Level of Detail Concepts in Data-Intensive Web Applications

Sara Comai

Politecnico di Milano, Dipartimento di Elettronica e Informazione
P.zza L. da Vinci, 32, 20133 Milano, Italy
sara.comai@polimi.it

Abstract. Current *data-intensive* Web applications, such as on-line trading, e-commerce, corporate portals and so on, are becoming more and more complex, both in terms of density of information and in terms of navigational paths. At this aim different techniques have been proposed in literature for optimizing the information to be shown to the user. In this paper we present a technique for tuning the amount of data presented to the user, directly inspired to the concept of *Levels of Details (LoD)*, commonly used in computer graphics. Like in computer graphics the idea is to simplify the original model, without loosing the main characteristics of the objects to be shown. The approach is based on the application of LoD operators to the compositional and navigational structure of a Web application, expressed through an hypertextual model.

1 Introduction

Today *data-intensive* Web applications, allowing access and maintenance of large amounts of structured data, typically stored in a database management system, represent a broad class of applications [4]. They include, for example, on-line trading and e-commerce applications, institutional Web sites of private and public organizations, corporate portals and so on. Thanks to the availability of CASE tools supporting the design and, possibly, their automatic generation, such applications are becoming more and more complex. The complexity entails both the density of the information presented to the user (like in portals, sites presenting news or products) and long navigational paths to reach the desired information content.

To deal with this increasing trend, the need for new techniques for optimizing the information to be shown to the user has been recognized. In literature, different approaches have been proposed, such as the adaptive hypermedia systems [1] [3], aiming at the adaptation of the content and of the navigational paths of the application according to the user interests. In this paper we present a different technique inspired to the concept of Levels of Details, which can be seen as complementary to adaptivity.

The concept of *Levels of Details (LoD)* was first introduced in computer graphics by Clark [2], for defining simpler versions of the geometry for objects that have lesser visual importance, such as those who would be far away from the viewer. Complex geometries can be simplified by removing graphical primitives in order to produce simpler models, which retain the important visual characteristics of the original object. This technique, applied to complex objects or to complex portions of the scene to be rendered, allows to obtain a whole series of simplifications with different *amount of detail*, and to choose the simplified version according to the size, distance or importance of the object components.

We believe that this kind of technique can be successfully transposed also in the Web design context, especially to data-intensive Web applications.

The idea is to exploit a dividi-et-impera technique, where different *hierarchical abstraction levels are applied to the representation of an hypertext.* While Web designers apply these techniques to master the complexity of the design of a Web application, here the same technique is offered to the final users, to master the complexity, in terms of *amount of data*, of the information provided by the site, so that the site be more readable, the loading of the pages becomes faster, the needed information can be easily found, and so on.

This kind of approach presents several benefits:

– The performances, in terms of amount of data to be transmitted to the client, can be improved, when low levels of details are required.
– The user can scale the information of the visited site according to her needs: typically, first users need more information (and appealing sites with lots of graphics); frequent users claim a fast access to the data.
– The usability of the site can be better supported, as stated also in [6].

Essential aspects of the LoD concept have been employed in the techniques for adaptivity [1] [3], where the content and the links of the pages are adapted according to the interests of the user. However, the proposed approach presents some differences compared to adaptivity. First of all, adaptivity is typically applied server-side: the application takes the decision of the content/link to present to the user, according to her past behavior or to her explicit declaration of interests. Therefore, a user model is built at this aim. Instead, the approach proposed in this paper does not rely on a user model (which could however be integrated), but provides a mean to the final user for tuning the amount of data rendered by the application. It can be seen as a client-side approach, which can be used also by non-registered users or by users refusing cookies. In any case, it does not exclude the application of adaptive techniques, which could be combined with it.

An approach similar to the one presented in this paper has been studied in [6], where a stratification is defined on the application data and access to the data is provided according to such stratification. However, as we will see, such an approach presents some limits when applied to complex Web applications. With respect to this work, in this paper the application of the Level of Details concept is defined on a conceptual hypertext model, suited for complex applications.

The paper is structured as follows: Section 2 presents an overview of the issues related to the LoD approach, which are then formalized in Section 3. Section 4 compares our approach with related work, and finally Section 5 draws the conclusions and presents future work.

