοι οι όροι του υποδένδρου της BT ιεραρχίας με ρίζα τον D, έστω ότι είναι το σύνολο RS_D. Από το σύνολο RS_D μπορούμε με κατάλληλη επερώτηση να βρούμε όλα τα έργα τέχνης RS_P που σχετίζονται με όρους του RS_D μέσω αντίστοιχης σχέσης Dcateg (αν ο D είναι στιγμιότυπο Material τότε Dcateg≡*consists of material*). Για το τελικό σύνολο έργων θα ισχύει $RS_F = RS_F \cap RS_P$ (στην πρώτη επανάληψη $RS_F = RS_P$). Αυτό σημαίνει ότι η διαδικασία υπολογίζει διαδοχικές τομές συνόλων (με κατάλληλες επερωτήσεις στο SIS) για να βρει τα έργα τέχνης που σχετίζονται με όλους τους όρους-παραμέτρους. Σε οποιαδήποτε στιγμή αν ο αριθμός στοιχείων του RS_F μηδενιστεί μετά από μια τομή, αυτό σημαίνει ότι η αναζήτηση συνολικά δεν έχει αποτελέσματα. Το ίδιο ισχύει και για τα βήματα 3 και 4.
3. Αν η παράμετρος Tcreator έχει τιμή γίνεται εύρεση των Actors που «ταιριάζουν» (matching) με την παράμετρο, έστω σύνολο RS_A, από το οποίο βρίσκονται με κατάλληλη επερώτηση και τα έργα RS_P αυτών των Actors. Οπότε υπολογίζεται και πάλι η τομή $RS_F = RS_F \cap RS_P$.
4. Το βήμα 3 επαναλαμβάνεται για την παράμετρο Tlocation.
5. Αν κανένα από τα 2-4 βήματα δεν έχει εκτελεστεί, το σύνολο RS_F δεν θα υπάρχει καν ($RS_F = -1$). Οπότε σε τέτοια περίπτωση για να συνεχιστεί η αναζήτηση για τον τίτλο και την ημερομηνία κατασκευής θα πρέπει να ανακτηθούν όλα τα στιγμιότυπα Man-Made Object και να τοποθετηθούν στο RS_F.
6. Ακολουθεί ανάκτηση τίτλου και ημερομηνίας κατασκευής για τα στοιχεία του συνόλου RS_F. Αν έχουν τιμές τα αντίστοιχα κριτήρια-παράμετροι τότε όποια στοιχεία του RS_F τα ικανοποιούν προστίθενται στο διάνυσμα P των τελικών αποτελεσμάτων, που θα μορφοποιηθούν σε XML.

Στο πλαίσιο φαίνονται τα βήματα με ψευδοκώδικα:

```
RS_F = -1
Αν Tregno έχει τιμή και υπάρχει Man-Made Object με κωδικό Tregno τότε
RS_F.insert(Tregno).
i=0
Για κάθε παράμετρο Descriptor D {
        RS_F = Traverse(D, BT, BACKWARDS)
        RS_P = GetLinkToByCategory(RS_F, Dcateg)
        Αν  i == 0  RS_F= RS_P  αλλιώς RS_F= RS_F∩RS_P
        Αν Cardinality(RS_F) == 0 τότε επέστρεψε RS_F
        i = i + 1
}
```

```
Αν Tcreator έχει τιμή {
        RSA = match(Actor, Tcreator)
        Αν LID <> 0 {
                RSA'=GetLinkToByCategory(RSA, actor_to_LANG)
                RSA = RSA □RSA'
        }
        RSP = GetLinkToByCategory(RSA, "created by")
        RSF= RSF∩RSP
}
Αν Cardinality(RSF) == 0 τότε επέστρεψε RSF
Αν Tlocation έχει τιμή {
        RSL = match(Place, Tlocation)
        RSP = GetLinkToByCategory(RSL, "production place")
        RSF= RSF∩RSP
}
Αν Cardinality(RSF) == 0 τότε επέστρεψε RSF
Αν RSF == -1 {
        RSF = get_instances(Man-Made Object)
}
Για κάθε s στο RSF {
        p = GetInfo(s)   //title, production date
        Αν p ικανοποιεί τα κριτήρια Ttitle, Tfromdate, Ttodate {
                P.add(p)
        }
}
WriteXML(P)
```

Στα αποτελέσματα που τελικά εμφανίζει αυτή η λειτουργία αναζήτησης υπάρχουν οι πληροφορίες κωδικός έργου, τίτλος και χρονικό διάστημα κατασκευής. Ο κωδικός έργου είναι σύνδεσμος που οδηγεί στο δελτίο συντήρησης του αντίστοιχου έργου.

4.2.4.2 Αναζήτηση για εκτιμήσεις

Η λειτουργία αυτή πραγματοποιεί αναζήτηση πάνω στις τιμές των εξής ιδιοτήτων των εκτιμήσεων χρηστών:

- Συγγραφέα (μερικό ταίριασμα κειμένου π.χ. Chatzid*)
- Θέμα (μερικό ταίριασμα κειμένου)
- Τίτλο έργου που αφορά (μερικό ταίριασμα κειμένου)
- Κωδικό έργου (ακριβές ταίριασμα κειμένου π.χ. Louvre`R F 1981-48)
- Method,Elaboration,Alteration,Conservation,Material(επιλογή όρων από λίστες)

Η εμφάνιση της φόρμας αναζήτησης γίνεται με την εξής κλήση Servlet:
SearchAssessments?LID=[lid]&TID=[tid]
και χρήση του αντίστοιχου αρχείου SearchAssessments.xsl .

Η αναζήτηση ουσιαστικά πραγματοποιείται με την εξής κλήση Servlet:

SearchAssessmentsResults?LID=0&TID=1&Ttitle=[title]&Tregno=[regno]&Tauthor=[author]&Tsubject=[subject]&DElaboration=[elaboration[]]&DAlteration=[alteration[]]&DConservation=[conservation[]]&DMaterial=[material[]]&DMethod=[method[]]&page=[page]

Η διαδικασία αναζήτησης είναι ίδιας φιλοσοφίας με την αναζήτηση έργων τέχνης.

4.2.4.3 Αναζήτηση για πολυφασματικές εικόνες

Η τρίτη και τελευταία λειτουργία αναζήτησης γίνεται πάνω στις τιμές των εξής ιδιοτήτων πολυφασματικών εικόνων:

- Εργαστήριο (μερικό ταίριασμα κειμένου π.χ. Louvre*)
- Θέμα (μερικό ταίριασμα κειμένου)
- Τίτλο έργου που αφορά (μερικό ταίριασμα κειμένου)
- Κωδικό έργου (ακριβές ταίριασμα κειμένου π.χ. Louvre`R F 1981-48)
- Method,Elaboration,Alteration,Conservation,Material(επιλογή όρων από λίστες)

Η εμφάνιση της φόρμας αναζήτησης γίνεται με την εξής κλήση Servlet:

SearchAssessments?LID=[lid]&TID=[tid]

και χρήση του αντίστοιχου αρχείου SearchAssessments.xsl .

Η αναζήτηση ουσιαστικά πραγματοποιείται με την εξής κλήση Servlet:

SearchAssessmentsResults?LID=0&TID=1&Ttitle=[title]&Tregno=[regno]&Tauthor=[author]&Tsubject=[subject]&DElaboration=[elaboration[]]&DAlteration=[alteration[]]&DConservation=[conservation[]]&DMaterial=[material[]]&DMethod=[method[]]&page=[page]

4.2.4.4 Εμφάνιση και σελιδοποίηση αποτελεσμάτων

Σε πολλές από τις λειτουργίες που υλοποιήθηκαν κι έχουν ήδη παρουσιαστεί εμφανίζεται η παράμετρος page. Η συγκεκριμένη παράμετρος προσδιορίζει τη σελίδα εκείνη από τα αποτελέσματα την οποία θέλουμε να εμφανίσει η αντίστοιχη λειτουργία. Μια σελίδα αποτελεσμάτων έχει τόσα αποτελέσματα όσα έχουν οριστεί στο αντίστοιχο αρχείο παραμέτρων config.xml για τον τύπο λειτουργίας. Οι λειτουργίες που στα αποτελέσματά τους περιλαμβάνουν εικόνες χρησιμοποιούν τις παραμέτρους της εφαρμογής *webapp_img_resppg* και *webapp_img_pppgset*, ενώ όσες δείχνουν μόνο κείμενο τις *webapp_search_resppg*, *webapp_search_pppgset*, για τη σελιδοποίηση των αποτελεσμάτων .

Τα DTD όλων των λειτουργιών που εμφανίζουν λίστα αποτελεσμάτων περιέχουν με μικρές παραλλαγές (ανάλογα με τον τύπο των αποτελεσμάτων) το στοιχείο PageInfo:

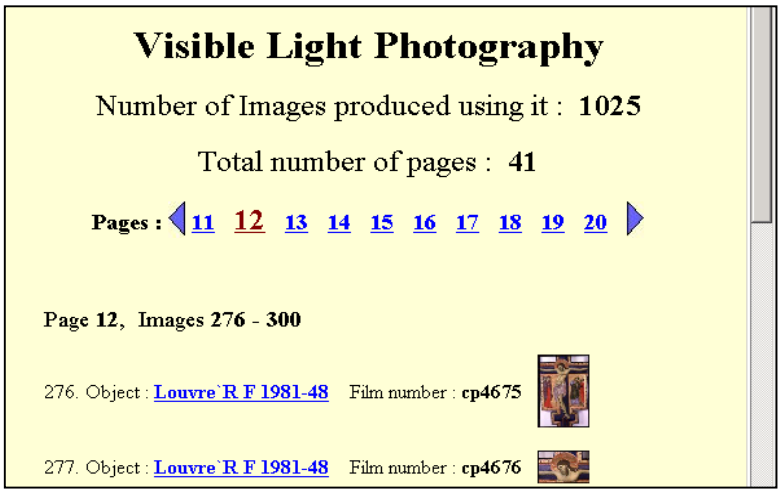
<!ELEMENT PageInfo (frompntg , topntg , nopntgs , page , frompage , topage , nopages , pageset , nopagesets)>

Αυτό το στοιχείο χρησιμοποιούν και τα αρχεία μορφοποίησης XSL για να κάνουν τη σελιδοποίηση αποτελεσμάτων στον φυλλομετρητή. Αν P το σύνολο των αποτελεσμάτων που πρέπει να δείξει μια λειτουργία τότε οι τιμές που παίρνουν τα υποστοιχεία του PageInfo υπολογίζονται ως εξής :

```
pppgset=I.webapp_img_pppgset;
resppg=I.webapp_search_resppg;
nopntgs=P.size();
if (P!=null && P.size()>0) {
        pageset=(page-1)/pppgset + 1;
        nopages = ((nopntgs-1)/resppg) + 1;
        nopagesets = ((nopntgs-1)/(pppgset*resppg)) + 1;
        frompage=(page-1)-(page-1)%pppgset + 1;
        if (frompage+pppgset-1 <= nopages) {
                topage=frompage+pppgset-1;
        } else {
                topage=nopages;
        }
        frompntg=(page-1)*resppg + 1;
        if (page*resppg <= nopntgs) {
                topntg=page*resppg;
        } else {
                topntg=nopntgs;
        }
} else {
        page=0;
}
```

Πρέπει να πούμε ότι το ρεύμα XML που παράγεται από αυτές τις λειτουργίες περιέχει μόνο τα αποτελέσματα εκείνα που ανήκουν στη σελίδα που υποδηλώνει η παράμετρος page. Αυτό συμβαίνει για λόγους ταχύτητας, αφού έτσι η μορφοποίηση γίνεται σε πολύ μικρό χρόνο, αλλά κι επειδή σε πολλές περιπτώσεις περιορίζουμε χρονοβόρες λειτουργίες ανάκτησης δεδομένων, που θα έπρεπε να γίνουν πάνω σε όλα τα στοιχεία του P, στο σύνολο P[frompntg, topntg].

Παρακάτω φαίνεται η σελιδοποίηση των αποτελεσμάτων για τη λειτουργία MethodImage που έχουμε αναφέρει σε προηγούμενη παράγραφο:



Σχήμα 4.10 - Σελιδοποίηση αποτελεσμάτων

4.2.5 Λειτουργία προσθήκης σελίδων με στατικό περιεχόμενο

Σε ένα ΠΣΒΠΙ με εκπαιδευτικό χαρακτήρα εκτός από τα δυναμικά δεδομένα που ανακτώνται με τη χρήση κατάλληλων λειτουργιών πολλές φορές είναι χρήσιμες και στατικές σελίδες γενικών πληροφοριών. Στο παρόν σύστημα τέτοιες σελίδες θα μπορούσε να περιέχουν :

- Γενικές πληροφορίες για το σύστημα
- Βοήθεια για τη χρήση του συστήματος
- Πληροφορίες για τις μεθόδους εξέτασης
- Πληροφορίες για την κατασκευή του θησαυρού

Οι διάφορες στατικές σελίδες πρέπει να μπορούν να είναι προσβάσιμες είτε μέσω του μενού επιλογών του συστήματος είτε από άλλες στατικές σελίδες. Έτσι είναι δυνατό να δημιουργηθεί ένα «δίκτυο» στατικών ιστοσελίδων προσαρτημένων στο σύστημα.

Για την προσάρτηση των στατικών ιστοσελίδων έχει κατασκευαστεί μια λειτουργία η οποία καλείται ως εξής:

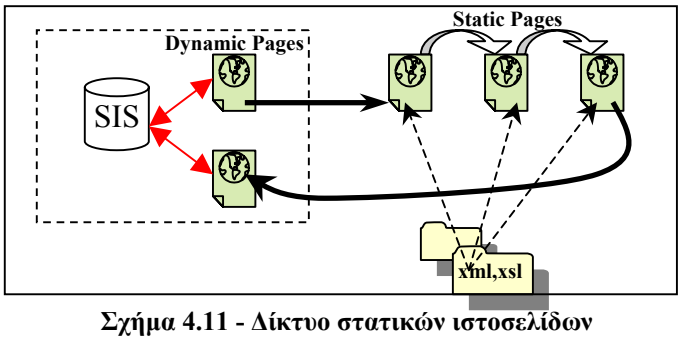
StaticPage?LID=[lid]&TID=[tid]&file=[file]

Η λειτουργία αυτή κάνει το εξής:

Αναζητά στο φάκελο της εφαρμογής που προσδιορίζεται από τις παραμέτρους LID, TID (βλ. §4.1.3) τα αρχεία file.xml και file.xsl. Στο αρχείο file.xml περιέχονται όλες οι πληροφορίες που θέλουμε να εμφανίσει η στατική λειτουργία, ενώ το αρχείο file.xsl είναι εκείνο που αναλαμβάνει τη μορφοποίηση τους. Το αρχείο file.xml δεν περιέχει την πληροφορία για το πού βρίσκεται το αρχείο μορφοποίησης γιατί αυτή μεταβάλλεται συναρτήσει των παραμέτρων LID, TID. Η λειτουργία διαβάζει το αρχείο file.xml γραμμή γραμμή και το στέλνει ως ρεύμα XML προς τον εξυπηρετητή του ΠΙ. Στην αρχή και αφού εντοπίσει την επικεφαλίδα "<?xml version" στέλνει μια επιπλέον γραμμή στο ρεύμα με τη διαδρομή για το αρχείο μορφοποίησης :

<?xml-stylesheet type=\"text/xsl\" href=\"" + I.webapp_formatting_rel_path + "/" + I.TEMPL[TID] + "/" + I.LANG[LID] + "/" + "file.xsl\"?>

Για την μετάβαση από μια στατική ιστοσελίδα sp1 σε μια άλλη sp2 αρκεί στο sp1.xsl να υπάρχει σύνδεσμος της μορφής: **…StaticPage?LID=[lid]&TID=[tid]&file=sp2** . Μετάβαση μπορεί να γίνει και προς δυναμική λειτουργία του συστήματος περιλαμβάνοντας στο sp1.xsl τον κατάλληλο σύνδεσμο για την αντίστοιχη κλήση Servlet.



