A Practitioner's Approach to Evolving and Remodeling Large-Scale WWW Sites

Paris A. Zafiris, Nektarios P. Georgantis, George E. Kalamaras, Sotiris P. Christodoulou
and Theodore S. Papatheodorou
High Performance Computing Lab, Computer and Informatics Engineering Dept.,
University of Patras, Rion, Patras, 26500, Greece
{paz, npg, gek, spc, tsp}@hpclab.ceid.upatras.gr

Abstract

This paper addresses the issues of evolving large-scale WWW sites, in order to migrate to more flexible forms of developing and managing content and applications. We first present an integrated process model for the life-cycle of a WWW site. Next, we focus on specific steps of this model to (i) examine a layered, next-generation architecture for large WWW sites, providing for content re-use and easier implementation of services and applications and (ii) to discuss the activities of remodeling our existing content infrastructure and applications, according to the specifications of this target architecture. In our case study, we apply the new architecture, through the remodeling activities, in the evolution of the official site of the Hellenic Ministry of Culture.

Keywords: Architectures**,** design, development, evolution, hypermedia, methodology, models, process, remodeling, systems, web

1. Introduction

It is widely recognized that most Web site installations, even large-scale ones, are still using obsolete technologies for content and applications management, as well as insufficient processes for carrying out their operation and maintenance. A simple example is the widespread coupling of content and applications through HTML, that is posing numerous problems for maintainability and extensibility. Therefore, a speedy evolution and the establishment of more systematic approaches for managing information, applications and operations is considered especially important.

In this work we are addressing the issues of major, transparent evolution of large scale WWW infosystems, an activity which involves significant remodeling of their core components. By the term "evolution" we may refer to

- the regular maintenance of the site, which can be perfective, adaptive, corrective, speculative, as in other software systems [15]
- a more extended restructuring and enhancement. This not only prevents the degradation of the site, but allows existing installations to maintain acceptable levels of manageability, to safely avoid the dangers of insufficient or immature technologies and to prepare for the future of online computing. Obviously, it is an action that should be performed using a careful and thorough approach, *at least once* in the life of any Web Site and even more often at large installations with increased complexity and potential points of failure.

In this paper we emphasize on the second case. Since this kind of evolution occurs at less frequent intervals, it is akin to migrating to a next generation of the site and requires serious remodeling of the overall architecture, information infrastructure (*infostructure*), applications/ services, system and user interfaces.

Remodeling, as a core step for a major evolution process, includes

- the definition of a next-generation site architecture, especially focusing on the management of content and applications
- the migration procedures from the existing state towards the new environment

In our approach, we first present an *integrated process model* for the life-cycle of large-scale Web sites, consisting of design, implementation, maintenance and evolution steps. We then focus on *two specific steps* of the process model:

(i) the introduction of a *layered architecture* for viable content management and easier deployment of application/services on-top-of the integrated content. Key notion is the proper (re)modeling of information, in different repositories, and its migration to a more streamlined "form" (including structure and semantics), which allows for easier maintenance and expansion and is the basis for easier applications and services development.

(ii) the *actual remodeling* of content repositories and applications into more flexible forms, according to the specifications of the target architecture. This is done in a systematic fashion, so that existing large sites can continue their operation and perform their evolution without sacrificing functionality.

Several other solutions and ideas appear naturally within the framework of our contribution but are not pursued further, in order to maintain focus.

Our approach borrows heavily from (i) extended practical experience in the maintenance and evolution of Web Sites and (ii) from our research on more systematic approaches to both Web content and hypermedia applications management [16,20].

Research efforts directly on the evolution of Web sites focus on analyzing the changes throughout the new versions of the sites and characterizing evolution in terms of metrics and models [15]. Other work on the evolution of web sites, as hypermedia systems, can be derived from academic research on well-known design models and methods for hypermedia and Web maintenance (e.g. HDM [6], RMM [21], HDM-Lite [13], Araneus [12], [14], [7]). Unfortunately, most of them are limited in actual adoption, primarily due to their complexity and prescriptive nature [4]. In the meantime, although technologies and commercial tools for developing large-scale and complex WWW sites appear often, they have several significant shortcomings and are also plagued by insufficient methodologies [16, 20].

Realizing these problems, new efforts are in place to design open-architecture and scalable systems for flexible development/ management of content and applications (e.g. CBD), while emphasizing on their applicability to practical situations. Several platforms have been introduced by research institutes and commercial organizations within the last year, which are indicative of the trends towards extensible systems:

- Web Modeling Language (WebML) [19], which is a notation for specifying complex Web sites at the conceptual level. WebML enables the high-level description of a Web site along multiple dimensions: its data content (structural model), the pages that compose it (composition model), the topology of links between pages (navigation model), the layout and graphic requirements for page rendering (presentation model), and the customization features for one-to-one content delivery (personalization model).
- The WebComposition System [11], which is mainly addressing the lack of current Web implementation models to provide abstractions for higher level design-concepts. The system is based on a fine-grained Object-Oriented web application model. Services are logical units or business processes represented as groups of WebComposition components. Automatic creation for this class of components is supported using service factories which are modeled on creational design patterns.
- Percussion's Rhythmyx Content Management System [17], which primarily addresses the challenge of maintaining effective Web content. It employs XML [28] /XSL [29] as the core technologies for describing Web content and rendering output for variable browsers and access devices. XML is used to build a "virtual" content layer that combines different underlying content repositories by mapping the global XML schemata [27] (or DTDs) of the content layer to actual content database (DB) tables and columns. Therefore, it offers the view of a single repository, with business logic (vocabularies, workflow metadata, security, content metadata) bound to it. Rhythmyx contains solutions for content entry and development, as well as for content delivery, with the main Rhytmyx server being a Java application, that applies business rules, security, workflow metadata to allow content to be processed by authors & approvers.