2 The LoD Approach Applied to Web Applications Concepts

In computer graphics the LoD technique allows the representation of graphical objects with a level of detail bound to the *distance/importance* of the object from the observer. Distant objects can be simplified; when they are in front of the viewer their

LoD is increased. In the same way it is possible to apply LoD techniques to Web pages, with a level of detail bound to the *amount of information* that the user would like to receive. Unlike in computer graphics, in case of Web applications, the user can tune the "distance/importance", which is the metric used to choose the desired LoD.

The amount of information can be simplified using alternative LoD operators, such as *filtering*, for cropping from the Web page the less important data, *summarization,* for substituting less important data with meta-data describing it; or, in a complementary way, important information can be highlighted with *zooming* operators (analogous operators have been define in adaptive hypermedia systems [1] [3]). In the sequel, we will refer to the filtering operator, but the results easily extend also to the other operators.

The LoD concept can be applied orthogonally to the three traditional design levels composing a Web application: the *data* of the application, the *hypertext* that represent the navigational interfaces used to publish the application data, and the *presentational aspects* of the hypertext. For example, among the data of a particular object (e.g., the data concerning a book) some pieces of information are considered fundamental, while others could be filtered when the user navigates with a low LoD. In the same way, inside a given page some objects (which can include also advertisements[1]) are more important than others and different LoDs could be defined. At the presentational level, LoD operators can be applied to the rendering of the graphical objects.

In order to consider all the main aspects of complex Web applications, we have conducted our analysis on a Web specification language, namely WebML [4]; however, the presented examples and results can be mapped also to other notations. A brief summary of the WebML concepts is presented to understand our solution. The LoD approach will be illustrated on a relatively simple Web application (the site about WebML), but containing a quite rich compositional and navigational structure.

2.1 A Brief Overview of the Web Concepts Through WebML

WebML consists of a *data model,* describing application data, of a *hypertext model*, expressing the Web interface used to publish this data, and of a *presentation model.*

The data model. The WebML data model is the standard Entity-Relationship (E-R) model. As an example, Fig. 1 shows the E-R schema describing part of the database of the webml site [11]. Every entity has a unique identifier attribute, named ID, which is not depicted to avoid clutter. It contains the news, the papers about WebML organized into different categories, and the persons working on the project and authors of the papers. Moreover, it stores the data of a book, organized in parts, chapters and sample pages; exercises and additional materials are associated to chapters.

The hypertext model. Upon the same data model it is possible to define different hypertexts (e.g., for different types of users or for different publishing devices), called site views. A *site view* is a graph of *pages*, allowing users from the corresponding group to perform their specific activities. Pages consist of connected *units*, representing at a conceptual level atomic pieces of homogeneous information. They publish

[1] The LoD technique presented in this paper to application data, can be applied also to commercial objects to be included in the site

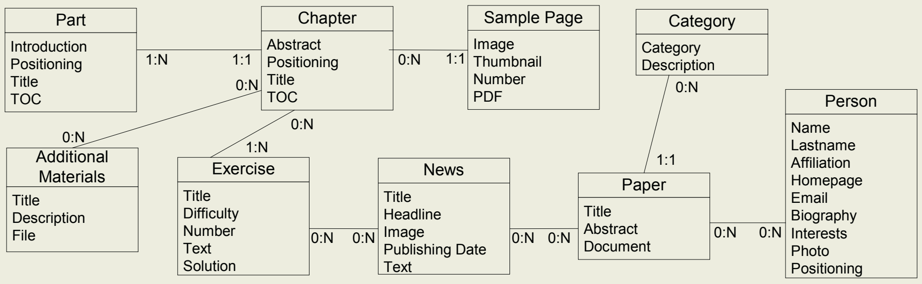

Fig. 1. Sample data model for the official WebML Web site

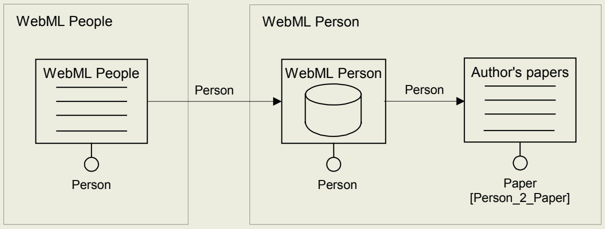

Fig. 2. Sample hypertext model for webml.org

content retrieved from the entities of the underlying data model. In particular, they show a subset of the attributes of a given entity. Units within a Web site are often related to each other through *links* carrying data from a unit to another, to allow the computation of the hypertext. To determine the data that are displayed by a unit, a *selector* is specified, which tests complex logical conditions over the unit's entity.