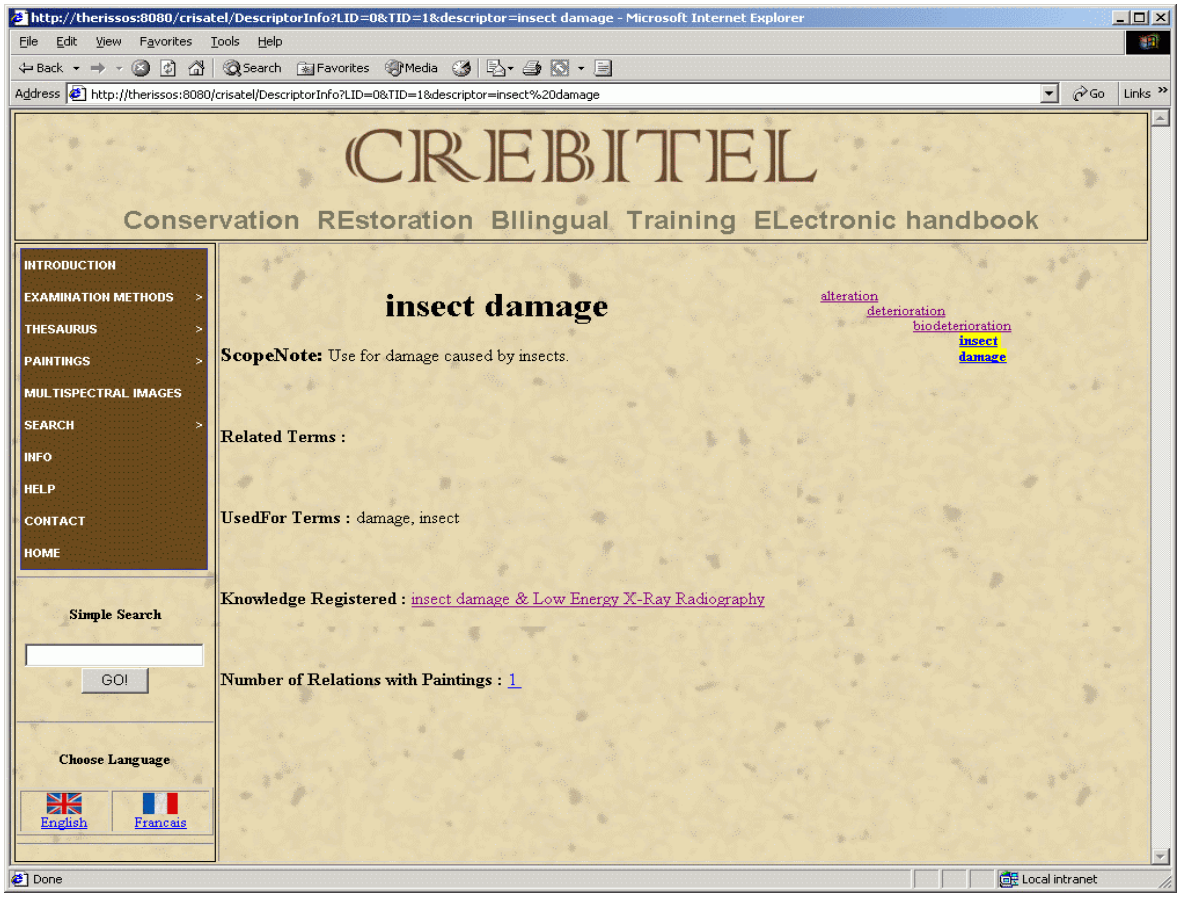
Σχήμα 4.11 - Δίκτυο στατικών ιστοσελίδων

4.3 Σχεδίαση Διεπαφής Χρήστη

Για το παρόν σύστημα σχεδιάστηκαν 2 στιλ παρουσίασης που έχουν την ίδια γενική φόρμα όσον αφορά την εμφάνιση των δεδομένων και διαφέρουν μόνο στο φόντο και στην χρήση banner (το ένα έχει ενώ το άλλο όχι).

Η σχεδίαση του συστήματος καθιστά άσκοπη τη χρήση πλαισίων (frames) κι αυτό διότι απαιτείται οτιδήποτε εμφανίζεται να ελέγχεται απόλυτα και δυναμικά από την εκάστοτε λειτουργία βάσει των παραμέτρων LID και TID. Η χρήση πλαισίων είναι χρήσιμη κυρίως στις περιπτώσεις όπου κάποια στοιχεία σελίδων ενός δικτυακού τόπου δε μεταβάλλονται, οπότε φορτώνονται μία και μόνο φορά. Αυτό δεν ισχύει για το παρόν σύστημα. Έτσι για τη διαίρεση των σελίδων σε περιοχές χρησιμοποιήθηκε το στοιχείο table της HTML μέσω των αρχείων μορφοποίησης XSL.

Στο 2^{ο} στιλ παρουσίασης (template1), το οποίο και θεωρούμε καλύτερο, η κάθε σελίδα διαιρείται σε 3 τμήματα. Το πρώτο τμήμα είναι μια λεπτή οριζόντια λωρίδα στην κορυφή για την τοποθέτηση banner. Το δεύτερο τμήμα είναι μια λεπτή κατακόρυφη λωρίδα στα αριστερά της εναπομείνουσας σελίδας για την τοποθέτηση των λειτουργιών και το τρίτο είναι η μεγάλη περιοχή της υπόλοιπης σελίδας για την εμφάνιση των αποτελεσμάτων λειτουργιών (βλ. σχήμα. 4.12).



Σχήμα 4.12 - Η διεπαφή χρήστη

Θεωρήθηκε περισσότερο λειτουργικό το μενού με τις λειτουργίες να είναι στ' αριστερά της οθόνης και να είναι σχετικά συμπαγές (με popup υπομενού) και γι' αυτό χρησιμοποιήσαμε το παραμετρικό εργαλείο δημιουργίας μενού mapbMenu που παρουσιάστηκε στο κεφάλαιο 2. Κοντά στο μενού λειτουργιών κρίθηκε σκόπιμο να υπάρχουν και μικρές σημαίες για την άμεση εναλλαγή γλώσσας.

Τρία ακόμα είναι τα σημεία που αξίζει να αναφέρουμε σχετικά με τη σχεδίαση της διεπαφής χρήστη:

1. Στην εμφάνιση ιεραρχιών γραφικά, το δένδρο της ιεραρχίας εκτείνεται από αριστερά προς τα δεξιά κι όχι από πάνω προς τα κάτω γεγονός που κάνει την εικονιζόμενη ιεραρχία αρκετά πιο συμπαγή (διαφορετικά το μήκος των όρων θα την έκανε τεράστια)
2. Η σελιδοποίηση των αποτελεσμάτων είναι σημαντική για το χρήστη που δε χρειάζεται να περιμένει πολύ χρόνο για την εμφάνιση όλων των αποτελεσμάτων π.χ. μιας αναζήτησης. Σ' αυτό βασικό ρόλο παίζει και η εύκολη πλοήγηση στις σελίδες των αποτελεσμάτων (βλ. σχήμα 4.10). Οι λειτουργίες που εμφανίζουν λίστες από αποτελέσματα με σελιδοποίηση είναι απαραίτητες γιατί απαλάσσουν άλλες λειτουργίες από αυτή τη δουλειά. Σκεφτείτε στη λειτουργία DescriptorInfo αντί π.χ. της πληροφορίας "Number of relations with paintings : 3000" να δείχναμε τους 3000 κωδικούς των σχετιζόμενων paintings! Θα είχαμε αρκετή καθυστέρηση για να δείξουμε στο χρήστη τη σελίδα, ενώ μπορεί αυτός να ενδιαφέρονταν να δει μόνο το scopenote του όρου...
3. Στη λειτουργία DescriptorInfo η εμφάνιση της θέσης του όρου στην ιεραρχία γίνεται στην πάνω δεξιά γωνία (βλ.σχήμα 4.12) και αυτό είναι το πιο σωστό σημείο, αφού πρέπει αφενός μεν να φαίνεται αμέσως, κι αφετέρου να αφήνει τις υπόλοιπες σχετικές με τον όρο πληροφορίες να έχουν κεντρική θέση στη σελίδα.

4.4 Επιδόσεις Λειτουργιών

Η μέτρηση των επιδόσεων των λειτουργιών του συστήματος δεν μπορεί να γίνει εκ των προτέρων (a priory) βρίσκοντας την πολυπλοκότητά τους, λόγω της κατά κόρον χρήσης ειδικών επερωτήσεων προς τον εξυπηρετητή του SIS, των οποίων δεν γνωρίζουμε την απόδοση[9]. Για το λόγο αυτό αποφασίστηκε η μέτρηση των επιδόσεων να γίνει εκ των υστέρων (a posteriori) με τη χρήση ειδικού μετρητικού πειράματος.

Για το πείραμα ακολουθήθηκε η εξής σειρά βημάτων:

[9] Γνωρίζουμε ωστόσο εμπειρικά ότι έχουν πολύ καλές επιδόσεις

1. Κατασκευάσθηκε πρόγραμμα για την παραγωγή επιλεγόμενου πλήθους εικονικών δεδομένων (σε γλώσσα Telos) σύμφωνων με το μοντέλο που παρουσιάστηκε στο κεφάλαιο 3. Το πρόγραμμα δέχεται σαν είσοδο τον αριθμό των έργων τέχνης που θέλουμε να δημιουργήσουμε και κατασκευάζει για καθένα απ' αυτά τυχαίο αριθμό (μέσα σε κάποια όρια) σχετιζόμενων εκτιμήσεων, εικόνων και συσχετίσεων με όρους του θησαυρού.
2. Με χρήση του παραγόμενου κώδικα Telos και του αντίστοιχου parser που διαθέτει το SIS δημιουργήθηκαν διαδοχικά στιγμιότυπα βάσεων με 150, 250, 500, 750, 1000 και 1500 έργα τέχνης.
3. Σε όλα τα Servlets που κατασκευάσθηκαν προστέθηκαν 3 - 4 γραμμές κώδικα για την μέτρηση της διάρκειας εκτέλεσής τους (σε milliseconds). Η εκτύπωση της διάρκειας γίνεται στο ρεύμα stdout, ανεξάρτητα δηλαδή από τα αποτελέσματα της λειτουργίας που πηγαίνουν προς τον εξυπηρετητή του ΠΙ.
4. Για όλα τα στιγμιότυπα βάσεων εκτελέσθηκε ένα συγκεκριμένο σενάριο χρήσης της δικτυακής εφαρμογής του συστήματος επί 5 φορές ώστε να υπάρξει εξομάλυνση των αποτελεσμάτων με τον υπολογισμό μέσων όρων . Η εκτέλεση του σεναρίου σε κάθε περίπτωση έγινε αμέσως μετά από εκκίνηση του εξυπηρετητή εφαρμογής και του εξυπηρετητή του SIS για να βρεθούν οι χειρότερες περιπτώσεις (χωρίς την εκμετάλλευση του caching). Οι παράμετροι που χρησιμοποιήθηκαν για κάθε λειτουργία ήταν τυπικές παράμετροι χρήσης, πράγμα που σημαίνει ότι οι ίδιες λειτουργίες μπορεί να δώσουν και καλύτερα αλλά και χειρότερα αποτελέσματα.
5. Το σενάριο εκτελέσθηκε και για 2 «ταυτόχρονους» χρήστες. Στην περίπτωση αυτή κάθε λειτουργία του σεναρίου εκτελούνταν ταυτόχρονα και από τους 2 χρήστες και υπολογίζονταν ο μέσος όρος της διάρκειας της λειτουργίας. Σημειωτέον ότι ο ταυτοχρονισμός στην εκτέλεση λειτουργιών δεν είναι πολύ συχνός παρά μόνο σε συστήματα που προσπελάζονται από πολλούς χρήστες.

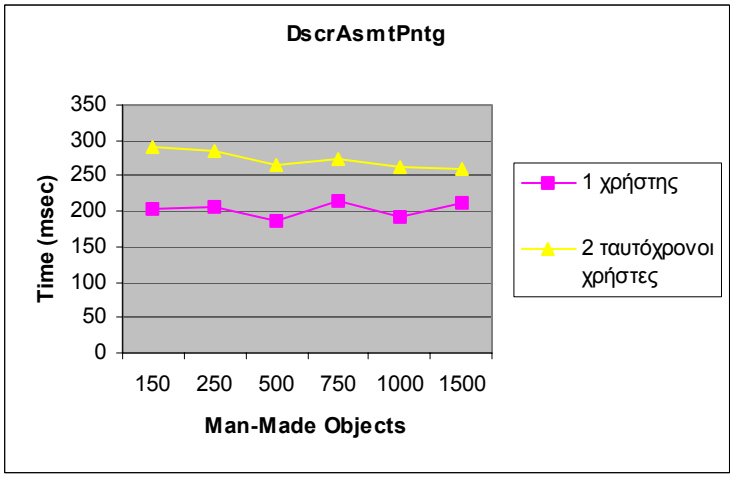
Πρέπει να επισημάνουμε ότι τα πειράματα μετρήσεων δείχνουν απλά μια γενική εικόνα των τάσεων των διαφόρων λειτουργιών όσον αφορά χρονική καθυστέρηση. Υπάρχουν περιπτώσεις αποκλίσεων στα αποτελέσματα απ' αυτό που κάποιος θα περίμενε και αυτό οφείλεται κυρίως στο ότι οι λειτουργίες είναι σχετικά σύντομες κι εξωγενείς παράγοντες (συνθήκες που επικρατούν στο μηχάνημα εκτέλεσης) εύκολα επηρεάζουν τις καθυστερήσεις. Στον πίνακα 4.1 φαίνονται οι μετρήσεις που έγιναν για 12 επιλεγμένες λειτουργίες. Παρατηρούμε ότι οι λειτουργίες που επικεντρώνονται σε ένα αντικείμενο της βάσης και ανακτούν γι' αυτό πληροφορίες μονότιμων ιδιοτήτων ή ιδιοτήτων με μικρό πλήθος τιμών[10] εμφανίζουν σχετική σταθερότητα στα αποτελέσματα των μετρήσεων. Τέτοιες λειτουργίες είναι οι KnowledgeInfo, AssessmentInfo, MultispectralImageView,

[10] ανεξάρτητο από το πλήθος των βασικών αντικειμένων Man-Made Object,Assessment,Multispectral Image

PaintingOfficialReport και DscrAsmtPntg (βλ. σχήμα 4.13). Παρόμοια σταθερότητα επίδοσης εμφανίζουν και οι λειτουργίες εμφάνισης των φορμών αναζήτησης, όπως η SearchAssessments, οι οποίες απλά ανακτούν τους όρους των ιεραρχιών του θησαυρού, το πλήθος των οποίων δε μεταβάλλεται.

Λειτουργίες / Πλήθος έργων/εκτιμήσεων/εικόνων M / A / I (1 χρήστης) (2 χρήστες)	DescriptorInfo	DescriptorPaintings	DscrAsmtPntg	KnowledgeInfo	AssessmentInfo	MultispectralImages	MultispectralImageView	PaintingOfficialReport	Paintings	ListPaintings	SearchAssessments	SearchAssessmentsResults
	Διάρκεια εκτέλεσης λειτουργιών για 1 και 2 χρήστες σε msec[11]											
150/680/1056	171	40	203	70	78	241	40	215	481	190	70	140
	195	70	291	91	105	351	44	285	755	236	88	155
250/1130/1517	181	110	206	70	80	270	30	195	621	161	80	147
	245	163	285	81	110	385	32	282	915	245	94	225
500/2120/3223	225	121	185	65	90	371	31	145	942	211	81	221
	310	165	265	85	125	485	32	228	1482	330	107	285
750/3360/4875	221	261	215	63	100	471	42	171	1272	250	80	240
	302	242	275	80	120	571	37	265	2043	366	105	301
1000/4560/6689	258	245	192	65	95	601	39	141	1702	300	75	271
	350	305	262	88	110	651	45	206	2819	406	91	406
1500/6735/9462	285	295	212	70	100	811	41	181	2634	370	80	320
	399	378	259	92	121	876	44	263	4296	539	101	436

Πίνακας 4.1 - Επιδόσεις λειτουργιών συστήματος σε εικονικές συνθήκες



Σχήμα 4.13 - Σταθερότητα επίδοσης της λειτουργίας DscrAsmtPntg

[11] Οι μετρήσεις έγιναν σε μηχάνημα Celeron 450MHz, 128K cache, 192MB RAM

Οι λειτουργίες DescriptorInfo και DescriptorPaintings επικεντρώνονται σε ένα αντικείμενο της βάσης και ανακτούν γι' αυτό εκτός των άλλων και πληροφορίες πλειότιμων ιδιοτήτων, το πλήθος των οποίων εξαρτάται εμμέσως από το πλήθος των Man-Made Objects της βάσης. Έτσι στην περίπτωση π.χ. που το πλήθος των πινάκων ζωγραφικής αυξάνεται, είναι πιθανό να αυξηθεί – πιθανά λίγο – και το πλήθος των αντικειμένων που έχουν ως βασική τεχνική ζωγραφικής tempera. Οπότε η λειτουργία DescriptorInfo θα καθυστερεί περισσότερο για τον όρο tempera, αφού θα πρέπει να βρει ένα μεγαλύτερο σύνολο σχετικών πινάκων ζωγραφικής και να παρουσιάσει το πλήθος του ως σύνδεσμο προς τη λειτουργία DescriptorPaintings (βλ. σχήμα 4.14).

Παρόμοια πράγματα ισχύουν και για τις λειτουργίες ListPaintings και SearchAssessmentResults που το πλήθος των αποτελεσμάτων που θα εμφανίσουν (όποιες κι αν είναι οι παράμετροι κλήσης τους) εξαρτάται έμμεσα από το πλήθος των Man-Made Objects.

Τέλος οι λειτουργίες Paintings και MultispectralImages εξαρτώνται άμεσα από το πλήθος των Man-Made Objects τα οποία και τις εικόνες των οποίων κατανέμουν. Φυσικό λοιπόν είναι να υπάρχει τουλάχιστο γραμμική σχέση Ω(N) μεταξύ του πλήθους των Man-Made Objects και του χρόνου εκτέλεσης αυτών των λειτουργιών. Βλέπουμε ωστόσο στα σχήματα 4.15 και 4.16 ότι για τις λειτουργίες αυτές προσεγγίζεται τετραγωνική πλοκή. Αυτό κατά πάσα πιθανότητα οφείλεται στην απόδοση της δομής HashTable της Java που χρησιμοποιήθηκε για την επιτέλεσή τους. Το φαινόμενο εμφανίζεται οξυμένο στη λειτουργία Paintings, πράγμα που αποδίδεται στην ανάγκη κατηγοριοποίησης βάσει χρονικών περιόδων, η οποία απαιτεί κάποια επαναληπτική διαδικασία όπως είδαμε στην §4.2.2.1.