Our approach shares a number of ideas and concepts with these platforms, as will become obvious in the remainder of the paper which is structured as follows: In Section 2 we present an integrated process model, which provides a roadmap for maintaining and evolving large-scale Web Sites. Section 3 focuses on the open target architecture which is a basic component on the overall success of the site's evolution. The specific tasks for remodeling existing content/applications and for migrating to this architecture are detailed in Section 4. Section 5 provides an overview of a case study of our approach on the site of the Hellenic Ministry of Culture, ULYSSES. Our conclusions are presented in Section 6.

2. A process model for maintenance and evolution of Web sites

In Fig. 1 we present a generic process model[1] for designing, implementing, maintaining and evolving large-scale sites, which we employ at governmental Web installations maintained by our research laboratory.

The process model acts as a blueprint for the tasks that are taking place in the various steps of a life-cycle of a large Web site. The model we outline is not exhaustive, but focuses on the most important activities carried out and the relationships among them.

Although it resembles a waterfall [25]-like model with prototypes, we must point out that various individual activities (such as implementation of applications) may by themselves require complex models, e.g. spiral [2], iterative enhancement [24]. This is indicative of the complexity of operations in a Web site and explains the frequent overlapping which can occur between tasks, causing potential problems in execution and management [10]. In practice, for instance, requirements and design

[1] A (software) process model is a framework within which project-specific software processes are defined. It establishes the structure, standards and relationships of the various process elements.

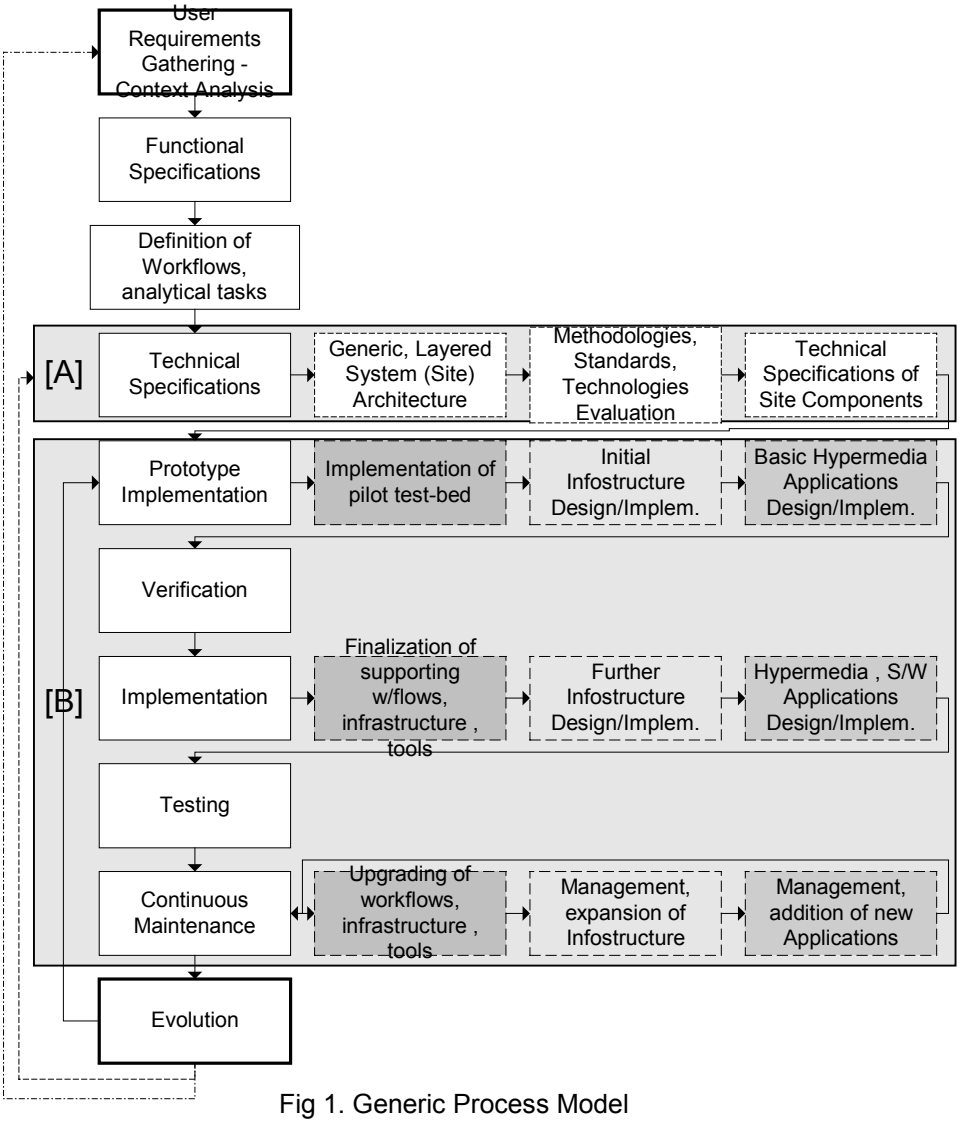

Fig 1. Generic Process Model

may still live throughout the development tasks and therefore should be constantly updated.