The hypertext fragment in Fig. 2 shows two pages: the WebML People page presents a list of persons (through an *index unit* called "WebML people" and defined over the Person entity); when the user selects a person from the index, the link exiting the unit is followed, carrying the identifier of the selected person. The WebML person page shows the details of the selected person (through a *data unit* called "WebML Person" defined over the Person entity). Notice that only the Person corresponding to the identifier received from the incoming link is shown. The person page lists also the papers written by the person (through the index unit called "Papers of Selected Author", defined over the Paper entity, and extended with a selector for retrieving only the papers of the selected person).

The language includes several other units and modularization constructs for organizing complex applications into hierarchies. For further details about the language the reader may refer to [4].

The **presentation model** addresses the definition of the presentation rules specifying the page layout, the position of the units and of the links in the page, and the position of the attributes specified inside a unit.

In the next subsections we present some simple examples specified in WebML to provide an overview of the possible issues concerned with LoDs. Then, in the section 3 we formalize our approach by describing how LoDs are built.

2.2 Different Hypertexts on the Same Data

The core model in the design of a Web application is represented by the *hypertext model, which is based on the data model and drives the presentation design.* We therefore face our analysis focusing on this model, which includes also the data to be shown. Indeed, as we will see, different hypertext fragments defined on the same data may require different LoD specifications with respect to such data.

Consider, for example, the hypertext fragment, shown in Fig. 3, representing the person and the paper pages in the webml Web site. Both are defined on the person and paper entities and their relationship. In the former page, the most important data appearing at the top of the page relate to the person, in the latter, the most important data concern the paper. They can be defined as the lowest level of detail to be shown in the page (LoD_1). In both cases, to the less important information (in LoD_2) a simplifying LoD operator could be applied. For example, if the parts in LoD_2 are summarized, they are replaced with meta-data indicating to the user which kind of data are not shown at the lowest level LoD_1. Notice, that even if the portion of data model on which the two hypertexts are based is the same, different stratifications of the data are implied from these two pages, as shown in the bottom part of Fig. 3. The LoD-based approach must be therefore applied *top-down*, starting from the hypertext and examining at a second level the underlying data, so that all the possible combinations of data can be considered.

2.3 Different Access Paths to the Same Hypertext Fragment

The application of the LoD technique to an hypertext must also consider all the possible access paths to a given page. Indeed, in complex Web applications the same page can be reached through different navigational paths, to facilitate access to users.

For example, consider the hypertext in Fig. 4: it shows an exercise (Exercise data unit), extracted from a particular chapter (whose details are retrieved by the Chapter data unit) of the book. Also the list of the exercises of the same chapter is shown (Exercises of the chapter index unit).

This page can be accessed in two ways: either with the link towards the Exercise data unit (access A) or with the link towards the Chapter data unit (access B). In the former case, the user, after selecting *a particular exercise* in another page, is led to this page to see the details about that exercise. The exercise data are therefore important (in LOD_1), while the list of all the other exercises of the chapter, to which the selected exercise belongs could be displayed at a second level (in LOD_2). In the latter case, the user, after selecting a *particular chapter* in another page, is led to this page to see one (or more) exercise(s) related to that chapter. In this case, the list of the exercises available for the selected chapter must be shown to user, so that she can select the desired exercise. If a default exercise is shown to the user for the selected chapter, also the Exercise unit plays an important role in this page: in such case all the units belong to the same initial LoD.

Different LoDs could be therefore defined according to the access path followed by the user. This simple example, shows that the same hypertext, used with different access paths, may imply different stratifications of the hypertext.