Αξίζει να σημειώσουμε δυο-τρία πράγματα σχετικά με τους χρόνους που μετρήσαμε για τις διάφορες λειτουργίες. Παρότι το πλήθος των έργων κυμαίνεται από 150 έως 1500 ο χρόνος εκτέλεσης των λειτουργιών οι οποίες έχουν με αυτό γραμμική σχέση δεν εμφανίζει τέτοια κλιμάκωση δηλαδή δεν δεκαπλασιάζεται. Αυτό εξηγείται εύκολα στατιστικά αφού η αύξηση του πλήθους των έργων δεν επιβαρύνει αποκλειστικά για παράδειγμα έναν όρο αλλά κατανέμεται στους όρους του θησαυρού του συστήματος (βλ. προηγούμενο παράδειγμα με τον όρο tempera).

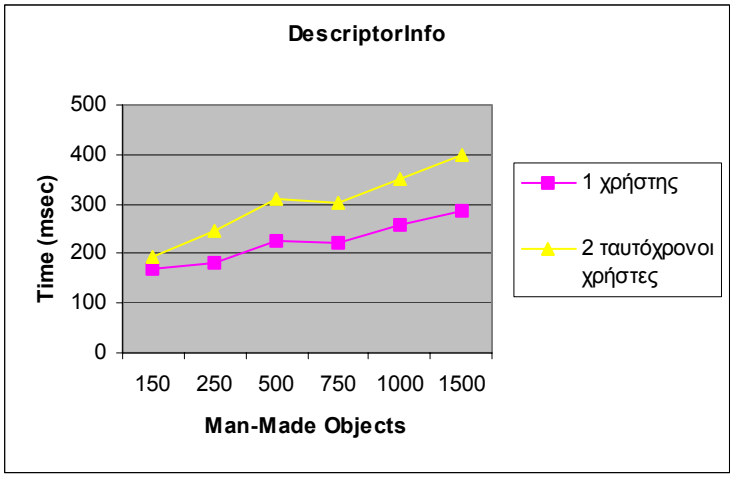
Το ίδιο ισχύει και για την περίπτωση των 2 «ταυτόχρονων» χρηστών όπου ο χρόνος εκτέλεσης μιας λειτουργίας κατά μέσο όρο είναι αρκετά μικρότερος από το διπλάσιο του χρόνου εκτέλεσής της για 1 χρήστη. Συγκεκριμένα ο χρόνος βλέπουμε από τον πίνακα 4.1 ότι κυμαίνεται κατά το πλείστο από 1.1 – 1.5 φορές το χρόνο εκτέλεσης για ένα χρήστη. Οι διακυμάνσεις οφείλονται α) στα ορίσματα εκτέλεσης των λειτουργιών που πιθανά διαφέρουν από τον 1 χρήστη και τους 2 «ταυτόχρονους» χρήστες, β) την τυχαιότητα κατασκευής των δεδομένων για τα διάφορα στιγμιότυπα της βάσης, γ) στις επικρατούσες συνθήκες στο μηχάνημα – εξυπηρέτη (server), που ειδικά για τις μικρές σε χρονικό κόστος εκτέλεσης λειτουργίες επηρεάζουν αρκετά τα αποτελέσματα και δ) στον τρόπο

υλοποίησης της πολυνηματικής επεξεργασίας τόσο από τον εξυπηρετητή εφαρμογής όσο και από τον εξυπηρετητή του SIS.

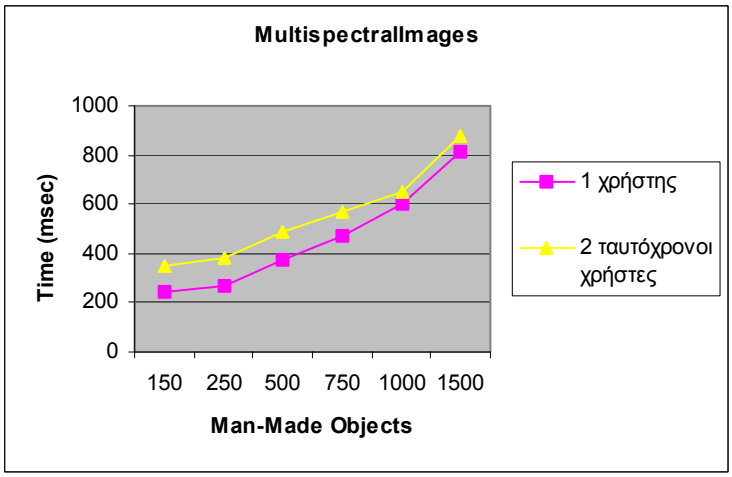
Στην ερώτηση αν θα περιμέναμε καλύτερα αποτελέσματα για την περίπτωση των 2 «ταυτόχρονων» χρηστών η απάντηση είναι όχι. Ο λόγος είναι ότι ο επεξεργαστής όπου εκτελείται ο εξυπηρετητής εφαρμογής είναι ένας. Αυτό σημαίνει ότι αυτός επιβαρύνεται με την εκτέλεση όλων των πράξεων για κάθε νήμα εκτέλεσης (thread). Μάλιστα υπάρχει και επιπλέον επιβάρυνση για την μεταγωγή μεταξύ νημάτων.

Ωστόσο κάποιος θα ισχυριστεί ότι οι δημοφιλείς εξυπηρετητές εφαρμογής είναι σχεδιασμένοι ώστε να εξυπηρετούν εκατοντάδες χρηστών χωρίς να παρατηρούνται ιδιαίτερες καθυστερήσεις. Αυτό όντως συμβαίνει αλλά κυρίως για αιτήσεις προσανατολισμένες σε ανάκτηση στατικών δεδομένων ή δυναμικών με εκτεταμένη χρήση caching και τεχνικών στατικής αποθήκευσης δεδομένων σε συνόδους (sessions).

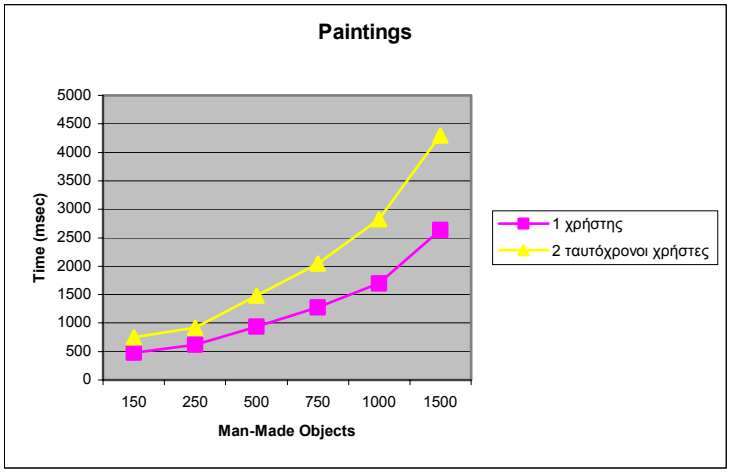
Στην περίπτωση του παρόντος συστήματος και για 2 «ταυτόχρονες» λειτουργίες η πολυνηματική επεξεργασία βοηθάει στη βελτίωση των επιδόσεων όταν για τη μία λειτουργία εκτελούνται πράξεις Java και για την άλλη επερωτήσεις προς το SIS. Σ' αυτή την περίπτωση η λειτουργία που εκτελεί πράξεις δεν επιβαρύνεται από την καθυστέρηση ανάγνωσης από το δίσκο της λειτουργίας που εκτελεί επερωτήσεις. Το θέμα είναι όμως ότι για τις «ταυτόχρονες» λειτουργίες αυτό δεν είναι συχνό φαινόμενο και γι' αυτό δε βλέπουμε ιδιαίτερες βελτιώσεις στις επιδόσεις. Πάντως οι επιδόσεις είναι σίγουρα καλύτερες απ' ότι αν είχαμε σειριακή εξυπηρέτηση των αιτήσεων.



Σχήμα 4.14 - Γραμμική επίδοση με μικρό συντελεστή της λειτουργίας DescriptorInfo



Σχήμα 4.15 - Επίδοση της λειτουργίας MultispectralImages



Σχήμα 4.16 - Επίδοση της χρονοβόρας λειτουργίας Paintings

Κεφάλαιο 5

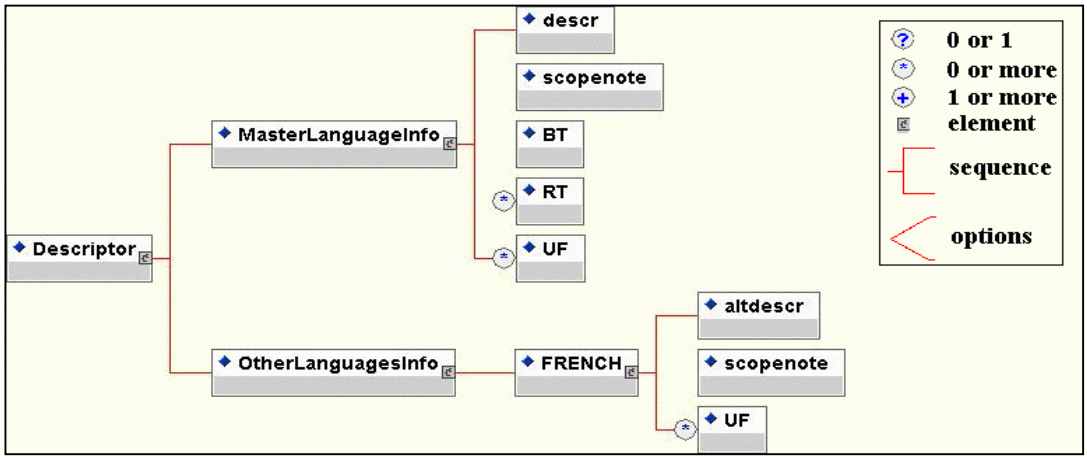
Σχεδίαση Εισαγωγής Δεδομένων στο Σύστημα

Στο κεφάλαιο αυτό παρουσιάζεται μια σχεδίαση της εισαγωγής δεδομένων στο σύστημα, η οποία μπορεί να χρησιμοποιηθεί ως βάση για την υλοποίηση της διαδικασίας αυτής. Αρχικά γίνεται διάκριση των βασικών δομικών μονάδων εισαγωγής και παρουσίαση των DTD τα οποία θα πρέπει να ακολουθούν οι δομικές αυτές μονάδες. Στη συνέχεια περιγράφονται εν συντομία οι λειτουργίες ενημέρωσης της βάσης και τα σημεία τα οποία θα πρέπει να προσεχθούν κατά την υλοποίηση.

5.1 Δομικές Μονάδες Εισαγωγής Δεδομένων

Οι δομικές μονάδες εισαγωγής δεδομένων δεν μπορούν παρά να επικεντρώνονται στις βασικές κλάσεις του συστήματος. Ακολουθούν παρακάτω τα DTD που κρίνουμε απαραίτητο να οριστούν:

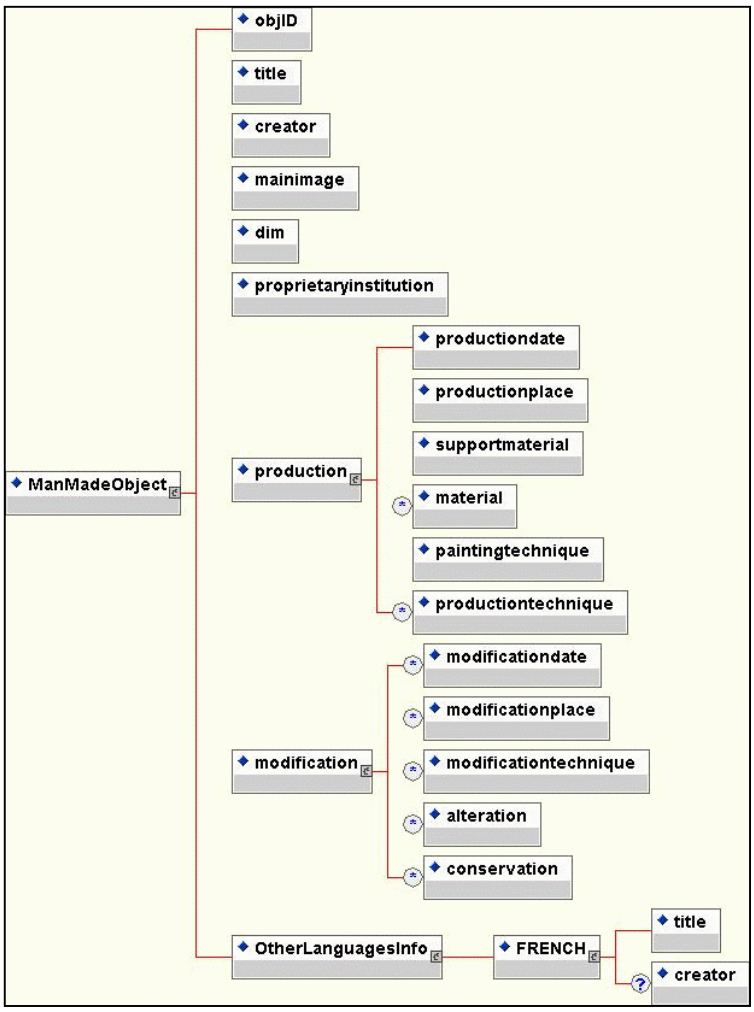
Descriptor.dtd : Υπάρχουν δυο ειδών πληροφορίες[12] που πρέπει να συνοδεύουν έναν Descriptor του θησαυρού. Οι πληροφορίες γι' αυτόν στη βασική γλώσσα και οι πληροφορίες στις υπόλοιπες γλώσσες, FRENCH, GERMAN κλπ. Στο σχήμα 5.1 φαίνεται καθαρά το DTD που προτείνουμε. Πρέπει να δοθεί η δέουσα προσοχή στο φορμαλισμό του σχήματος. Έτσι κάθε όρος έχει 1 όνομα descr, ακριβώς 1 BT, 0 ή περισσότερους RT, 0 ή περισσότερους UF και 1 scopenote στη βασική γλώσσα. Επίσης έχει 1 όνομα altdescr, 1 scopenote και 0 ή περισσότερους UF σε κάθε δευτερεύουσα γλώσσα.



Σχήμα 5.1 - Descriptor.dtd

[12] Οι «πληροφορίες» αντιστοιχούν σε tags των DTD. Λέγοντας πληροφορία X εννοούμε αντίστοιχο tag X.

ManMadeObject.dtd : Για ένα έργο τέχνης πρέπει να υπάρχουν
α) οι γενικές πληροφορίες objID, title, creator (βασική γλώσσα), mainimage (όνομα αρχείου), dim, proprietary institution (Place)
β) οι πληροφορίες κατασκευής (production)
γ) οι πληροφορίες τροποποιήσεων (modification)
δ) οι πληροφορίες title και creator στις δευτερεύουσες γλώσσες.
Η ύπαρξη και της μετάφρασης του creator στο συγκεκριμένο DTD είναι για να μην χρειάζεται ξεχωριστή λειτουργία εισαγωγής για τους Actors, που απαιτούν μόνο όνομα και μετάφραση.
Στο διπλανό σχήμα φαίνονται όλες οι πληροφορίες για ένα έργο τέχνης και ο απαιτούμενος φορμαλισμός για τις διάφορες σχέσεις.



Σχήμα 5.2 - ManMadeObject.dtd

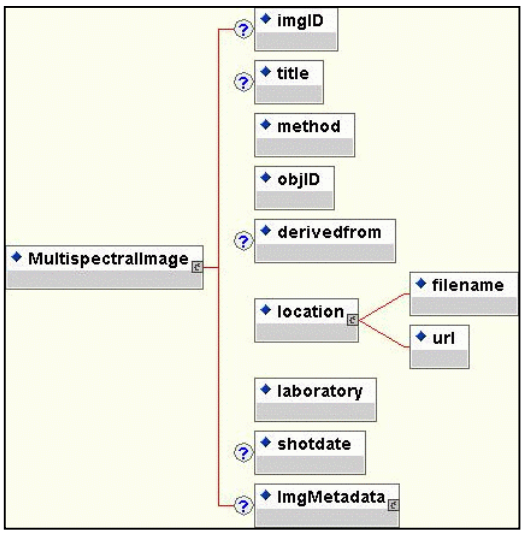
MultispectralImage.dtd : Για μια πολυφασματική εικόνα θα υπάρχουν υποχρεωτικά οι πληροφορίες method, objID, location (filename ή URL), laboratory. Προαιρετικές θα είναι οι πληροφορίες imgID[13], title, derivedfrom, shotdate και η σύνθετη πληροφορία ImgMetaData. Για την ImgMetaData πρέπει να πούμε ότι θα πρέπει να κατασκευαστεί από ειδικούς ένα δικό της DTD, το οποίο και θα ενσωματωθεί στο παρόν DTD. Η πληροφορία derivedfrom είναι πληροφορία συσχέτισης αντίστροφη από την derivative του μοντέλου για να μην απαιτείται για την καταχώρηση μιας εικόνας η καταχώρηση πιο πριν όλων των παραγόμενων απ' αυτήν.

Assessment.dtd : Για μια εκτίμηση θα υπάρχουν υποχρεωτικά οι πληροφορίες author, plainText (και στη βασική και στις δευτερεύουσες γλώσσες) και τουλάχιστο 1 imgID.