As illustrated above, the outcome of the "evolution" task, depending on the extend of changes/remodeling required, is to branch (i) to implementing a prototype of our revised content or application (regular maintenance), (ii) to revising our technical specifications, our architecture and site components, or even (iii) to fully reviewing the user requirements, preparing for a massive overhaul of our site. In this work we are examining scenario (ii).

For the process model presented above, we are going to bypass the analysis of the first steps, since numerous publications exist on these topics, and focus directly on the revised architecture and the remodeling tasks. We will briefly list a compact collection of common requirements of the various "stakeholders" participating in the lifecycle of a Web Site, since this is used to derive functional and technical specifications for the development of our architecture and supporting information technology. In Table 1 we summarize (from [16], [20]) and extend the requirements (i) of developers (DIT) of the supporting Information Technology environments for content and overall Web site management, of content authors (CA), (ii) of developers of end-user applications/services - including graphics artists (DEUA) and (iii) of end users (EU). These are the roles involved both in design issues and in the day-to-day workflow of a Web site and are critical in properly designing, maintaining and evolving a Web Site:

Table 1. Stakeholders Requirements in Web Sites

Requirements / Roles	DIT	CA	DEUA	EU
Scalable architecture / services of the supporting WWW-IT	x		x	
Clear and well-defined modules / interfaces for module interoperability	x		x	
Integration with existing infrastructure	x			
Adaptability to new standards and tools	x			
Easy-to-use tools and applications		x	x	x
Re-usable content		x		x
Support for multilingual content		x		x
Easy creation & maintenance of content		x		
Support for "legacy" content		x		
Scalable and extensible content		x		
Flexible content definitions, variety of vocabularies		x		x
Streamlined workflow and methodology for content creation and maintenance		x		
Online interfaces, for remote/distributed access		x		
Integrated and seamless access to heterogeneous material		x	x	
Security and access control mechanisms		x	x	
Clear interfaces and mechanisms for application development on-top-of the content			x	
Support for multiple developers		x	x	
Collaboration mechanisms		x	x	
Fast, consistent and reliable information				x
Customizable views into the content				x
Dynamic delivery of information				x
Accessibility through varied terminals/browsers				x

3. Target site architecture

As stated in the previous sections, especially important for the success of the site evolution is the definition of an extensible new architecture. Thus we focus on Stage [A] of the generic process model (Fig. 1) and, more specifically, on the properties of the target, layered site architecture.

Our architecture follows closely the guidelines of an integrated framework for modeling and management of content, which was originally outlined in [16, 20]. It attempts to satisfy the user requirements of the various stakeholders presented in Section 2, as well as functional and technical specifications which can be derived thereof.

Characteristic technical specifications of the architecture include the decoupling of content and its structure from the business logic and the presentation; the migration towards repositories of structured information (vs. unstructured documents); the support of core standards for content modeling, metadata and ontologies; the integration of specific workflows for managing content; and the support for distributed content authoring / management.

We focus on scalability and transparency which will ease the development of new or remodeling of existing applications and services. Our objective is to create a "pool" of re-usable and flexible content, independent of the repository technologies employed, on-top-of which a layer of metadata offers repositories' integration and consistent access from applications and services.

Therefore, we aim for an open architecture, which consists of three layers, as displayed in Fig. 2 in the next page:

- The Data Repositories layer
- The Metadata layer (which, along with the Data Repositories layer, forms the Information Repository)
- The Application and Services layer

The *Data Repositories (DR) layer* is characterized by its heterogeneity, as is obvious in all Web sites and especially the large ones. We can classify the various DR into Databases (fully-structured info), collections of structured documents (semi-structured info) and collections of un-structured media (text files, HTML files, audio, video, etc.).

Remodeling of data repositories equals the speedy migration from un-structured (current situation) towards semi- or fully-structured forms of information. Our aim is to convert all content categories which are document-based to classes of XML files, following their careful contextual analysis (semantics, domain and application ontologies) and the specification of their syntax/structure using DTDs or XML Schemata. In effect, we have classes of documents, with each class representing a specific category of information. XML is our core technology, due to its almost universal acceptance as an open standard, its flexibility, support of localization, and the abilities it provides to implement text documents which are both human- and machine-understandable.

In remodeling of data repositories, into collections (classes) of XML documents, although this is more costly, attention should be paid to leveraging application domain ontologies. This allows us to have a more complete perspective of approaches to describing our content, although it is natural in most cases to revert to over-simplified versions.

Unstructured data can also be migrated into DBs, which are (a) exceptionally flexible in issues of modeling information structure, (b) supporting constraints, (c) tools for management and especially performance; however, this process is usually too costly in human effort or time required.