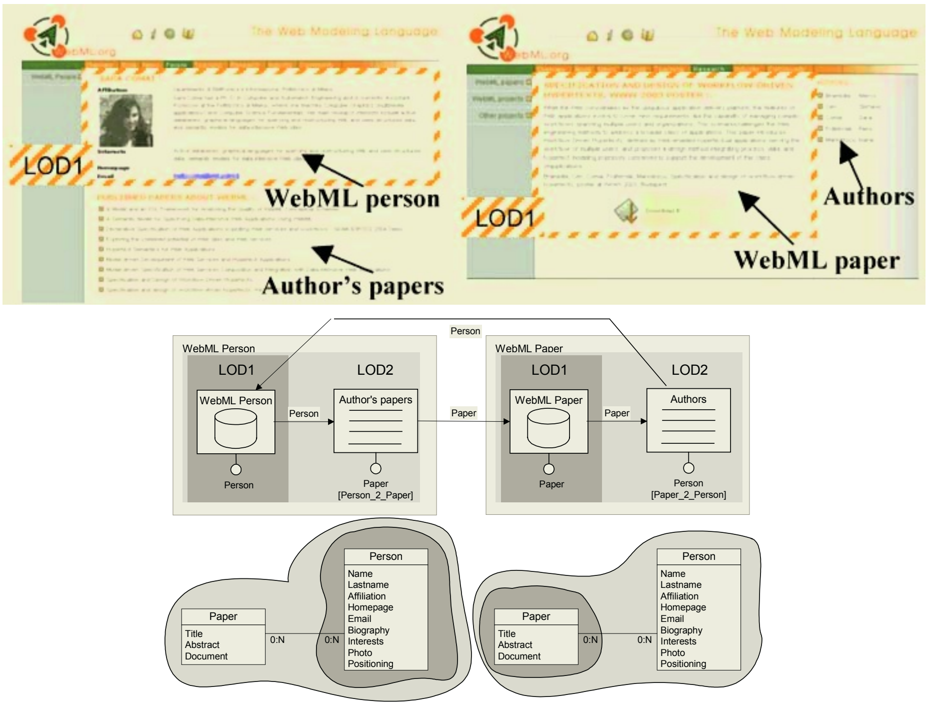

Fig. 3. Rendering of the WebML person page and of the WebML paper page; their hypertextual representation and the inferred stratification on the data model

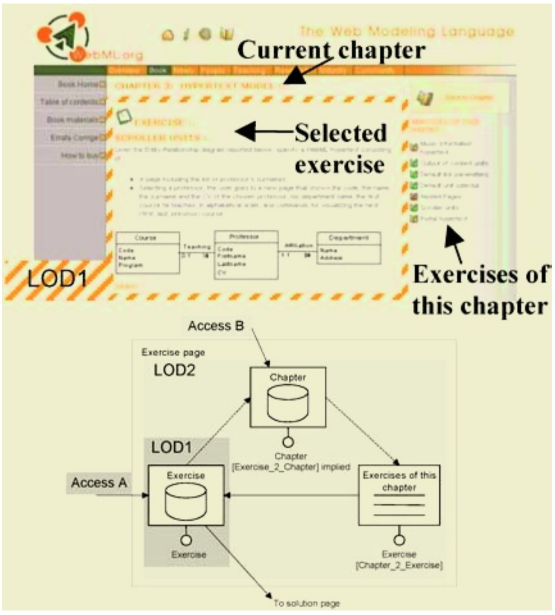

Fig. 4. The hypertext model of the WebML exercise page and its rendering

3 LoD Definition

In order to apply the LoD technique it is necessary to assign priorities to the elements of a page, and render them according to such priorities. However, some rules are needed to maintain the consistency with the complete Web application and to preserve the meaning conveyed by the designer for its pages. An important *hypothesis* at the basis of our technique is that any LoD operation must preserve the topology of the Web application and the general layout of the page, in order to preserve also its usability.

In an hypertext, LoDs can be applied at three main concepts: the content of a page, the content of a single information unit, and the links. We will focus on the former, since it represents the core element of a Web application.

3.1 Content of a Page and Rules for Defining the LoDs

A page shows the content of different units, typically having a certain degree of cohesion. In WebML the specification of the content of a page can be seen as a *graph of units, connected by links*.

In particular, in WebML we can identify:

– *Context-free units*, which do not depend on other units (they have no incoming links).
– *Externally dependent units*, requiring mandatory input (through a link) from a unit in a different page.
– *Internally dependent units*, requiring input only from units inside the same page.