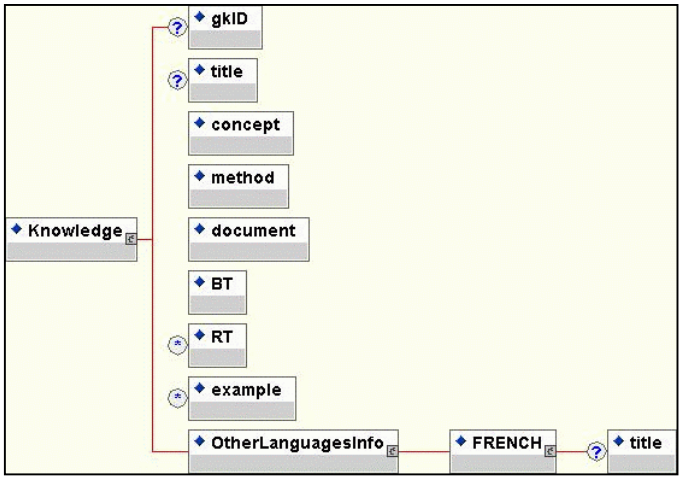
[13] Στην καταχώρηση το σύστημα θα δίνει αυτόματα κωδικό όνομα στην εικόνα, στη διόρθωση όμως αυτό πρέπει να εμφανίζεται.

Προαιρετικά θα υπάρχουν οι πληροφορίες subject (είτε στη βασική είτε στη δευτερεύουσα γλώσσα), regdate και οι relelaboration, relalteration, relmaterial, relconservation (0 ή περισσότερες φορές). Για τις συσχετίσεις που υπάρχουν στο μοντέλο και δεν εμφανίζονται στο παρόν DTD θα αναφερθούμε στην αντίστοιχη λειτουργία ενημέρωσης της βάσης.

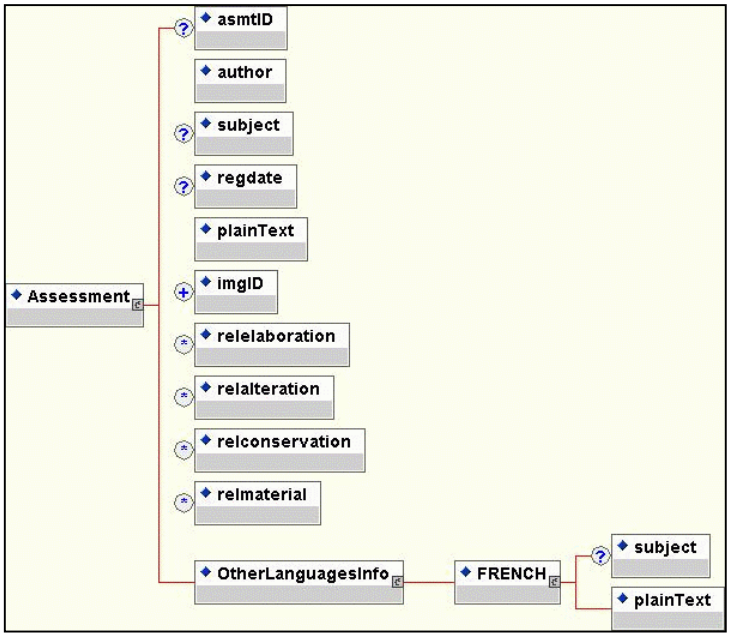
Knowledge.dtd : Για ένα κόμβο γνώσης θα υπάρχουν υποχρεωτικά οι πληροφορίες concept, method, document και BT. Προαιρετικές θα είναι οι πληροφορίες title (σε βασική και δευτερεύουσες γλώσσες), RT, example. Το κωδικό όνομα του κόμβου γνώσης gkID δεν απαιτείται κατά την εισαγωγή, αλλά καλό είναι εμφανίζεται στη διόρθωση, όπως συμβαίνει και για τις περιπτώσεις εικόνων και εκτιμήσεων.



Σχήμα 5.3 - MultispectralImage.dtd



Σχήμα 5.4 - Knowledge.dtd



Σχήμα 5.5 - Assessment.dtd

5.2 Λειτουργίες Ενημέρωσης της Βάσης

Για τις λειτουργίες ενημέρωσης της βάσης προτείνεται η χρήση XML Entry-Forms. Η σχετική δουλειά παρουσίασε πρώτη αυτή την τεχνική είναι το "Thesaurus Management Using XML" [53]. Σ' αυτήν την εργασία πελάτης του Συστήματος Διαχείρισης Θησαυρού Όρων (ΣΔΘΟ) είναι μια εφαρμογή αναπτυγμένη στον κειμενογράφο XML –XMetal– η οποία μέσω μιας σύνδεσης με το SIS, υλοποιημένης σε Java, παίρνει, επεξεργάζεται και στέλνει πάλι πίσω XML δεδομένα για την ενημέρωση του συστήματος. Πιο συγκεκριμένα ο διαχειριστής του θησαυρού ζητά μέσω της διεπαφής χρήστη έναν όρο του θησαυρού για ανανέωση. Κατάλληλα υλοποιημένη κλάση της Java αναλαμβάνει να συλλέξει όλες τις πληροφορίες που απαιτούνται για τον όρο, με τη χρήση επερωτήσεων, ώστε μορφοποιώντας τις να ακολουθείται προκαθορισμένο DTD. Το ρεύμα XML που παράγεται με αυτόν τον τρόπο αποθηκεύεται σε αρχείο στο περιβάλλον εργασίας του διαχειριστή. Ο τελευταίος έχει πλέον τη δυνατότητα να τροποποιήσει στοιχεία του όρου. Όταν γίνουν οι απαραίτητες τροποποιήσεις ο διαχειριστής μπορεί να ζητήσει ενημέρωση του ΣΔΘΟ. Με το πάτημα του κατάλληλου κουμπιού : α) ελέγχεται από το XMetal αν με τις αλλαγές που έγιναν εξακολουθούν τα στοιχεία του όρου να συμμορφώνονται με το προκαθορισμένο DTD, β) Το αρχικά αποθηκευμένο αρχείο XML και το νέο που θέλει να αποθηκεύσει ο διαχειριστής στο ΣΔΘΟ, μετασχηματίζονται από κατάλληλη Java κλάση σε DOM (Document Object Model) έγγραφα. γ) Τα 2 DOM έγγραφα συγκρίνονται από άλλη Java κλάση με χρήση κατάλληλων συναρτήσεων του λεγόμενου Java API for DOM και παράγεται ένα αρχείο κειμένου το οποίο περιέχει με συγκεκριμένη μορφοποίηση όλες τις αλλαγές που ο διαχειριστής επιθυμεί για τον όρο. δ) Μια τελευταία Java κλάση αναλαμβάνει την ανάλυση (parsing) του αρχείου και ενημέρωση του ΣΔΘΟ με κατάλληλες επερωτήσεις.

Στο παρόν σύστημα μπορεί να χρησιμοποιηθεί ως βάση η συγκεκριμένη εργασία για την εισαγωγή και ανανέωση δεδομένων. Αυτό απαιτεί την ύπαρξη πολλαπλών DTD (γι' αυτό και ορίστηκαν στην §5.1) και πολλαπλών κλάσεων για την ενημέρωση του συστήματος. Τα καίρια σημεία που πρέπει να προσεχθούν είναι οι ενέργειες που πρέπει να γίνουν κυρίως στις περιπτώσεις αλλαγών και όχι στις εισαγωγές. Οι εισαγωγές και οι αλλαγές μπορούν να γίνουν με τη χρήση XML Entry-Forms αλλά οι διαγραφές προτείνουμε να γίνουν με ξεχωριστές λειτουργίες.

5.2.1 Ενημέρωση όρων θησαυρού

Προτείνουμε 2 λειτουργίες για την ενημέρωση των όρων του θησαυρού, τις UpdateDescriptor και DeleteDescriptor. Η πρώτη θα αναλαμβάνει τις εισαγωγές και ανανεώσεις όρων με τη χρήση XML Entry-Forms, ενώ η δεύτερη τις διαγραφές και ό,τι αυτές συνεπάγονται με αυτόνομο πρόγραμμα.

Στις ακόλουθες περιγραφές των λειτουργιών θα χρησιμοποιήσουμε τον φορμαλισμό: <X> είναι το tag με όνομα X, |<X>| είναι το περιεχόμενο-τιμή του tag <X>, <X>/<Y> είναι το tag <Y> που έχει πατρικό <X>.

UpdateDescriptor : Αν πρόκειται για εισαγωγή ακολουθούνται τα εξής βήματα (για κατανόηση των βημάτων απαραίτητο είναι το σχήμα 5.1) :

1. Εξετάζεται αν υπάρχει ο όρος |<BT>| και σε ποια κλάση είναι στιγμιότυπο
2. Ο όρος |<descr>| υποστασιοποιείται κάτω από την κλάση αυτή και συνδέεται με τον όρο |<BT>| με την ομώνυμη σχέση.
3. Για κάθε γλώσσα <OtherLanguageInfo>/<LANG> κατασκευάζονται στιγμιότυπα LANG_Descriptor με αντίστοιχα αναγνωριστικά |<altdescr>|
4. Κατασκευάζεται στιγμιότυπο ScopeNote με κείμενο το |<MasterLanguageInfo>/<scopenote>|, αναγνωριστικό |<descr>|-encomment, και συνδέεται με τον όρο με τη σχέση *has scopenote*
5. Κατασκευάζονται στιγμιότυπα LANG_ScopeNote για κάθε γλώσσα LANG και συνδέονται με το |<descr>|-encomment με τη σχέση *scopenote_to_LANG*
6. Για κάθε |<MasterLanguageInfo>/<UF>| κατασκευάζονται στιγμιότυπα ENGLISH_UFTerm (αν βασική γλώσσα είναι η αγγλική) και συνδέονται με τον όρο με τη σχέση *uf_ENGLISH*. Παρόμοια για κάθε |<OtherLanguageInfo>/<LANG>/<UF>| κατασκευάζονται στιγμιότυπα LANG_UFTerm και συνδέονται με τον όρο με τις σχέσεις *uf_LANG*.
7. Για κάθε <MasterLanguageInfo>/<RT> εξετάζεται αν υπάρχει ο αντίστοιχος όρος, κι αν ναι συνδέεται με τον όρο |<descr>| με την ομώνυμη σχέση.

Αν πρόκειται για ανανέωση υπάρχοντος όρου:

- Αλλαγή του |<descr>| σε |<descr>|' συνεπάγεται επερώτηση μετονομασίας (Rename_Node) αρκεί να μην υπάρχει άλλος όρος με αναγνωριστικό |<descr>|'. Το ίδιο ισχύει και για τους όρους |<OtherLanguageInfo>/<LANG>/<altdescr>|.
- Αλλαγή του |<MasterLanguageInfo>/<scopenote>| συνεπάγεται αλλαγή της τιμής[14] του γνωρίσματος *plainText* για τον κόμβο |<descr>|-encomment. Όμοια ισχύουν και για τα scopenote των δευτερευουσών γλωσσών και τους κόμβους |<descr>|-frcomment κ.ο.κ.
- Αλλαγή του |<BT>| επιτυγχάνει μόνο εφόσον ο νέος ευρύτερος όρος υπάρχει και είναι στιγμιότυπο της ίδιας κλάσης όπως και ο προηγούμενος. Σ' αυτήν την περίπτωση έχουμε αλλαγή της τιμής για την ιδιότητα *BT* του όρου |<descr>|.

[14] Αν η αλλαγή τιμής ιδιότητας δεν είναι υλοποιημένη στο SIS-API θέλουμε διαδοχικά διαγραφή και νέα εισαγωγή

Βλέποντάς το «μακροσκοπικά» επιτελείται μια μετακίνηση υπο-δένδρου μέσα σε μια ιεραρχία π.χ. των alterations.

- Αλλαγή του περιεχομένου κάποιου από τα <RT> επιτυγχάνεται μόνο εφόσον ο νέος σχετιζόμενος όρος υπάρχει, οπότε έχουμε αλλαγή της τιμής για την συγκεκριμένη ιδιότητα *RT* του όρου |<descr>|. Σβήσιμο κάποιου <RT> και του περιεχομένου του συνεπάγεται διαγραφή της αντίστοιχης συσχέτισης |<descr>|→*RT*, όχι όμως και του κόμβου προορισμού!
- Αλλαγή κάποιου |<UF>| σε |<UF>|' συνεπάγεται σβήσιμο της σχέσης |<descr>|→*uf*_LANG→|<UF>|. Ο όρος |<UF>| σβήνεται μόνο αν δεν έχει εξαρτήσεις με άλλα αντικείμενα της βάσης. Στη συνέχεια ακολουθεί προσθήκη του |<UF>|' κάτω από την κλάση LANG_UFTerm και εγκατάσταση της συσχέτισης |<descr>|→*uf*_LANG→|<UF>|'.

Πρέπει να σημειώσουμε ότι για κάθε μια από τις ιεραρχίες του θησαυρού απαιτείται με την κατασκευή του μοντέλου να τοποθετηθούν κάτω από τις αντίστοιχες κλάσεις οι αντίστοιχοι κορυφαίοι όροι. Έτσι κάτω από την κλάση Method πρέπει να τοποθετηθεί ο όρος method, κάτω από την Alteration ο όρος alteration κ.ο.κ. Οι όροι αυτοί είναι οι μόνοι όροι των ιεραρχιών που δεν έχουν κάποιον ευρύτερο όρο και δεν μπορούν να καταχωριστούν με την εισαγωγή δεδομένων που μόλις παρουσιάστηκε.

DeleteDescriptor : Η διαγραφή ενός Descriptor – έστω descr – είναι μια αυτόνομη διαδικασία η οποία πρέπει να αφαιρέσει από τη βάση τον όρο, όλες τις συσχετίσεις του με άλλα αντικείμενα και τέλος όλα τα αντικείμενα των οποίων η ύπαρξη οφείλονταν στην δική του ύπαρξη. Έτσι πρέπει να σβηστούν όλα τα αντικείμενα τύπου ScopeNote, LANG_ScopeNote και οι ιδιότητές τους που τα συνδέουν μεταξύ τους και με τον όρο descr. Επίσης πρέπει να σβηστούν όλες οι σχέσεις *uf_LANG* και οι αντίστοιχοι όροι LANG_UFTerm που δεν έχουν άλλες εξαρτήσεις. Πρέπει να σβηστούν όλες οι συσχετίσεις τύπου *RT* όχι όμως κι οι κόμβοι προορισμού των. Τέλος πρέπει να σβηστεί η σχέση *BT*, γεγονός απλό αν ο όρος είναι φύλλο της ιεραρχίας που ανήκει. Αν δεν είναι φύλλο θα πρέπει για όλους τους στενότερους του όρους να αλλάξει ο κόμβος προορισμού της σχέσης *BT* αυτών και οι σχέσεις να «δείχνουν» πλέον στον ευρύτερο όρο του descr.
Όλα τα παραπάνω θα γίνουν μόνο εφόσον δεν υπάρχουν ιδιότητες άλλων αντικειμένων (π.χ. Man-Made Objects) που να καταλήγουν στον όρο descr.

5.2.2 Ενημέρωση έργων τέχνης

Για την ενημέρωση των έργων τέχνης του παρόντος ΠΣΒΠΙ προτείνουμε όμοια τις λειτουργίες ενημέρωσης UpdateManMadeObject και διαγραφής DeleteManMadeObject.