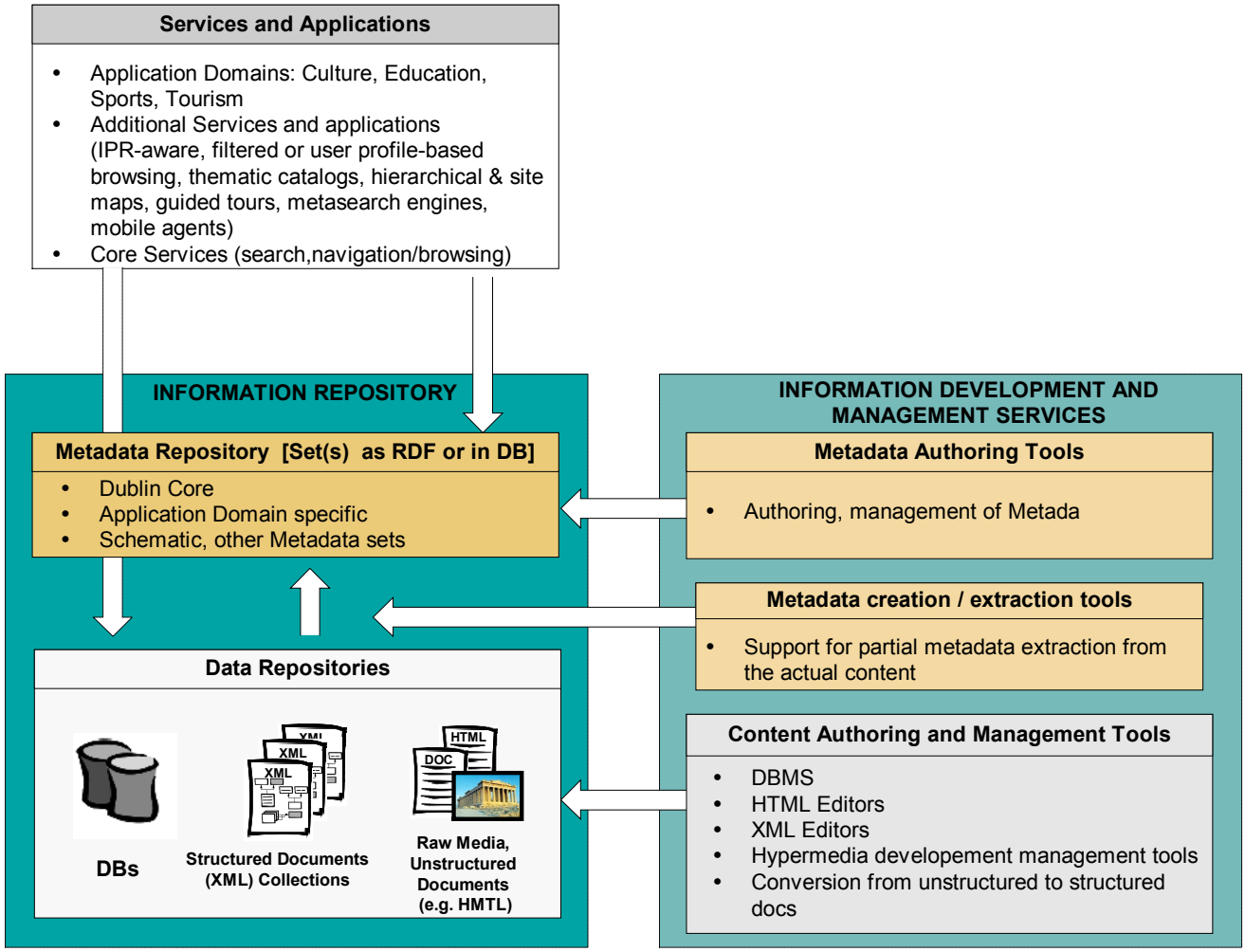

Fig 2. Target Site Architecture

The Services that should be provided in this layer include (in parentheses, we present sample tools that implement the services):

- Querying (e.g. for HTMLs, via the Harvest systems, for XMLs, via Visual XML Query, for DBs, via standard SQL interfaces)
- Data processing (e.g. Visual XML Tree, X-IT, IBM Visual DTD, Xerces – XML4J, Xalan, XIT, HomeSite, native DB tools)
- Other content conversions (e.g. Visual XML Creation, FOP, XSplit)
- Versioning (e.g. RCS)
- Access Control (e.g. WWW server security rights)

The *Metadata (MD) layer* acts as the "glue" or the common denominator (abstraction layer) among our different underlying repositories, by having all repositories "export" or publish the metadata related to their resources. Thus, remodeling of data repositories as a concept *includes* the generation of the appropriate metadata from each repository.