UpdateManMadeObject : Στην περίπτωση εισαγωγής έχουμε την εξής ακολουθία βημάτων (απαραίτητη η θέαση του σχήματος 5.2 για κατανόηση) :

1. Προσθήκη στιγμιοτύπου με αναγνωριστικό |<objID>| κάτω από την κλάση Man-Made Object αν δεν υπάρχει άλλο έργο τέχνης με το ίδιο αναγνωριστικό
2. Προσθήκη των κατάλληλων primitive ιδιοτήτων του μοντέλου στο αντικείμενο |<objID>|, για τα tags <title>, <OtherLanguageInfo>/<LANG>/<title>, <dim>, <production date> και <modification date>. Οι τιμές των ιδιοτήτων θα είναι το περιεχόμενο των tags. Στην περίπτωση των ιδιοτήτων *production date* και *modification date* πρέπει να προστεθεί επίσης ένα link με αρχή τις ιδιότητες αυτές και προορισμό τις χρονικές περιόδους –στιγμιότυπα της κλάσης Period– όπου ανήκουν χρονικά οι τιμές τους.
3. Προσθήκη των ιδιοτήτων *production place* και *modification place* με τιμές τα περιεχόμενα των αντίστοιχων tags αν και μόνο αν αυτά είναι ονόματα από αναγνωριστικά τόπων (στιγμιότυπα Place). Το ίδιο ισχύει και για την ιδιότητα *proprietary institution*. Θα αναφέρουμε σε επόμενη παράγραφο τι προτείνουμε για την καταχώρηση στιγμιοτύπων κλάσεων όπως η Place.
4. Προσθήκη κατάλληλου στιγμιοτύπου imgN κάτω από την κλάση Multispectral Image και απόδοση τιμής σε μια εκ των ιδιοτήτων αυτού *file location* ή *url location* ανάλογα με το |<mainimage>|. Επίσης προσθήκη της συσχέτισης |<objID>|→*main image*→imgN. Το 'N' στο κωδικό όνομα imgN είναι ένας αύξων αριθμός που υποδηλώνει ότι η συγκεκριμένη εικόνα είναι η υπ' αριθμό N που καταχωρίστηκε στη βάση. Πριν τη συγκεκριμένη καταχώρηση στην κλάση Multispectral Image η primitive ιδιότητα κλάσης *last_id*[15] είχε την τιμή N-1 και πλέον πρέπει να ενημερωθεί με την τιμή N.
5. Προσθήκη στιγμιοτύπου κάτω από την κλάση Actor με αναγνωριστικό το |<creator>| και προσθήκη της συσχέτισης |<objID>|→*created by*→|<creator>|. Επίσης δημιουργία στιγμιοτύπων |<OtherLanguageInfo>/<LANG>/<creator>| και συσχετίσεων με το στιγμιότυπο |<creator>| με τη σχέση *actor_to_LANG*. Με αυτό τον τρόπο επιτυγχάνεται η προσθήκη Actors χωρίς να υλοποιηθεί ξεχωριστή λειτουργία.
6. Δημιουργία των συσχετίσεων που ξεκινούν από το |<objID>| και καταλήγουν σε όρους του θησαυρού με την διενέργεια των απαραίτητων ελέγχων. Έτσι π.χ. αν το |<supportmaterial>| ≡ panel, poplar , θα εξεταστεί αν ο συγκεκριμένος όρος υπάρχει

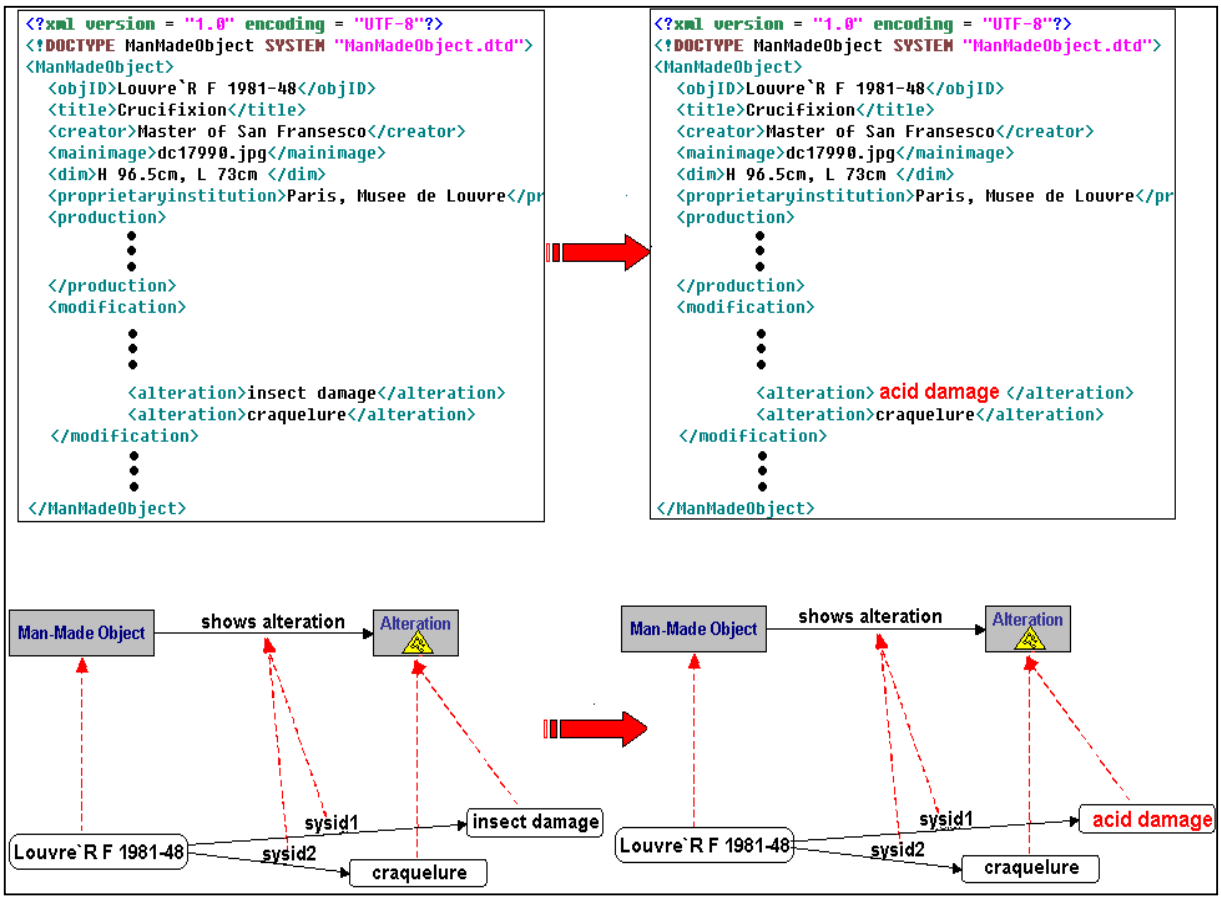
[15] Η συγκεκριμένη ιδιότητα δεν παρουσιάστηκε στο μοντέλο του κεφαλαίου 3

και είναι στιγμιότυπο της κλάσης Material. Αν ναι τότε θα δημιουργηθεί η συσχέτιση |<objID>|→*support material*→panel, poplar.

7. Δημιουργία στιγμιοτύπου Document με αναγνωριστικό docN (το N προκύπτει από την τιμή της ιδιότητας last_id της κλάσης Document) και προσθήκη ιδιότητας |<objID>→*described in document*→docN. Αυτή η συσχέτιση είναι απαραίτητη ώστε να τηρηθεί ο περιορισμός που λέει ότι κάθε έργο τέχνης έχει ακριβώς ένα στιγμιότυπο Document που παριστάνει το δελτίο συντήρησής του.

Στην περίπτωση των ενημερώσεων :

- Αλλαγή του |<objID>| σε |<objID>|' συνεπάγεται μετονομασία (Rename_Node) του κόμβου που παριστά το έργο τέχνης, αρκεί να μην υπάρχει άλλος κόμβος με το αναγνωριστικό |<objID>|'. Οι συσχετίσεις άλλων αντικειμένων με το |<objID>| δεν επηρεάζονται αφού ο κόμβος εξακολουθεί να έχει το ίδιο εσωτερικό αναγνωριστικό.
- Αλλαγή του περιεχομένου των <title> (βασικής και δευτερευουσών γλωσσών), <dim>, <productiondate> και <modificationdate> συνεπάγεται την αλλαγή των τιμών των αντίστοιχων ιδιοτήτων του μοντέλου με κατάλληλες επερωτήσεις. Πιθανό όπως είπαμε και στην προηγούμενη παράγραφο αυτό να απαιτεί διαγραφή και εκ νέου ανάθεση τιμών για τις ιδιότητες ανάλογα με τις συναρτήσεις που παρέχονται από το SIS-API.
- Αλλαγή του περιεχομένου των tags που παριστάνουν όρους θησαυρού π.χ. των <supportmaterial>,<paintingtechnique>,<modificationtechnique> κ.ο.κ συνεπάγεται αλλαγή της τιμής προορισμού της αντίστοιχης ιδιότητας του αντικειμένου |<objID>|. Για να γίνει κατανοητό τι θα πρέπει να συμβεί στην περίπτωση πλειότιμων ιδιοτήτων που καταλήγουν σε όρους του θησαυρού υπάρχει το σχήμα 5.6. Στο σχήμα αυτό αν στο επεξεργαζόμενο XML file σβήνονταν εντελώς το tag <alteration>insect damage</alteration> τότε αυτό θα συνεπάγονταν και σβήσιμο της σχέσης |<objID>|→*shows alteration*→insect damage από τη βάση.
- Αλλαγή του |<mainimage>|, το οποίο είναι όνομα αρχείου ή url, συνεπάγεται εξέταση για το αντικείμενο imgN (για το οποίο ισχύει |<objID>|→*main image*→imgN) των ιδιοτήτων του *file location* και *url location* και ανάθεση μιας εκ των δύο με το νέο περιεχόμενο του tag. Αν το νέο περιεχόμενο είναι url και το προηγούμενο ήταν filename τότε απαιτείται διαγραφή της ιδιότητας imgN→*file location* και προσθήκη της ιδιότητας imgN→*url location*. Αν το περιεχόμενο του tag ήταν και παραμένει filename τότε απλά έχουμε αλλαγή της τιμής της ιδιότητας *file location*.



Σχήμα 5.6 - Παράδειγμα ενημέρωσης έργου τέχνης

- Αλλαγή των |<productionplace>| και |<modificationplace>| συνεπάγεται αλλαγή της τιμής των αντίστοιχων ιδιοτήτων του μοντέλου, αρκεί το νέο περιεχόμενο των tags να είναι αναγνωριστικό κάποιου στιγμιοτύπου της κλάσης Place.
- Το ίδιο ισχύει και για αλλαγή του |<creator>|. Σ' αυτήν την περίπτωση όμως αν υπήρχε η συσχέτιση |<objID>|→*created by*→|<creator>| και το στιγμιότυπο |<creator>| δεν σχετίζεται με κανένα άλλο αντικείμενο της βάσης, τότε αυτή η αλλαγή θα σημάνει και σβήσιμο του στιγμιοτύπου.
- Προσοχή πρέπει να δοθεί στις αλλαγές των |<productiondate>| και |<modificationdate>| , όπου αυτό συνεπάγεται εκτός από αλλαγές των τιμών των αντίστοιχων ιδιοτήτων στη βάση και σβήσιμο των link *in period* που ξεκινούν απ' αυτές (τις ιδιότητες) και καταλήγουν σε στιγμιότυπα Period. Στη συνέχεια πρέπει να δημιουργηθούν νέα links προς τα κατάλληλα στιγμιότυπα Period ανάλογα με τις τιμές των ιδιοτήτων |<objID>|→*production date* και |<objID>|→*modification date*.

DeleteManMadeObject : Το στιγμιότυπο με αναγνωριστικό objID δεν μπορεί να διαγραφεί στις περιπτώσεις που υπάρχουν links προς αυτό από άλλα αντικείμενα όπως στιγμιότυπα Assessment και Multispectral Image. Στην περίπτωση που αυτό δεν ισχύει τότε αρχικά γίνεται διαγραφή όλων των ιδιοτήτων που ξεκινούν από το στιγμιότυπο

objID. Πρέπει να επισημάνουμε ότι στις περιπτώσεις των ιδιοτήτων *production date* και *modification date* πρέπει να διαγραφούν και τα links *in period* που ξεκινούν απ' αυτές. Στην περίπτωση της ιδιότητας *created by* αν το αντικείμενο προορισμός τύπου Actor, έστω actorX, δεν σχετίζεται με άλλα αντικείμενα της βάσης, τότε θα πρέπει αυτό να διαγραφεί. Επίσης αν υφίστανται σχέσεις της μορφής actorX→actor_to_LANG→actorL τότε και οι ιδιότητες *actor_to_LANG* και τα αντικείμενα actorL θα πρέπει να διαγραφούν. Στο τέλος μπορεί να διαγραφεί και το ίδιο το στιγμιότυπο objID.

5.2.3 Ενημέρωση πολυφασματικών εικόνων

UpdateImage : Για την περίπτωση εισαγωγής εικόνας έχουμε τα εξής βήματα (απαραίτητη η θέαση του σχήματος 5.3):

1. Σύνθεση του αναγνωριστικού που θα έχει η εικόνα ως "img" + 'N', αν N-1 είναι η τιμή της ιδιότητας *last_id* της κλάσης Multispectral Image. Στη συνέχεια καταχώρηση του στιγμιοτύπου με το αναγνωριστικό αυτό. Αυτό σημαίνει ότι το <imgID> είναι καθαρά "read-only" tag για την περίπτωση διόρθωσης (θέαση του αναγνωριστικού).
2. Προσθήκη της ιδιότητας *file location* ή *url location* ανάλογα με το tag <location> της XML φόρμας της εικόνας
3. Προσθήκη των ιδιοτήτων *produced using method*, *depicts* με τιμές τα |<method>| και |<objID>| , αν και μόνο αν αυτά είναι αναγνωριστικά στιγμιοτύπων αντιστοίχως των κλάσεων Method και Man-Made Object.
4. Προσθήκη της ιδιότητας imgN→*laboratory*→|<laboratory>|. Έπειτα δημιουργία στιγμιοτύπου Actor με αναγνωριστικό το |<laboratory>| αν δεν υπάρχει τέτοιο στιγμιότυπο ήδη.
5. Αν υπάρχει το <derivedfrom> και το περιεχόμενό του είναι αναγνωριστικό στιγμιοτύπου Multispectral Image τότε προσθήκη της σχέσης <derivedfrom>→*derivative*→imgN.
6. Προσθήκη των ιδιοτήτων *shot on date*, *title* και *conditions description* με τιμές τα |<shotdate>|, |<title>|, |<ImgMetaData>|[16] αν αυτά βέβαια υπάρχουν.

Στην περίπτωση τροποποιήσεων των στοιχείων μιας εικόνας:

- αλλαγές στο |<imgID>| δε λαμβάνονται υπόψη
- αλλαγή στο |<method>| σημαίνει αλλαγή της τιμής της ιδιότητας *produced using method*, αρκεί το νέο περιεχόμενο να είναι αναγνωριστικό υπάρχοντος στιγμιοτύπου της κλάσης Method. Το ίδιο ισχύει και για το |<objID>| όπου το νέο

[16] σύνθεση όλων των υπο-tags του στοιχείου αυτού ως XML string

περιεχόμενο θα πρέπει να είναι αναγνωριστικό υπάρχοντος στιγμιοτύπου της κλάσης Man-Made Object.

- Αλλαγές των |<title>|, |<shotdate>|, |<ImgMetadata>|, συνεπάγεται αλλαγή των τιμών των αντίστοιχων primitive ιδιοτήτων στη βάση. Αν τα tags δεν υπήρχαν καν αρχικά στην XML form για την εικόνα, τότε για όποια απ' αυτά προστεθούν θα υπάρξει προσθήκη αντίστοιχων ιδιοτήτων στη βάση για το αντικείμενο |<imgID>|.
- Αλλαγή στο |<location>| θα έχει την εξής επίδραση: αν πριν το περιεχόμενο ήταν το tag <filename> και τώρα είναι πάλι <filename> με διαφορετικό περιεχόμενο τότε θα έχουμε απλά αλλαγή της τιμής της ιδιότητας *file location* στη βάση. Αν όμως πριν το περιεχόμενο ήταν <filename> και τώρα είναι <url> τότε θα έχουμε διαγραφή της ιδιότητας *file location* του αντικειμένου imgID στη βάση και προσθήκη νέας ιδιότητας imgID→*url location*→|<url>|.
- Αλλαγή του |<laboratory>| θα σημάνει αλλαγή της τιμής της αντίστοιχης ιδιότητας στη βάση, δημιουργία στιγμιοτύπου Actor με αναγνωριστικό το περιεχόμενο του tag (αν δεν υπάρχει τέτοιο ήδη) και διαγραφή του στιγμιοτύπου Actor που πριν ήταν συνδεδεμένο με το imgID αν αυτό δεν έχει άλλες εξαρτήσεις

DeleteImage : Η διαγραφή μιας εικόνας θα είναι εφικτή μόνο αν δεν υπάρχουν Assessments που την έχουν χρησιμοποιήσει και δεν υπάρχουν Man-Made Objects που την έχουν ως *main image*. Στην περίπτωση που η εικόνα απλά είναι *derivative* κάποιας άλλης αυτό δεν είναι απαγορευτικό για τη διαγραφή της. Έτσι λοιπόν αν η διαγραφή είναι δυνατή τότε διαγράφονται όλες οι ιδιότητες που ξεκινούν απ' αυτή και μόνο αυτές. Μόνο στην περίπτωση της ιδιότητας *laboratory*, αν το στιγμιότυπο Actor με το οποίο η εικόνα συνδέονταν δεν έχει άλλες εξαρτήσεις, τότε θα διαγραφεί και το ίδιο.

5.2.4 Ενημέρωση εκτιμήσεων

UpdateAssessment : Για την περίπτωση εισαγωγής στιγμιοτύπου επίσημης εκτίμησης –όχι από χρήστη του συστήματος– έχουμε τα εξής βήματα (απαραίτητη η θέαση του σχήματος 5.5):

1. Σύνθεση του λογικού αναγνωριστικού για το προς καταχώρηση στιγμιότυπο. Θα έχει τη μορφή asmtN, όπου 'N' προκύπτει από την τιμή της primitive ιδιότητας *last_id* της κλάσης Assessment. Υποστασιοποίηση αυτού στη συνέχεια κάτω από την κλάση Assessment.
2. Προσθήκη των ιδιοτήτων *written by*, *registered on date*, *subject_ENGLISH*[17], *plainText_ENGLISH* με τιμές |<author>|, |<regdate>|, |<subject>|, |<plainText>| αντίστοιχα.

[17] Βασική γλώσσα στο παρόν ΠΣΒΠΙ έχει θεωρηθεί η αγγλική, εξού και ENGLISH

3. Προσθήκη των ιδιοτήτων *subject_LANG*, *plainText_LANG* με τιμές τα περιεχόμενα των <subject>, <plainText> που βρίσκεται κάτω από το <OtherLanguagesInfo>.
4. Προσθήκη των ιδιοτήτων *related with elaboration*, *related with alteration*, *related with conservation*, *related with material* με τιμές τα |<relelaboration>|, |<relalteration>|, |<relconservation>|, |<relmaterial>| αν και μόνο αν αυτά είναι αναγνωριστικά στιγμιοτύπων αντίστοιχα των κλάσεων Elaboration, Alteration, Conservation και Material.
5. Προσθήκη ιδιοτήτων asmtN→*used image*→|<imgID>|. Απαραίτητη προϋπόθεση τα περιεχόμενα των <imgID> να είναι λογικά αναγνωριστικά στιγμιοτύπων Multispectral Image.
6. Ακολουθώντας τους περιορισμούς ακεραιότητας που αναφέρθηκαν στο κεφάλαιο 3 πρέπει να πούμε ότι για κάθε ιδιότητα asmtN→*used image*→|<imgID>| που θα προστεθεί, αν υφίστανται και οι ιδιότητες |<imgID>|→*used method*→methodX και |<imgID>|→*depicts*→objID τότε θα προστεθούν και οι σχέσεις asmtN→*used method*→methodX και asmtN→*about object*→objID. Επίσης αν υφίσταται η σχέση objID→*described in document*→docID, τότε θα προστεθεί επίσης η ιδιότητα asmtN→*part of*→docID.

Στην περίπτωση ενημέρωσης μιας εκτίμησης:

- Αλλαγές στο αναγνωριστικό |<asmtID>| δε θα λαμβάνονται υπόψη
- Αλλαγές |<author>|, |<regdate>| αλλά και |<subject>|, |<plainText>| (βασική και δευτερεύουσες γλώσσες) θα συνεπάγεται αλλαγές στις τιμές των αντίστοιχων ιδιοτήτων του στιγμιοτύπου asmtID στη βάση. Σβήσιμο των tags <subject> και <regdate>, που δεν είναι υποχρεωτικά, θα σημάνει διαγραφή και των αντίστοιχων ιδιοτήτων από τη βάση
- Αλλαγές στα |<relelaboration>| κλπ. που καταλήγουν σε όρους του θησαυρού θα σημάνουν αλλαγές τιμών στις αντίστοιχες ιδιότητες της βάσης. Σβήσιμο κάπου tag π.χ. <relalteration>insect damage</relalteration> θα έχει ως αποτέλεσμα διαγραφή της ιδιότητας |<asmtID>|→*related with alteration*→insect damage.
- Αλλαγή κάποιου περιεχομένου |<imgID>| σε |<imgID>|' , θα έχει ως αποτέλεσμα την αλλαγή της τιμής της αντίστοιχης ιδιότητας |<asmtID>|→*used image*→|<imgID>| σε |<asmtID>|→*used image*→|<imgID>|'. Αυτό θα έχει σχετικά περίπλοκη συνέπεια που θα είναι δυσνόητο να την πούμε με λόγια. Μπορούμε να πούμε όμως ότι η υλοποίηση θα πρέπει να είναι τέτοια ώστε μια σχέση asmtID→*used method*→methodX θα υφίσταται μόνο αν υπάρχει έστω 1 στιγμιότυπο |<imgID>| τέτοιο ώστε να ισχύουν |<asmtID>|→*used image*→|<imgID>| και |<imgID>|→*used method*→methodX. Το ίδιο θα πρέπει να ισχύει και για την ιδιότητα *about object*. Άρα με την αλλαγή |<imgID>| σε

|<imgID>|' ίσως να απαιτείται διαγραφή κάποιας σχέσης asmtID→*used method* ή/και asmtID→*about object*.

DeleteAssessment : Ένα στιγμιότυπο Assessment θα μπορεί να διαγραφεί μόνο εάν δεν αποτελεί παράδειγμα για κάποιο στιγμιότυπο GeneralAssessmentKnowledge και αν δεν αποτελεί βάση για την διατύπωση κάποιου άλλου στιγμιοτύπου Assessment (ανακλαστική σχέση *took into account*). Αν ισχύουν αυτές οι προϋποθέσεις τότε μπορούν να διαγραφούν όλες οι ιδιότητες που ξεκινούν απ' αυτό και μόνο αυτές και στην συνέχεια να διαγραφεί και το ίδιο από στιγμιότυπο της κλάσης Assessment.

5.2.5 Ενημέρωση κόμβων γνώσης

UpdateKnowledge: Για την περίπτωση εισαγωγής στιγμιοτύπου γνώσης GeneralAssessmentKnowledge έχουμε τα εξής βήματα (απαραίτητη η θέαση του σχήματος 5.4):

1. Σύνθεση του λογικού αναγνωριστικού για το προς καταχώρηση στιγμιότυπο. Θα έχει τη μορφή GKN, όπου 'N' προκύπτει από την τιμή της primitive ιδιότητας *last_id* της κλάσης GeneralAssessmentKnowledge. Στη συνέχεια υποστασιοποίησή του κάτω από την κλάση.
2. Προσθήκη στο GKN των ιδιοτήτων *title*, *title_LANG* για όσα από τα αντίστοιχα tags υφίστανται στην XML form
3. Προσθήκη των ιδιοτήτων *is about concept* και *is obtained by method* αρκεί τα |<concept>| και |<method>| να είναι αναγνωριστικά στιγμιοτύπων Conservation Concept (εμμέσως) και Method αντίστοιχα.
4. Δημιουργία συσχετίσεων GKN→*BT*, GKN→*RT* για τα περιεχόμενα των ομώνυμων tags αν και μόνο αυτά είναι αναγνωριστικά στιγμιοτύπων Knowledge.
5. Προσθήκη ιδιοτήτων GKN→*has example* με τιμές τα περιεχόμενα των tags <example> αρκεί αυτά να είναι αναγνωριστικά στιγμιοτύπων Assessment.
6. Προσθήκη στιγμιοτύπου docN κάτω από την κλάση Document (το 'N' προκύπτει από την ιδιότητα *last_id* της κλάσης) και ιδιότητας *file location* σ' αυτό με τιμή το |<document>|. Επίσης δημιουργία της συσχέτισης GKN→*has scopenote*→docN. Επίσης για κάθε γλώσσα LANG που υποστηρίζει το σύστημα δημιουργία στιγμιοτύπων docNL κάτω από τις κλάσεις LANG_Document και προσθήκη σ' αυτά ιδιότητας *file location* με την ίδια τιμή όπως και για το docN. Τέλος προσθήκη στο docN όλων των ιδιοτήτων της μορφής docN→*document_to_LANG*→docNL.

   Προσοχή!, αυτό εκ των υστέρων απαιτεί τοποθέτηση από τους διαχειριστές του συστήματος αρχείων με ονόματα ίδια με την τιμή της ιδιότητας του docN *file*

location, σε όλους τους φακέλους της μορφής **I.project_docs_rel_path/ I.LANG[lid], ∀lid** (βλ. §4.1.1)

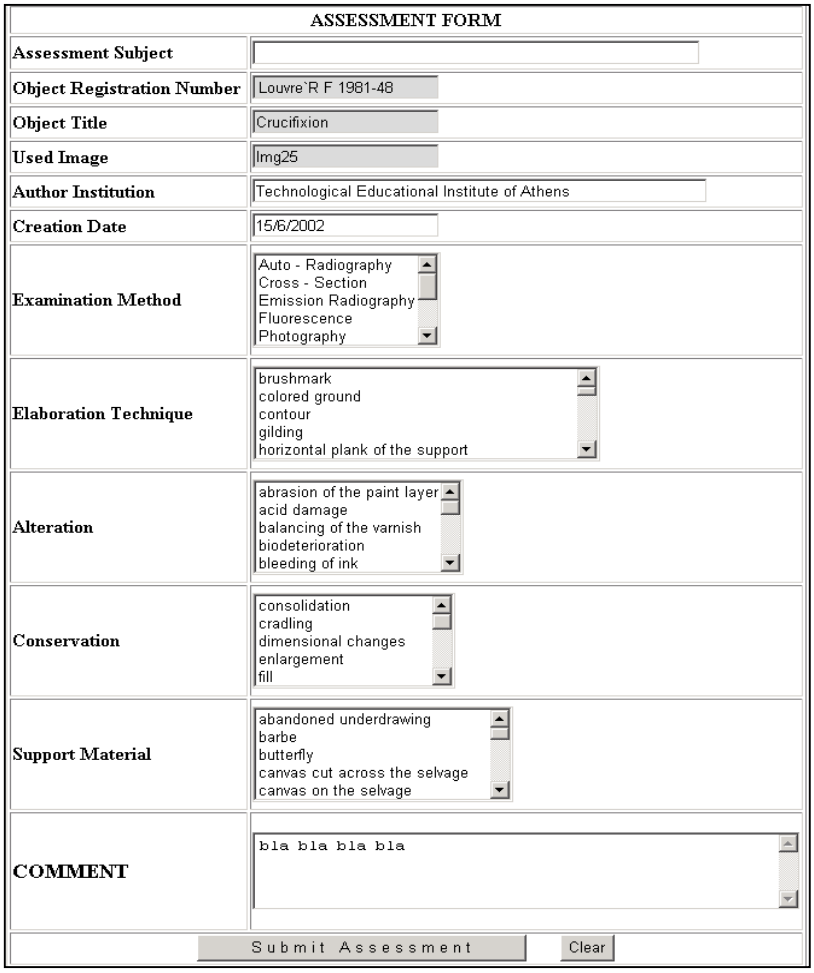
Στην περίπτωση ενημερώσεων :

- Αλλαγές στο |< gkID>| παραβλέπονται
- Αλλαγές στα |<title>| (βασικής και δευτερευουσών γλωσσών) συνεπάγονται αλλαγές τιμών για τις αντίστοιχες ιδιότητες *title*, *title_LANG*.
- Αλλαγές στα |<method>|, |<concept>| συνεπάγεται αλλαγή των τιμών των ιδιοτήτων *is related with concept* και *is obtained by method* αρκεί τα νέα περιεχόμενα των tags να είναι αναγνωριστικά στιγμιοτύπων των κλάσεων Conservation Concept (έμμεσα) και Method αντίστοιχα
- Αλλαγές στο |<document>| σημαίνει ενημέρωση των τιμών των ιδιοτήτων *file location* του στιγμιοτύπου Document (docN) και των στιγμιοτύπων LANG_Document (docNL) για τα οποία ισχύει GKN→*has scopenote*→docN και docN→*document_to_LANG*→docNL. Προσοχή και πάλι στην προσθήκη των κατάλληλων αρχείων στους κατάλληλους φακέλους της εφαρμογής.
- Αλλαγή των |<RT>| και |<example>| συνεπάγονται την αλλαγή των τιμών των ιδιοτήτων *RT* και *has example* αρκεί τα νέα περιεχόμενα των tags να είναι αναγνωριστικά στιγμιοτύπων GeneralAssessmentKnowledge και Assessment αντίστοιχα. Διαγραφή κάποιου τέτοιου tag συνεπάγεται διαγραφή της αντίστοιχης ιδιότητας.
- Αλλαγή του |<BT>| θα έχει σαν αποτέλεσμα την αλλαγή της αντίστοιχης ιδιότητας στη βάση, αρκεί το νέο περιεχόμενο του tag να είναι αναγνωριστικό κάποιου άλλου κόμβου γνώσης.

DeleteKnowledge: Η διαγραφή ενός στιγμιοτύπου GeneralAssessmentKnowledge GKID γενικά επιτρέπεται εκτός κι αν αυτό αποτελεί τον κορυφαίο κόμβο στην ιεραρχία των γνώσεων. Μια τέτοια διαγραφή θα έχει ως συνέπεια της διαγραφή όλων των ιδιοτήτων που ξεκινούν απ' αυτό, όλων των στιγμιοτύπων Document και LANG_Document που συνδέονται άμεσα κι έμμεσα μ' αυτό και αντίστοιχα των ιδιοτήτων *has scopenote* και *document_to_LANG*. Πριν σβηστεί η σχέση *BT* που το συνδέει με την υπόλοιπη ιεραρχία των γνώσεων πρέπει να γίνει εξέταση αν αυτό είναι φύλλο, οπότε δε θα χρειαστεί καμιά ενέργεια, ή αν δεν είναι φύλλο οπότε για κάθε «άμεσα στενότερο» κόμβο γνώσης θα απαιτηθεί αλλαγή της τιμής της ιδιότητας του *BT* ώστε να έχει προορισμό τον «ευρύτερο» του GKID.

5.2.6 Εκτιμήσεις Χρηστών Συστήματος

Οι εκτιμήσεις χρηστών θα πρέπει να γίνονται με ειδική λειτουργία που θα παρέχεται από τη δικτυακή εφαρμογή του συστήματος. Η λειτουργία αυτή προτείνεται να είναι προσβάσιμη μέσα από τη λειτουργία θέασης πληροφοριών πολυφασματικής εικόνας (MultispectralImageView) με επιλογή κατάλληλου συνδέσμου.



Σχήμα 5.7- Φόρμα εισαγωγής εκτίμησης χρήστη

Η λειτουργία θα εμφανίζει μια φόρμα, όπως αυτή του σχήματος 5.7 στην οποία θα είναι αυτόματα συμπληρωμένα ο κωδικός και ο τίτλος του έργου στο οποίο αναφέρεται η εικόνα και ο κωδικός εικόνας. Ο χρήστης θα πρέπει να συμπληρώσει τα υπόλοιπα στοιχεία.

Η διαφορά με τις εκτιμήσεις των χρηστών είναι ότι δεν αποτελούν τμήμα κάποιου στιγμιοτύπου Document και δεν εμφανίζονται στο επίσημο δελτίο συντήρησης έργου αλλά στην ξεχωριστή λειτουργία PaintingUserAssessments. Το κείμενο που εισάγεται προτείνουμε να καταχωρίζεται αυτούσιο ως τιμή για όλες τις ιδιότητες *plainText_LANG* (το ίδιο και για το Assessment Subject) γιατί διαφορετικά θα έπρεπε κάποιος συντηρητής να ενημερώνει όλες τις εκτιμήσεις χρηστών κάνοντας μεταφράσεις σε πολλές γλώσσες...

5.2.7 Εισαγωγές Periods και Places

Όπως έχουμε προαναφέρει τα στιγμιότυπα κάτω από τις κλάσεις Period και Place θέλουμε να είναι δομημένα ιεραρχικά με χρήση των ανακλαστικών σχέσεων *falls within*. Προτείνουμε οι εισαγωγές στιγμιοτύπων γι' αυτές τις κλάσεις να γίνει εξ' αρχής και να μην υλοποιηθούν αντίστοιχες λειτουργίες ενημερώσεων. Εξάλλου πιστεύουμε ότι τέτοιου είδους ιεραρχίες δεν έχουν την δυναμική εξέλιξης που εμφανίζουν οι ιεραρχίες του θησαυρού, είναι δηλαδή κατά κάποιο τρόπο στατικές. Πιθανές αλλαγές που ίσως χρειαστούν εύκολα θα μπορούν να γίνουν με χρήση των SIS-Entry Forms [54].

Σημειωτέον ότι στην ιεραρχία Place πρέπει να τοποθετηθούν και τα ιδρύματα που είναι υπεύθυνα για τα έργα τέχνης (*proprietary institutions*). Έτσι η ιεραρχία αυτή θα μπορούσε να έχει για παράδειγμα μια διαδρομή:

WholeWorld→Europe→France→Musee de Louvre.

Κεφάλαιο 6

Επίλογος / Βελτιώσεις / Επεκτάσεις

Στην εργασία αυτή παρουσιάστηκε η σχεδίαση και μερική υλοποίηση ενός ΠΣΒΠΙ για εκπαιδευτική χρήση από συντηρητές έργων τέχνης. Η υλοποίηση ενός τέτοιου συστήματος κρίθηκε επιβεβλημένη στα πλαίσια του προγράμματος CRISATEL (Conservation Restoration Innovation Systems for image capture and digital Archiving to enhance Training Education and lifelong Learning). Το σύστημα που υλοποιήθηκε αποτελεί ένα μέσο :

- εξοικείωσης με τις έννοιες συντήρησης και αποκατάστασης έργων τέχνης
- θέασης υπαρκτών παραδειγμάτων-πινάκων ζωγραφικής που σχετίζονται ποικιλοτρόπως με τις παραπάνω έννοιες
- πλοήγησης στην καταχωρισμένη γνώση για το συγκεκριμένο πεδίο εφαρμογής
- επικοινωνίας μεταξύ συντηρητών διαφορετικών εθνικοτήτων

6.1 Σύνοψη εργασίας

Για την μοντελοποίηση του πεδίου εφαρμογής χρησιμοποιήθηκε το μοντέλο δεδομένων της Telos και για την αποθήκευση, διαχείριση και ανάκτηση δεδομένων το Σύστημα Σημασιολογικού Ευρετηριασμού. Η μοντελοποίηση έγινε κατά τέτοιο τρόπο ώστε να υποστηρίζονται τα βασικά χαρακτηριστικά πολύγλωσσων θησαυρών, να υπάρχει αναπαράσταση και ιεραρχική οργάνωση της γνώσης του πεδίου και να υποβοηθείται η ανάκτηση παραδειγμάτων σχετικών με ευρύτερες και στενότερες έννοιες.