This use of metadata to enhance the discovery of electronic resources is widespread. Metadata can be navigational, schematic and associative (descriptive, restrictive, supportive, etc.) and are arranged within metadata sets. A well-known, domain-independent set, which is actually a standard, is the Dublin Core (DC) [5], with 15 elements for abbreviated descriptive cataloguing. DC is, however, not enough to offer an integrated and more complete view into our resources. We further need our own metadata sets, with the following functionalities:

- application domain – specific content description
- more consistent content description (using auxiliary sets such as glossaries, thesauri)
- exposing the schema/structure of underlying data repositories, in order to assist in the development of services and applications
- controlling security, by specifying security rights for content authors and end-users, along with Intellectual Property Rights information

- User profiling, assisting in regulating the navigation paths throughout the content and in customizing the information presented to the user

A widespread mechanism for specifying and aggregating metadata sets is the Resource Definition Framework (RDF) [18]. RDF also uses XML as the meta-language for metadata definition and DTD (or XML Schemata) for defining the structure of the metadata files. However, since RDF's support by most tools is not yet mature, it is suggested that metadata sets are maintained within a standard RDBMS, with proven facilities for query and retrieval of information.

The primary service of this layer is the authoring of metadata, which can either be manual or semi-automated, with fields retrieved from the actual content. The degree of success of the automated extraction depends on the semantics available within the actual content and on whether data elements can easily be mapped to metadata fields.

In the *application/services layer*, we have the core services of searching, navigation, browsing of *data* and *metadata* (which can actually be implemented by tools in the respective layers). Building up on them, more complex hypermedia and even software applications can be developed. Examples include tables of contents (TOCs), site maps, guided tours, adaptive/profile-based browsing, metasearch engines, navigational aids. These applications contain the navigational or, more broadly, business logic, as well as the User Interface.

With the underlying metadata repository, we can adapt the applications to operate in two stages: restrict our work to the metadata, until the user requests additional information, in which case the information is fetched from the respective data repository. This allows us to achieve faster user navigation and the ability to toggle among "broader" and "narrower" views of the content.

Thus, we address the need for building applications which can access, parse and re-use content from various repositories in more streamlined ways.

The *presentation*, although not examined as a separate layer like in [20], should always be differentiated from the navigational/business logic. This allows us to use various methods for the final rendering and interaction, from schemes such as XSL and/or CSS [3] to scripting and other programming tools (e.g. Javascript, Java, ActiveX, Flash).

4. Remodeling content and applications

We now focus on Stage [B] of the process model (Fig.1) and describe the steps to remodel our site's content and applications, while converging towards the architecture presented in the previous Section. More specifically, these tasks include:

1. Prototype implementation. A pilot test-bed should be created in development systems separate from the staging servers, enabling us to experiment with the new infostructure and adapted (hypermedia) applications. More specifically, this includes:

- Creation of test development environment, including configuring/integrating/building tools that implement services for data/metadata management (cf. Section 3) and for hypermedia applications management [20]
- Initial design and implementation of infostructure:
  a. Identify discrete content categories/ repositories, with material maintained as unstructured documents
  b. Select a limited number of document categories/repositories, to be remodeled within the context of the prototype
  c. Evaluate domain and application-specific ontologies, select data and metadata elements for analytical content description
  d. Design the new syntax/structure (DTDs or XML schemata) for the selected content categories
  e. Design syntax/structure (DTDs) for auxiliary/shared entities such as users, groups, organizations, keywords
  f. Design metadata sets and implement initial schema of the metadata DB repository
  g. Remodel selected content categories - convert existing HTMLs or other unstructured document types into XML files according to the appropriate DTDs. In parallel, populate metadata DB with the respective new records
- Basic Hypermedia applications design/ implementation:
  a. Design presentation schemes for the migrated content (XSL/CSS)
  b. Build renderer for generating appropriate versions of the structured documents for access devices (e.g. HTML 4.0 for latest Web browsers, WML for cellular phones)
  c. Remodel test application (e.g. Tables of Content) to operate on top of the integrated content/metadata (cf. next paragraphs for application remodeling). The application should have a preliminary working version of search and browsing facilities over metadata.

2. Verification of workflows, tools performance, content structure and proper function of applications. We verify

- that all initial information is preserved within the structured document

- that no hyperlinks to other external documents have been "lost"
- that the output of our remodeled test applications is consistent and contains all relevant information

  Methods of automated validation are desirable, but have not been explored yet.

3. Further implementation
   - Streamline workflows for content authors, graphics designers and programmers, fine-tune tools and supporting environment
   - Infostructure implementation:

     a. Expand the remodeling of other content categories to XML collections or to Databases, ie. design their new structure and perform conversion using existing tools or custom code. In parallel, populate the metadata DB with the respective new records

     b. Populate the metadata DB with content from pre-existing Databases repositories and structured document collections. This allows us to enrich our metadata repository with material from the entirety of our site, which might previously have not been explicitly cross-linked.

     c. Enhance and expand the metadata sets (i.e. schema of the metadata DB), following the ongoing restructuring of the content categories.