Για την υλοποίηση του συστήματος ακολουθήθηκε server-side αρχιτεκτονική 4-επιπέδων όπου τα δυο μεσαία επίπεδα καταλαμβάνουν ο εξυπηρετητής του ΠΙ και ο εξυπηρετητής εφαρμογής. Σ' αυτού του είδους την αρχιτεκτονική όλη η λογική επεξεργασίας βρίσκεται στον εξυπηρετητή εφαρμογής ως μια δικτυακή εφαρμογή. Ο εξυπηρετητής εφαρμογής είναι ο παροχέας περιβάλλοντος εκτέλεσης για τις λειτουργίες της εφαρμογής. Η λειτουργία της αρχιτεκτονικής είναι η εξής: ο χρήστης ζητάει την εκτέλεση μιας λειτουργίας μέσω URL και του πρωτοκόλλου HTTP. Η αίτησή του φτάνει στον εξυπηρετητή του ΠΙ, ο οποίος την εξετάζει και αν χρειάζεται την προωθεί στον εξυπηρετητή εφαρμογής. Ο τελευταίος εκτελεί την λειτουργία στο περιβάλλον του και επιστρέφει τα αποτελέσματα στον εξυπηρετητή του ΠΙ ο οποίος τα στέλνει μαζί με όσα άλλα παρελκόμενα αρχεία απαιτούνται στον χρήστη.

Η δικτυακή εφαρμογή του συστήματος σχεδιάστηκε να είναι πλήρως παραμετρική ώστε να μπορεί να εγκατασταθεί με απλό τρόπο οπουδήποτε. Γι' αυτήν κατασκευάστηκαν ένα σύνολο λειτουργιών που κρίθηκαν απαραίτητες για ένα εκπαιδευτικό ηλεκτρονικό εγχειρίδιο για συντηρητές. Τα αποτελέσματα των λειτουργιών είναι βασισμένα πλήρως στις γλώσσες XML-XSL. Οι επιδόσεις τους κρίθηκαν ικανοποιητικές αφού προσεγγιστικά εμφανίζουν γραμμική πολυπλοκότητα.

Τέλος προτάθηκε ένα σύνολο λειτουργιών για προσθήκη-ενημέρωση αντικειμένων βασικών κλάσεων με τη χρήση XML Entry-Forms καθώς και διαγραφών με τη χρήση αυτόνομων προγραμμάτων. Οι λειτουργίες αυτές βασίζονται σε συγκεκριμένα απλά DTD σχεδιασμένα με τέτοιο τρόπο ώστε να απλοποιείται η όλη διαδικασία εισαγωγής δεδομένων για τους διαχειριστές του συστήματος.

6.1.1 Οφέλη και νεωτερισμοί στο πεδίο εφαρμογής

Το συγκεκριμένο ΠΣΒΠΙ περιλαμβάνει ένα σύνολο χαρακτηριστικών που είναι χρήσιμα για όσους ασχολούνται με τη συντήρηση έργων τέχνης.

Πρώτα απ' όλα η γραφική εμφάνιση ιεραρχιών βοηθάει στην ευκολότερη κατανόηση των διαφόρων όρων ή ακόμα και στην εύρεση της ονομασίας κάποιου όρου σε περίπτωση που είναι γνωστό μόνο τι αυτός εκφράζει.

Η πολυγλωσσία και η άμεση εναλλαγή γλώσσας χρησιμεύει στη συνεννόηση συντηρητών διαφορετικών εθνικοτήτων όσον αφορά έργα τέχνης που μεταφέρονται από μουσείο σε μουσείο για λόγους δανεισμού, εκθέσεων και συντήρησης.

Η σύνοψη των περιεχομένων της βάσης με αριθμούς ουσιαστικά παρουσιάζει τις τεχνοτροπίες και υλικά που κυριαρχούσαν στις διάφορες χρονικές περιόδους όσον αφορά την κατασκευή έργων τέχνης.

Η παράσταση γνώσης του πεδίου βοηθάει τους εκπαιδευόμενους συντηρητές να γνωρίσουν πώς οι διάφορες μέθοδοι εξέτασης πολυφασματικών εικόνων βοηθούν σε θέματα διάγνωσης, συντήρησης και αποκατάστασης. Η θέαση παραδειγμάτων έχει βοηθητικό χαρακτήρα σ' αυτό.

Οι λειτουργίες αναζήτησης βοηθούν στον εντοπισμό αντικειμένων που ικανοποιούν σύνολα κριτηρίων και είναι αρκετά χρήσιμες για τους γνώστες του πεδίου οι οποίοι έτσι μπορούν να εστιάσουν εύκολα σε συγκεκριμένα παραδείγματα που τους ενδιαφέρουν.

Τέλος και σημαντικότερο όλων είναι η πρόσβαση όλων των παραπάνω λειτουργιών μέσω του Παγκόσμιου Ιστού που, σε συνδυασμό με τη δυνατότητα κατάθεσης εκτιμήσεων χρηστών, μπορεί να κάνει το παρόν ΠΣΒΠΙ ένα διεθνές forum σε θέματα συντήρησης κι αποκατάστασης.

Οι νεωτερισμοί που παρουσιάζονται με το παρόν σύστημα είναι τόσο στο επίπεδο του πεδίου εφαρμογής, όπου δεν υπάρχουν ΠΣΒΠΙ με παρόμοια χαρακτηριστικά για τη συντήρηση έργων τέχνης, όσο και στο επίπεδο της δικτυακής εφαρμογής όπου ελάχιστα

συστήματα χρησιμοποιούν παρόμοιες τεχνικές. Πιο συγκεκριμένα νεωτερισμούς αποτελούν:

- η χρήση του μοντέλου δεδομένων της Telos και του SIS που ουσιαστικά επιτρέπει στις λειτουργίες της εφαρμογής να εκμεταλλευτούν την ιεράρχηση που εμφανίζουν οι όροι και οι γνώσεις του πεδίου εφαρμογής
- η πολυγλωσσία με άμεση εναλλαγή γλώσσας που επιτυγχάνεται με συνδυασμό της μοντελοποίησης, της παραμετροποίησης της εφαρμογής και των ορισθέντων DTD
- οι πολλαπλές άμεσα εναλλασσόμενες παρουσιάσεις
- Η εμφάνιση ιεραρχιών όρων μέσα σε φυλλομετρητή του ΠΙ
- Η πλήρης ανεξαρτητοποίηση λογικής επεξεργασίας και λογικής παρουσίασης με τη χρήση XML-XSL.

6.2 Βελτιώσεις - Επεκτάσεις

Για την ολοκλήρωση και τελειοποίηση του συστήματος που αναπτύχθηκε στην παρούσα εργασία υπάρχει ένας αριθμός βελτιώσεων και επεκτάσεων που μπορούν να υλοποιηθούν μελλοντικά. Ορισμένες από αυτές αφορούν στο παρόν σύστημα αποκλειστικά, ενώ άλλες συνιστούν χρήσιμες επεμβάσεις που θα μπορούσαν να γίνουν στο SIS ή αυτονόμηση στοιχείων αυτού.

Η βελτίωση της λειτουργίας γραφικής απεικόνισης ιεραρχιών όρων ώστε να μπορεί αυτή να απεικονίσει οποιονδήποτε προσανατολισμένο γράφο, με καλό αισθητικά τρόπο, κρίνεται επιθυμητή. Κάτι τέτοιο θα μπορέσει να καλύψει την παρουσίαση πολυ-ιεραρχιών, όποτε αυτό χρειαστεί για το παρόν σύστημα. Για το σκοπό αυτό ίσως μπορέσει να χρησιμοποιηθεί ο αλγόριθμος που περιγράφεται στο [51] με αποσύνδεση του κώδικα αυτού από το σύστημα GAIN και τυποποίηση της εισόδου και της εξόδου του. Αν αυτό ωστόσο δεν είναι εφικτό μπορεί να χρησιμοποιηθούν και έτοιμα εργαλεία δημιουργίας γράφων που παίρνουν τυποποιημένη είσοδο και βγάζουν έξοδο σε γνωστά format όπως .jpg, gif, .ps, .vrml και .svg! Ένα απ' αυτά είναι το Graphviz [55].

Τροποποιήσεις των λειτουργιών οι οποίες χειρίζονται ημερομηνίες, και οι οποίες προς το παρόν παρίστανται ως διαστήματα ακεραίων (από έτος, έως έτος), είναι επιβεβλημένες. Τέτοιες αλλαγές θα βελτιώσουν τις επιδόσεις τόσο των λειτουργιών αναζήτησης που σχετίζονται με ημερομηνίες, όσο και της λειτουργίας κατανομής έργων τέχνης βάσει χρονικών περιόδων, υλικών υποστήριξης κλπ. – λειτουργία Paintings –. Η τελευταία λειτουργία μπορεί να βελτιωθεί ακόμα περισσότερο, αν γίνουν πρώτα οι προηγούμενες αλλαγές, με εκμετάλλευση του link *in period* που ξεκινά από την ιδιότητα *production date* του Man-Made Object και καταλήγει σε Period. Αυτό θα την απαλλάξει

από την επαναληπτική διαδικασία εντοπισμού της χρονικής περιόδου που ανήκει ένα έργο τέχνης βάσει του *production date*. Η μη χρήση του πρωτογενούς τύπου δεδομένων Telos_Time στην υλοποίηση του παρόντος συστήματος, οφείλεται στην μη ύπαρξη αντίστοιχης κλάσης Time στο Java API του SIS. Θεωρούμε ότι αυτή η βελτίωση του API είναι απολύτως απαραίτητη.

Η σημαντικότερη επέκταση για την ολοκλήρωση του παρόντος ΠΣΒΠΙ είναι η υλοποίηση του τμήματος εκείνου που θα αναλάβει την εισαγωγή, ανανέωση και διαγραφή δεδομένων του συστήματος. Το τμήμα αυτό θα πρέπει να βρίσκεται εγκατεστημένο σε διαχειριστικούς σταθμούς εργασίας στο τοπικό δίκτυο όπου τρέχει ο εξυπηρετητής του SIS. Σύμφωνα με τις προτάσεις του κεφαλαίου 5 οι σταθμοί εργασίας πρέπει να είναι εφοδιασμένοι με επεξεργαστή XML (όπως το XMetal), με το σύνολο των ορισμένων για καταχώρηση DTD και το σύνολο των κλάσεων Java που θα αναλαμβάνουν την ενημέρωση του SIS. Οι κλάσεις θα πρέπει να χρησιμοποιούν την πολυνηματική έκδοση του API του SIS για να μπορούν να συμβαίνουν ταυτόχρονες ενημερώσεις του συστήματος από πολλούς διαχειριστές.

Η υποδομή που υπάρχει στο σύστημα για πολλαπλές παρουσιάσεις κάνει χρήσιμη την ύπαρξη ενός εργαλείου δημιουργίας των παρουσιάσεων αυτών με απλό τρόπο. Ένα τέτοιο εργαλείο θα βοηθούσε στον καθορισμό της φόρμας μέσα στην οποία θα εμφανίζονται τα αποτελέσματα των λειτουργιών, όπως για παράδειγμα θέση του μενού, σημείο και τρόπο αλλαγής γλώσσας, θέση banner κλπ. Επίσης θα μπορούσε να αναδιατάσσει τη σειρά και να αλλάζει τη θέση παρουσίασης των αποτελεσμάτων λειτουργιών. Για παράδειγμα στη λειτουργία εμφάνισης πληροφοριών όρου θα μπορούσε να δείξει την indented list με τη θέση του όρου στην ιεραρχία από την αριστερή πλευρά κι όχι από τη δεξιά, το scopenote τελευταίο κι όχι πρώτο κ.ο.κ. Το εργαλείο αυτό θα ήταν καλό να παρέχει εξωτερικά ένα εύχρηστο γραφικό περιβάλλον και η εσωτερική του λειτουργία προφανώς θα ήταν να παράγει τον πηγαίο κώδικα για το σύνολο των XSL files όλων των λειτουργιών της δικτυακής εφαρμογής (όπως στο [16]).

Η εύκολη δημιουργία πολλαπλών παρουσιάσεων μπορεί εύκολα να οδηγήσει το σύστημα στην εξατομίκευση όψεων. Θα μπορούσε δηλαδή ο κάθε χρήστης κατά την πρώτη επίσκεψη του στο δικτυακό τόπο του συστήματος να δημιουργεί ένα προφίλ χρήστη. Μια από τις παραμέτρους του προφίλ θα μπορούσε να είναι και το template παρουσίασης που επιθυμεί (με επιλογή από λίστα). Έτσι στο εξής σε κάθε επίσκεψη του χρήστη, μετά την ταυτοποίησή του απ' το σύστημα, εκείνος θα βλέπει τη δικτυακή εφαρμογή σύμφωνα με τις δικές του επιλογές.

Παράρτημα Α – Συντομογραφίες

AL	Assertional Language
API	Application Programming Interface
C2RMF	Centre de Recherche et de Restauration des Musees de France
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CREBITEL	Conservation REstoration BIlingual Training ELecrtonic handbook
CRISTAL	Conservation & Restoration Institutions for Scientific Terminology dedicated to Art Learning Network
DBMS	DataBase Management System
DHTML	Dynamic HyperText Markup Language
DOM	Document Object Model
DSS	Decision Support System
DTD	Document Type Definition
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IT	Information Technology
J2EE	Java 2 platform Enterprise Edition
JDBC	Java DataBase Connectivity
JSP	Java Server Pages
JVM	Java Virtual Machine
KMS	Knowledge Management System
NARCISSE	Network of Art Research Computer Image SystemS in Europe
ODBC	Open DataBase Connectivity
RDBMS	Relational DataBase Management System
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SIS	Semantic Index System
SVG	Scalable Vector Graphics
TMS	Thesaurus Management System
UI	User Interface
URL	Unified Resource Location
WBIS	Web-Based Information System
WWW	World Wide Web
XML	eXtensible Markup Language

XSL	eXtensible Stylesheet Language
ΕΛ	Ενδιάμεσο Λογισμικό
ΕΛΒΔ	Ενδιάμεσο Λογισμικό για Βάσεις Δεδομένων
ΠΙ	Παγκόσμιος Ιστός
ΠΣΒΠΙ	Πληροφοριακό Σύστημα Βασισμένο στον Παγκόσμιο Ιστό
ΣΔΒΔ	Σύστημα Διαχείρισης Βάσεων Δεδομένων
ΣΔΘ	Σύστημα Διαχείρισης Θησαυρών
ΣΔΘΟ	Σύστημα Διαχείρισης Θησαυρού Όρων
ΣΣΔΒΔ	Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων
ΣΣΕ	Σύστημα Σημασιολογικού Ευρετηριασμού

Παράρτημα Β – Αγγλο-ελληνικό γλωσσάριο

Alteration	Αλλοίωση
Application Programming Interface	Διεπαφή Προγραμματισμού Εφαρμογής
application server	εξυπηρετητής εφαρμογής
Assertional Language	Δηλωτική Γλώσσα
Assessment	εκτίμηση
caching	εκμετάλλευση λανθάνουσας μνήμης, για το SIS εκμετάλλευση συνόλων με προσωρινά αποτελέσματα που βρίσκονται στη μνήμη
client	πελάτης
client-server architecture	αρχιτεκτονική πελάτη-εξυπηρετητή
Conservation	Συντήρηση
Conservation Report	Δελτίο Συντήρησης
constraint	περιορισμός ακεραιότητας
daemon	δαίμονας
deductive rule	επαγωγικός κανόνας
Descriptor	Περιγραφέας - Χρησιμοποιείται για τους όρους του θησαυρού
desktop environment	περιβάλλον εργασίας
dispatcher	διανεμητής
driver	οδηγός
effectiveness	αποτελεσματικότητα
efficiency	αποδοτικότητα
Elaboration	Τεχνική Κατασκευής
format	μορφότυπος
Graphical User Interface	Γραφική Διεπαφή Χρήστη
host	υπολογιστής δικτύου
Information Technology	Πληροφορική Τεχνολογία
instance	στιγμιότυπο
instantiation	υποστασιοποίηση
Internet	Διαδίκτυο
load balancing	στάθμιση φόρτου
Man-Made Object	ανθρώπινο κατασκεύασμα
Material	Υλικό
Method	Μέθοδος
Middleware	Ενδιάμεσο Λογισμικό
modularity	στοιχειομέρεια
module	στοιχείο
multilinguality	πολυγλωσσία

multi-presentation	πολλαπλές παρουσιάσεις
multispectral	πολυφασματικός
multithread	πολυνηματικός
multi-user	πολλών χρηστών
network congestion	συμφόρηση δικτύου
persistent	μόνιμος
port	πόρτα
Remote Procedure Call	Απομακρυσμένη Κλήση Διαδικασιών
scalability	κλιμάκωση
server	εξυπηρετητής
simplicity	απλότητα
tag	ετικέτα διαμόρφωσης
User Interface	Διεπαφή Χρήστη
web-application	εφαρμογή Παγκόσμιου Ιστού
web-browser	φυλλομετρητής του Παγκόσμιου Ιστού
web-programming	προγραμματισμός Παγκόσμιου Ιστού
web-server	εξυπηρετητής του Παγκόσμιου Ιστού
works-of-art	έργα τέχνης
wrapper	μεταφραστής

Παράρτημα Γ – Στήσιμο Εφαρμογής

Η εφαρμογή έχει εγκατασταθεί επιτυχώς στις πλατφόρμες Windows 98, Windows NT 4.0 και Windows 2000. Ακολουθούν οι οδηγίες για εγκατάστασή της σε Windows 2000 μαζί με περιβάλλον ανάπτυξης για περαιτέρω επεκτάσεις. Τα αρχεία και οι κατάλογοι που αναφέρονται στις οδηγίες υπάρχουν στον κατάλογο-ρίζα του CDROM που συνοδεύει την παρούσα εργασία.

1. Εγκατάσταση του Java 2 Software Development Kit **(j2sdk-1_3_1_01-win.exe**) στον κατάλογο **X:\jdk1.3.1_01** (X : γράμμα του partition του σκληρού δίσκου)

2. Εγκατάσταση του Apache Tomcat 4 **(jakarta-tomcat-4.0.1.exe**) στον κατάλογο **X:\Program Files\Apache Tomcat 4.0**. Προσοχή να γίνει πλήρης εγκατάσταση "Full(w/Source Code)" ώστε ο εξυπηρετητής εφαρμογής να εκτελείται σαν service

3. Αποσυμπίεση (extraction) του αρχείου **jakarta-ant-1.5-bin.zip** στον κατάλογο **X:\Program Files\Apache Tomcat 4.0**

4. Αντιγραφή του καταλόγου **CRISATEL_src** στον κατάλογο **X:\Program Files\Apache Tomcat 4.0\webapps**

5. Ανάθεση των εξής μεταβλητών περιβάλλοντος

JAVA_HOME	**X:\jdk1.3.1_01**
JAVA_BIN	**%JAVA_HOME%\bin**
JAVA_LIB	**%JAVA_HOME%\lib**
JAVA_INCLUDE	**%JAVA_HOME%\include**
CATALINA_HOME	**X:\Program Files\Apache Tomcat 4.0**
Path	**%Path%;%JAVA_BIN%;%JAVA_LIB%;%JAVA_INCLUDE%; %CATALINA_HOME%\jakarta-ant-1.5\bin**

6. Αντιγραφή των **jar-dll\japi13.jar** και **jar-dll\crimson.jar** στον κατάλογο **X:\Program Files\Apache Tomcat 4.0\common\lib**

7. Μεταγλώττιση της εφαρμογής μέσω γραμμής εντολών και μέσα από τον κατάλογο **X:\Program Files\Apache Tomcat 4.0\webapps\CRISATEL_src** με την εντολή **ant deploy**

8. Αντιγραφή των αρχείων **cw3220.dll, Qclass.dll, tmsapi_dll.dll, TMSAPIClass.dll** στον κατάλογο **\WINNT\SYSTEM32**

9. Αντιγραφή των αρχείων **config.dtd** και **config.xml** από τον κατάλογο **X:\Program Files\Apache Tomcat 4.0\webapps\crisatel** στον κατάλογο **\WINNT\SYSTEM32** αφού γίνει αλλαγή των μεταβλητών **webapp_machine**, **sis_machine** και **webapp_local_path** του **config.xml** στις κατάλληλες τιμές (IP διευθύνσεις είναι προτιμητέες από hostname). Προσοχή αν ο Tomcat θα τρέχει από κονσόλα κι όχι σαν service τότε τα **config.xml** και **config.dtd** θα πρέπει να τοποθετηθούν στον κατάλογο **X:\Program Files\Apache Tomcat 4.0**

10. Εγκατάσταση του SIS-TMS Development Kit v1.4.2.1 (κατάλογος **TMS-1-4-2-1-01**) στον κατάλογο **X:\ICS-FORTH\TMS** . Στο port που ζητείται βάλτε όποιο θέλετε να χρησιμοποιεί η εφαρμογή (**έστω 1201**) αρκεί να μη χρησιμοποιείται από κάποια άλλη

11. Αντιγραφή του καταλόγου **TMS-1-4-2-1-01\DeveloperSIS2.2** στον κατάλογο **X:\ICS-FORTH\TMS**

12. Αντιγραφή του καταλόγου **CRISATEL** στον κατάλογο **X:\ICS-FORTH\TMS\Applications**

13. Επεξεργασία του αρχείου **X:\ICS-FORTH\TMS\Applications\CRISATEL\remodel.bat** . Κατάλληλες ανάθεση στις γραμμές 1,2,3 των **SIS_PATH**, **SIS_PORT**, **MACHINE_NAME**. Με εκτέλεση του αρχείου **remodel.bat** μέσα από το φάκελο **X:\ICS-FORTH\TMS\Applications\CRISATEL** γίνεται επαναδημιουργία της βάσης, βάσει των περιεχομένων (Telos) των αρχείων model.tls, data.tls, data2.tls, data3.tls, data4.tls, data5.tls και data6.tls. Στο αρχείο model.tls περιέχεται ο κώδικας του μοντέλου, στο data.tls ο κώδικας για την καταχώρηση των όρων του θησαυρού του Narcisse, στο data2.tls ο κώδικας για τις μεταφράσεις κάποιων όρων, στα data3.tls, data4.tls, data5.tls ο κώδικας για την καταχώρηση 3 έργων τέχνης και στο data6.tls υπάρχει κώδικας για την καταχώρηση στιγμιοτύπων γνώσης του πεδίου.

14. Δημιουργία αντιγράφων των αρχείων **X:\ICS-FORTH\TMS\TMSServer.bat , X:\ICS-FORTH\TMS\TMSGain.bat** και **X:\ICS-FORTH\TMS\env\TMSSetup.bat** στους ίδιους καταλόγους και με αντίστοιχα ονόματα που στη θέση του "TMS" θα έχουν "CRISATEL". Επεξεργασία των αρχείων αυτών και αλλαγή του pattern TMS σε CRISATEL παντού. Εκτέλεση του **X:\ICS-FORTH\TMS\CRISATELServer.bat** πλέον ξεκινάει τον SISServer για τη βάση του συστήματος.

15. Εγκατάσταση στο μηχάνημα-client του Internet Explorer 6.0 : **Internet Explorer 6.0\ie6setup.exe**
    Επίσης εγκατάσταση του Adobe SVG viewer : **SVGView.exe**

16. Από τη γραμμή διεύθυνσης του Internet Explorer 6.0 εκτελούμε το servlet
    http://serverip:8080/crisatel/ProduceHierarchy?LID=[lid]&TID=[tid]&hierarchy=[topterm]&mode=[mode]
    για όλους τους συνδυασμούς lid={0,1}, tid={0,1} και topterm={elaboration, alteration, treatment, material and elements}. Το mode είναι 0 για όλες αυτές τις περιπτώσεις.
    Το ίδιο servlet πρέπει να τρέξει με topterm={method} και mode=1 για όλους τους συνδυασμούς και πάλι των lid,tid. Συνολικά το servlet πρέπει να τρέξει 20 φορές.

17. Η εφαρμογή της παρούσας εργασίας μπορεί να προσπελαστεί πλέον με το URL:
    http://serverip:8080/crisatel/Index

Βιβλιογραφία

[1]. G. Doukidis. *Information Technologies: Concepts & Management.* Department of Management Science & Technology, Athens University of Economics and Business. April 2000

[2]. Journal of Research and Practice in Information Technology. *Special Issue on Web-Based Information Systems*. M. Schrefl and J. Roddick (Eds.), Vol 34, Issue 1, 2002

[3]. J. Painter and M.Pearson. *An analysis of WWW-based Information Systems*. Chow, W.S. (ed.), Multimedia Information Systems in Practise, Springer, Singapore, 53-63, 1998

[4]. Software Engineering Institute (SEI), Carnegie Melon University**.** *Client/Server Software Architectures--An Overview*. Software Technology Review. Available at <URL: http://www.sei.cmu.edu/str/descriptions/clientserver_body.html>(Last Modified: September 2000)

[5]. Αικατερίνη Χ. Τόλιου. *Πύλη Διασύνδεσης του Παγκόσμιου Ιστού με το Σύστημα Σημασιολογικού Ευρετηριασμού*. Μεταπτυχιακή Εργασία, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, Οκτώβριος 1998.

[6]. Philip A Bernstein. *Middleware: A Model for Distributed System Services*. Communications of the ACM 39, 2 (February 1996): 86-97.

[7]. Richard Schreiber. *Middleware Demystified.* Datamation 41,6 (April 1995):41-45

[8]. George Schussel. *Client/Server Past, Present, and Future.* Available at <URL: http://www.dciexpo.com/geos/> (1995)

[9]. Wayne W. Eckerson. *Three Tier Client/Server Architecture: Achieving Scalability, Performance and Efficiency in Client Server Applications.* Open Information Systems 10, 1 (January 1995): 3(20)

[10]. Institute of Computer Science, Foundation for Research and Technology, Hellas (ICS-FORTH). *SIS – Data Entry Language: User's Manual*, version 2.2, 1998

[11]. Panos Constantopoulos and Martin Doerr. *The Semantic Index System.* ICS-FORTH, May 1993

[12]. Apache Software Foundation. *The Apache Jakarta Project – Apache Tomcat*. Available at <URL: http://jakarta.apache.org/tomcat/tomcat-4.0-doc/index.html>

[13]. Institute of Computer Science, Foundation for Research and Technology, Hellas (ICS-FORTH). *Narcisse – Network of Art Research Computer Image SystemS in Europe*. Available at <URL: http://zeus.ics.forth.gr/forth/ics/isl/projects/cristal-french/narcisse.html>

[14]. Institute of Computer Science, Foundation for Research and Technology, Hellas (ICS-FORTH). *CRISTAL – Conservation & Restoration Institutions for Scientific Terminology dedicated to Art Learning Network*. Available at <URL:http://www.ics.forth.gr/isl/projects/ projects_individual.jsp?ProjectID=5>

[15]. C. Lahanier, G. Aitken, P. Cotte, J. Dufresne, R.Pillay, J. Shindo, J. Stevenson. *Information Technologies developed for Museum Collections.* 2002 Euro-China Co-operation Forum on the Information Society, May 2002

[16]. Jen-Shin Hong, Bai-Hsuen Chen, Jien Hsiang. *XSL-based Content Management for Multi-presentation Digital Museum Exhibition.* P. Constantopoulos and I.T. Solvberg (Eds) LNKCS 2163, pp. 378-389, 2001

[17]. European Commission – The Fifth Framework Programme**.** *CRISATEL-Conservation Restoration Innovation Systems for image capture and digital Archiving to enhance Training Education and lifelong Learning*. Proposal number: IST-1999-20163, pp. 1-2, January 2000

[18]. J. Mylopoulos, A. Borgida, M. Jarke and M. Koubarakis. *Telos: Representing Knowledge About Information Systems*. ACM Transactions on Information Systems, 8(4):325-362, October 1990

[19]. S. Greenspan. Requirements Modelling: A Knowledge Representation Approach to Requirements Definition. Ph.D. thesis, Department of Computer Science, University of Toronto, 1994

[20]. M. Stanley. CML: A Knowledge Representation Language with Application to Requirements Modelling. M.Sc. thesis, Department of Computer Science, University of Toronto, 1986

[21]. InformatikV, Aachen University of Technology (RWTH), Germany. *Concept Base V5.0 User Manual*. M. Jarke, M. A. Jeusfeld, C. Quix (Eds). March 2000

[22]. Πάνος Κωνσταντόπουλος. *Επισκόπηση της μοντελοποίησης πληροφοριών, Σύντομες σημειώσεις*. ΗΥ-465, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, Μάρτιος 2000

[23]. Institute of Computer Science, Foundation for Research and Technology, Hellas (ICS-FORTH). *SIS–Application Programmatic Interface:Reference Manual* v2.2.1, June 2002

[24]. Pyung-Chul Kim. *A Taxonomy on the Architecture of Database Gateways for the Web.* In Proceedings of the 13th International Conference on Advanced Science and Technology. (ICAST97), Schaumburg, IL, 1997

[25]. Open Market Inc. *FastCGI: A High-Performance Web Server Interface.* Available at <URL: http://www.fastcgi.com/devkit/doc/fastcgi-whitepaper/fastcgi.htm> (April 1996)

[26]. Netscape Communications Corporation. *DevEdge Newsgroup FAQ: NSAPI*. DevEdge Online Archive. Available at <URL: http://developer.netscape.com/support/faqs/champions/nsapi.html> (September 1998)

[27]. Microsoft Corporation. *ISAPI and the Web Application Architecture.* MSDN Library. Available at <URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/iisref/html/psdk/asp/isgu80v9.asp>

[28]. Barry & Associates Inc. *Application server definition.* Available at <URL: http://www.Java-application-servers.com/articles/application_server_definition.html> (Last Modified: October 2001)

[29]. GBdirect Ltd, Training Division, United Kingdom. *Active Web Sites and Comparison of Scripting Languages.* Available at <URL: http://training.gbdirect.co.uk/courses/php/comparison_php_versus_perl_vs_asp_jsp_vs_vbscript_web_scripting.html>

[30]. M. Pelletier, A. Latteier, C. McDonough. *The Zope Developer's Guide (Zope 2.4 Edition)*. Zope Corporation . Available at <URL: http://www.zope.org/Documentation/Books/ZDG/current/index_html> (May 2001)

[31]. BEA Systems, Inc. *BEA WebLogic Server Overview White Paper*. December 2002

[32]. Borland Software Corporation. *Borland Enterprise Server, AppServer Edition features and benefits*. June 2002

[33]. IBM Corporation Software Group. *IBM WebSphere Application Server Version 4.1, Enterprise Edition*. March 2002

[34]. Andreas Schaefer. *JBoss 3.0 Quick Start Guide.* JBoss Group, USA, July 2002

[35]. Macromedia Inc. *Macromedia JRUN 4 Overview datasheet.* Available at <URL: http://www.macromedia.com/software/jrun/whitepapers/pdf/jr4_product_overview.pdf>

[36]. Craig R. McClanahan. *Catalina – A Proposed Architecture for Tomcat.* Available at <URL: http://www.ingrid.org/jajakarta/tomcat/tomcat-4.0b5/src/catalina/docs/dev/proposal.html> (Last Revised: January 2000)

[37]. David M. Johnson. *UML Diagram : Tomcat 4.0 – Overview.* Available at <URL: http://relayirc.sourceforge.net/catalina/> (October 2000)

[38]. Danny Coward. *Java Servlet Specification Version 2.3, Final Release.* Sun Microsystems. September 2001.

[39]. Refsnes Data. *XML Tutorial – Introduction to XML.* W3Schools.com. Available at <URL: http://www.w3schools.com/xml/xml_whatis.asp>

[40]. World Wide Web Consortium. *Scalable Vector Graphics (SVG) 1.0 Specification. W3C Recommendation 04 September 2001.* Available at <URL: http://www.w3.org/TR/2001/REC-SVG-20010904/> (September 2001)

[41]. Adobe Systems Incorporated. *SVG Zone – Overview, What is SVG?* Available at <URL: http://www.adobe.com/svg/overview/svg.html>

[42]. Adobe Systems Incorporated. *Adobe SVG Viewer for Windows. Release Notes. Version 3.0 (Build 76).* November 2001.

[43]. World Wide Web Consortium. *Extensible Stylesheet Language (XSL) Version 1.0 W3C Recommendation 15 October 2001.* Available at <URL: http://www.w3.org/TR/2001/REC-xsl-20011015/> (October 2001)

[44]. Refsnes Data. *XHTML Tutorial – XHTML Introduction.* W3Schools.com. Available at <URL: http://www.w3schools.com/xhtml/default.asp>

[45]. Refsnes Data. *DHTML Tutorial –Introduction to DHTML.* W3Schools.com. Available at <URL: http://www.w3schools.com/dhtml/dhtml_intro.asp>

[46]. Miquel Angel Pintanel. *mapbMenu Version 0.8 documentation.* mapbDHTML website. Available at <URL: http://perso.wanadoo.es/mapintanel/mapbdhtml/menu.html> (Last Modified: June 2002)

[47]. Pascal Forget. *Apache 1.3.23 + Tomcat 4.0.2 + Load Balancing*. UBEANS Technologies. Available at <URL: http://www.ubeans.com/tomcat/> (Last Modified: January 2002)

[48]. M. Doerr and I. Fundulaki. *SIS-TMS A Thesaurus Management System for Distributed Digital Collections.* Proceedings of the Second European Conference for Digital Libraries ECDL'98 , Crete Greece. September 1998.

[49]. Dimitris Plexousakis. *Integrity Constraint and Rule Maintenance in Temporal Deductive Knowledge Bases.* VLDB-93.

[50]. John Mylopoulos. *Conceptual Modelling and Telos,* Technical Report DKBS-TR-91-3, Department of Computer Science, University of Toronto. November 1991.

[51]. Μαρία Καραβασίλη. *Ένας Αλγόριθμος για το Σχεδιασμό Προσανατολισμένων Γράφων.* Μεταπτυχιακή Εργασία, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, Φεβρουάριος 1994.

[52]. Institute of Computer Science, Foundation for Research and Technology, Hellas (ICS-FORTH). *SIS – Graphical Analysis Interface:User's Manual, version 2.3*, November 2000

[53]. A. Misargopoulos, G. Kontolemakis, M. Masvoula. *Thesaurus Management using XML version 1.2*, Technical Report, Institute of Computer Science Foundation for Research and Technology – Hellas (ICS-FORTH), October 2001

[54]. Institute of Computer Science, Foundation for Research and Technology, Hellas (ICS-FORTH). *SIS – Entry Form:User's Manual, version 2.3*, September 1999

[55]. AT&T Labs – Research. *Graphviz - open source graph drawing software.* Available at <URL: http://www.research.att.com/sw/tools/graphviz/> (Last Modified: October 2000)