     In most cases, following the initial top-down site design, the maintainers fail or are unable to regularly recheck about relationships between material from different content sections. Using this approach, we have a bottom-up dynamic (re)building of a complete blueprint of the semantics of our entire site. The quality of this blueprint is better than the results typically generated by "smart" catalogue-building engines and search systems.
   - Hypermedia and Software applications implementation:

     a. Continuous design presentation schemes for the migrated content (XSL/CSS)

     b. Build generic data access library for interfacing with the metadata repository and other structured repositories. This library will be reused by the various applications to be remodeled and implements the core services of search and browsing of information elements (cf. next paragraphs for application remodeling)

     c. Remodel existing hypermedia and software applications (e.g. thematic catalogs) to operate on top of the integrated content/metadata, using the generic data access library
4. Testing and validation of workflow functionalities, of proper content structure, assess content reuse and modularity, applications flexibility and usability
5. Continuous maintenance
   - Recurring calibration of workflows, tools, addition of new tools for improved implementation of services (e.g. for XML management)
   - Continuous content (data/metadata) authoring in its new form (as structured/XML documents or within a Database).

     a. We should point out that restructuring of XML content categories (ie DTD or XML Schema, and all dependent XML files) can also occur, but without severe repercussions, since it is easier to migrate between structured information representations. If such a restructuring is performed by the content designers, then it is possible that also the metadata schema (and likewise the metadata records) should be updated
   - Evolution of more complex applications (e.g. search system, thematic catalogs, personalized guided tours and exhibitions, customizable table of contents, index of terms) on top of the integrated content, using shared libraries to the maximum extent.

Assuming that *applications* follow the typical 3-tier model (data access, business logic, user interface), their *remodeling* includes two stages:

1. In the data access part, implementing the necessary interface to the metadata DB repository is required. Further, we may use the schematic metadata sets stored in this repository to "lookup" available content classes (structured documents, Databases) and their fields. These operations, as indicated in the previous paragraphs, can be implemented as a re-usable data access library, shared to all remodeled applications.
2. For the business (or navigation) logic part, the programmer may optionally add support for the two-stage operation on metadata / data repositories presented in Section 3. This will enhance the performance of the application and provide links to other content categories, of which the user need not have previous knowledge of. Thus, even the casual navigation through the site can easily be enhanced with navigational aids (bird's eye and local views), referrals to "See related", pointers to dictionary of terms, without having the overhead of migrating all our content to one single repository.

5. The case study of ULYSSES

To validate our approach, we applied it in the evolution of the official site of the Hellenic Ministry of Culture (HMC), ULYSSES [23], which is maintained by our laboratory. ULYSSES is of sufficient complexity, with numerous different categories of material, underlying repositories, application types and languages. Content was originally either in static form (HTML documents, built at frequent time periods) or dynamic (database-driven). In an intermediate generation of content management, we developed a custom template-based system for documents management, Dynabuild. However, the rate of growth of the server could not be successfully met by this "virtual documents" solution. Furthermore, the degree of complexity in the content authoring process[2] clearly underlined the necessity of evolving to a new generation of the site, in order to minimize problems in dealing with heterogeneous content management and to allow for more flexible applications development.

Since the process model in use for "ULYSSES" was similar to the one presented in Section 2, we were able to directly start applying the analytic remodeling tasks, with the objective of implementing the architecture of Section 3.

In the current section, we discuss noteworthy technical issues and observations from this evolution. We will refrain from repeating all the steps of Section 4, but will focus on our technical comments. Most of them were derived during the implementation of the prototype.

5.1. Implementation of prototype

The *test development* environment primarily included tools from IBM's Alphaworks project for XML management and transformation, along with more typical programs for HTML and DB management

The *ontologies* examined were pertinent to the domain of Culture, and included ICOM [22]/CIDOC International Guidelines for Museum Object Information. Research on cultural metadata included the results of several cultural projects (e.g. Aquarelle[1])

The *metadata sets* we designed and implemented in a database repository included:

- Dublin Core
- Culture-specific (materials, artistic, glossaries of terms, keywords)
- Schematic (classes of XML content, elements of DTDs, database tables)
- Geographic-position related

Since metadata are stored within a relational database, it is easy to support features such as multiplicity of metadata fields and qualifiers (e.g. the LANG qualifier of Dublin Core). This also reduces data redundancy and improves integrity, as often-occurring text (e.g. keywords, locations) can be kept in separate sets and re-used by being linked to metadata fields, instead of being repeated in each metadata record.

For the *updating of metadata records*, we use custom code both when updating XML documents (in Perl) and database records (in Visual Basic, see below). Database forms are used for the manual input of fields which can not be automatically populated.

Three different content categories were selected to be *remodeled* into more structured forms.

For two of them ("Profile and Projects of the HMC", "Contemporary Cultural Heritage"), the appropriate DTDs were built and the categories' documents were converted into *collections of XML documents* using Percussion's XSplit. Apart from the actual content DTDs, auxiliary XML files were built to store:

- the table of categories currently converted to XML classes of documents
- shared entities, such as responsible personnel, cultural organizations, cultural terms

More importantly, we had to implemented (using Perl code and XSL) basic mechanisms of E-R databases for managing the various classes of XML content, in order to maintain content consistency and to sidestep the pending problems of efficient linking in XML (due to immature support for Xpointer [26]). These mechanisms include:

- Unique ID of each file
- 1-1 and 1-n relationships among XML files, by embedding the ID(s) of the target file(s) in the appropriate element of the source file (e.g. "organizational parent")
- n-n relationships among XML files, using "external" XML files containing the classes (ie DTD of content category) and IDs of source and target files. The element of the source XML file also references the element in the target XML file
- Referential integrity for the IDs, with cascaded update-delete

For the category of the "Cultural Heritage Map", with more than 2,000 documents on archaeological sites, museums and monuments, we implemented a new *relational database*. The existing content was migrated to the database through custom Visual Basic code, with support for regular expressions and patterns processing. Compared to the transformation of the two sample categories above into XML, the conversion to the DB

[2] With numerous applications and services (Interactive Cultural Map of Greece, full-text multilingual search system, thematic catalogs, databases of cultural events and announces, special exhibitions) and more than 150,000 discrete visitors monthly, ULYSSES is the primary dissemination site for all cultural heritage and contemporary cultural information authored by HMC's 170 regional organisational units and supervised cultural entities (e.g. museums), distributed in more than 50 cities throughout Greece.

required at least three times as much manpower. However, the requirements for complex interlinking among sites, museums, collections and artifacts made the use of a database unavoidable. We should, therefore, underline the ease of use that XML appears to offer, but also point out the fact that many functionalities have to be implemented from scratch and thus add to the initial, "set-up" effort.

Finally, we *adapted* a custom generator of our "tables of content" to operate on-top-of the metadata repository and the integrated three categories of content.

5.2. Other stages of the evolution

Verification demonstrated slight problems with the transformation of material, especially in the custom code migrating data into the DB of the Cultural Map. Fixes were applied and incremental updates were performed to the database and the respective metadata records.

During the *further implementation* period, additional categories of the site were remodeled into XML classes of documents and databases. Our metadata sets where enhanced and one more set, security-related was added, for defining access rights of specific categories of users.

The common data access library was developed and used in the remodeling of our thematic catalogue generator [8]. Originally, the purpose of the thematic catalogue was the presentation of cultural organizations, hierarchically organized according to different criteria (geography, organization kind, activities, etc.). The same engine, with the underlying metadata repository, was used to generate thematic catalogs of extra information, e.g. official departments of the ministry, archaeological research projects.

Testing revealed some usability problems when remodeled applications, which were originally optimized for a specific content category, were used to display information from other repositories. To solve this, we added fields to the metadata sets indicating length of information elements, data type, possible existence of null values.

In the current stage of *continuous maintenance*, categories are gradually being restructured and more complex software applications are being remodeled. A characteristic example is the Interactive Cultural Map, a Java application for geographical navigation of the cultural sights throughout Greece. Initially supporting only museums/monuments/sites, it is expanded to provide access to the entire spectrum of cultural information, by using the common data access engine.

A snapshot of the remodeled architecture of "ULYSSES" is portrayed in Fig. 3:

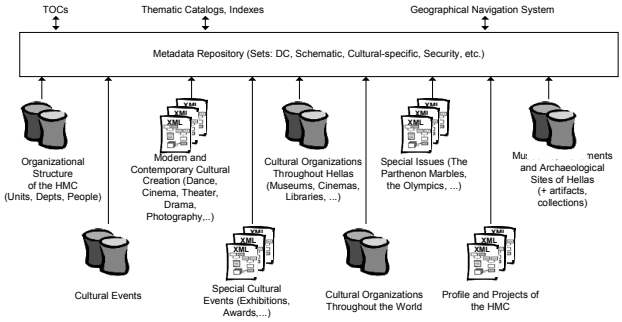

Fig 3. The new architecture of "ULYSSES"

5.3. Assessment

The qualitative improvement in the overall operations of the site is very positive:

- Content management is more streamlined. The rate of errors by the content authors is significantly reduced, since the content is structured and all authoring is done through forms-based interfaces
- The remodeling of applications is completed at a fraction of the overall effort required for their initial implementation. This holds both for simple hypermedia applications (e.g. thematic catalogues) and for more complex software applications (e.g. the Interactive Cultural Map)
- New enhancements to existing applications are implemented with the core data access library, whereas previously all logic for interfacing to the different repositories had to be individually coded.

The tight deadlines involved with evolving the whole site did not allow us, unfortunately, to collect analytic metrics from the various steps, in order to perform a quantitative evaluation of the success of our approach. However, we plan to apply our evolution methodology to two other large-scale sites, operated by our laboratory for the Ministries of Sports and Environment, where we will emphasize on

- automating specific tasks, especially validation and testing
- selecting appropriate metrics for monitoring our hypertext qualities (e.g. path complexities, size and volume of code, modularity, cohesion, tree impurity, flow and node complexity, fan-in/fan-out) and overall Web site properties (links, pages, frames, documents, images, embedded software items, Henry-Kafura/Shepperd metric)
- building tools for automatically gathering and analyzing measurements from the entire site, according to the previous metrics

The results of these analyses will also allow us to provide an objective comparison to other potential methodologies of evolving and remodeling Web Sites.

6. Conclusions

In this paper we have presented an integrated approach for evolving large-scale Web sites, by applying an open architecture and remodeling content and applications. First, we outlined a process model for the life-cycle of such sites, including the required design, implementation, maintenance and evolution steps. We then focused on two specific steps of the process model, which we analyzed in detail.

More specifically, we examined the layers and services of an open architecture for viable content management and for easier deployment of application/services on-top-of the integrated content. Furthermore, we proposed specific tasks for the actual remodeling of content repositories and applications into more flexible forms, according to the specifications of the target architecture. These tasks, including prototype, verification, further implementation (of remodeled content, hypermedia and software applications), testing, were analytically presented.

Finally, we demonstrated the practical value of our approach through its application on the evolution of a complex, large-scale Web site of the Hellenic Ministry of Culture. The approach can be used by other large sites which have legacy content repositories and wish to evolve to a more open architecture for content and applications.

We underline, however, that some limitations are identified in the degrees of freedom provided by the generic process model, as well as the core architecture, especially in the issues of applications development. In no case should our contribution be considered a complete hypermedia (application) design and management system for large-scale Web Sites. It is a focused approach, with practical applicability to many existing large-scale Web installations.

7. References

[1] Aquarelle: The Information Network on Cultural Heritage, http://aqua.inria.fr

[2] B. Boehm, “A Spiral Model of Software Development and Enhancement”, ACM Software Engineering Notes 11, 4 (1986), pp. 14-24

[3] Cascading Style Sheets, level 2, W3C Recommendation, http://www.w3.org/TR/REC-CSS2/

[4] D. Lowe, A. Bucknell, and R. Webby, “Improving Hypermedia Development: A Reference Model-Based Process Assessment Method”, ACM HyperText 99, pp. 139-146

[5] Dublin Core Element Set 1.0, http://purl.org/metadata/dublin_core_elements

[6] F. Garzotto, P. Paolini, and D. Schwabe, “HDM, a Model for the Design of Hypertext Applications”, ACM Hypertext 91, pp. 313-328

[7] F.G. Halasz, and M. Schwarz, “The Dexter Hypertext Reference Model”, ACM Commun, Vol 37, 2, 1994, pp. 30-39

[8] G. Styliaras, P. Zafiris, and T.S. Papatheodorou, "Implementing highly configurable Subject Trees: The ITC system", WebNet 98, November 1998

[9] H. Gellersen, R. Wicke, and M. Gaedke, “WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle”, WWW6

[10] H. Watts, and M. Kellner, “Software Process Modeling: Principles of Entity Process Models”, Proceedings of the 11th int. conference on Software engineering, 1989, pp. 331 - 342

[11] M. Gaedke, J. Rehse, and G. Graef, “A Repository to facilitate Reuse in Component-Based Web Engineering”, Web Engineering workshop / WWW8

[12] P. Atzeni, G. Mecca, P. Merialdo, A. Masci, and G. Sindoni, “The Araneus Web-Base Management System”, ACM SIGMOD, 1998, pp. 544 - 546

[13] P. Fraternali, and P. Paolini, “A Conceptual Model and a Tool Environment for Developing More Scalable and Dynamic Web Applications”, Proc. Int. Conf. on Extending Database Technology, Valencia, EDBT98, pp. 421-435

[14] P. Fraternali, “Tools and approaches for developing data-intensive Web applications: a survey”, ACM Comput. Surv. 31, 3 (Sep. 1999), pp. 227 - 263

[15] P. Warren, C. Boldyreff, and M. Munro, “The Evolution of Websites”, Proceedings of the Seventh International Workshop on Program Comprehension, 1999, Pittsburgh, pp. 178-185

[16] P.A. Zafiris, N.P. Georgantis, G.E. Kalamaras, S.P. Christodoulou, and T.S. Papatheodorou, “A Framework for Content Modeling and Management in Large-Scale WWW Sites”, SSGRR 2000, L’ Aquila, Italy [to be published]

[17] Percussion, the Rhytmyx Content Management System & the XspLit conversion tool, http://www.percussion.com

[18] RDF Model and Syntax, W3C Recommendation, http://www.w3.org/TR/REC-rdf-syntax

[19] S. Ceri, P. Fraternali, and A.Bongio, “Web Modeling Language (WebML): a modeling language for designing Web sites”, WWW9 Conference, Amsterdam, May 2000

[20] S.P. Christodoulou, P.A. Zafiris, and T.S. Papatheodorou, “WWW 2000: The Developers’ View and a Practitioner’s Approach to Web Engineering”, ICSE 2000, 2nd Web Engineering Workshop, Ireland, June 2000

[21] T. Isakowitz, E.A. Sthor, and P. Balasubranian, “RMM: a methodology for structured hypermedia design”, ACM Communications, Vol 38, 8, 1995, pp. 34 - 44

[22] The International Council of Museums, www.icom.org

[23] ULYSSES, official WWW site of the Hellenic Ministry of Culture, http://www.culture.gr

[24] V. Basili, and A. Turner, “Iterative Enhancement: A Practical Technique for Software Development”, IEEE Transactions on Software Engineering SE-1, 4 (1975)

[25] W. Royce, “Managing the Development of Large Software Systems”, Proceedings of the 9th International Conf. on Software Engineering, IEEE, 1987, pp. 328-338

[26] XML Pointer Language (XPointer) Version 1.0, W3C Candidate Recommendation, http://www.w3.org/TR/xptr

[27] XML Schema, W3C Working Draft, http://www.w3.org/TR/xmlschema-[0,1,2]

[28] XML, Extensible Markup Language (XML), http://www.w3.org/XML/

[29] XSL Transformations (XSLT) 1.0, W3C Recommendation, http://www.w3.org/TR/xslt