For example, consider the WebML Chapter page (see Fig. 5): it describes the content of a chapter (Chapter data unit), and provides some sample pages (Sample pages index), a link to the part containing such chapter (Part data unit), and indexes of related exercises and additional materials. Other chapters can be browsed through scrolling commands (Chapter browsing scroller unit).

According to the above classification, the Chapter browsing scroller unit is a context free unit since it has no incoming links, the Chapter data unit is an externally dependent unit since it has an external incoming link, and the other units are internally dependent unit, since their incoming links depart from internal units.

Taking into account the composition of a page, we can define a set of rules for assigning correct LoDs to the units.

Let us call *K-unit*, the key unit with the highest priority to be always shown in the page[2]. In the Chapter page in Fig. 5 the key unit is the Chapter data unit, showing the details about the chapter. Notice that the K-unit may be any kind of unit (a data, an index, and so on), and may be positioned in any node in the graph.

A unit is (statically) *computable* if all the units providing input to it can be computed. For example, the index unit Exercises needs in input the current chapter to compute the list of exercises of the chapter: it can be computed once the Chapter unit has been computed. Notice that context-free units are always computable. Externally

[2] In this paper we limit our analysis to exactly one key unit per page, which is consistent with most of the current data-intensive Web sites

dependent units are computable when their link coming from the external page is navigated and, if other inputs are needed from internal units, also such the internal units can be computed.

This property can be checked statically on the graph of units, for each possible access modality to the page. The assignment of the LoDs to the page must take into account the computability of the units, to guarantee that the units belonging to a given LoD be actually computable.

A *K_1-cluster* is composed by a set of units (and their connecting links) containing in particular a sequence u_1, ..,u_n where u_1 is a) an externally dependent unit, if the page is accessed through it, or b) a context-free unit otherwise, and u_n is a K-unit (u_1 may coincide with u_n).

A *K_1-cluster is computable* if the K-unit contained in it is computable.

For each access modality to the page a computable K_1-cluster has to be defined.

Starting from a computable K_i-cluster it is possible to extend it to a computable K_{i+1} cluster (for each 1<=i <max level), by adding a set of units that are computable starting from the units in the K_i-cluster.

To each K_i-cluster a LoD is assigned, which can be defined on the hypertext by assigning *priorities* to the units. The units having a priority less or equal to a given value i belong to the same LoD. Notice that there are as many priorities for each unit as the number of different distinct access modalities.

The LoDs generated from the above definitions allow to easily implement a mechanism to provide *feedback to the user* about the amount of filtered information, calculated as the difference of the LoD at the maximum level and the LoD of the current level.

Consider, for example, the Chapter page in Fig. 5, whose K-unit is represented by the Chapter data unit: two access modalities are possible, either through a link pointing to the whole page, or through the link towards the Chapter data unit. In the former case the K_1-cluster must contain at least the Chapter browsing unit (which is the only context-free unit contained in this page) and the Chapter data unit (which is the K-unit). In the latter, the Chapter data unit represents both the externally dependent unit and the K-unit, so it can be the only unit in K_1-cluster. According to the definition the K_1-cluster may contain also other units, if they are statically computable. For example, in both access modalities the set {Chapter browsing scroller unit, Chapter data unit, Part data unit} is a valid K_1-cluster, since it is possible to statically compute all such units. The next clusters may then extend the K_1-cluster with the other units, to which different priorities may be applied. For example, K_2-cluster= K_1-cluster U {Sample Pages index} and K_3-cluster= K_2-cluster U {Additional Materials index, Exercises index} is a valid set of clusters (see Fig. 6). Notice that, as shown in this example, it is always possible to define a K_1-cluster including all the access modalities: in this case, a unique priority for each unit can be defined.

As another example, consider the Exercise page in Fig. 4: through access A the described LoDs are determined by K_1-cluster={Exercise data unit} and K_2-cluster= K_1-cluster U {Chapter data unit, Exercises of this chapter index}; for access B the K_1-cluster contains all the units. Notice that for access A the K_1-cluster could also include all the units, or valid LoDs could be produced also with K_1-cluster={Exercise data unit}, K_2-cluster= K_1-cluster U {Chapter data unit}, and K_3-cluster= K_2-cluster U {Exercises of this chapter index}.

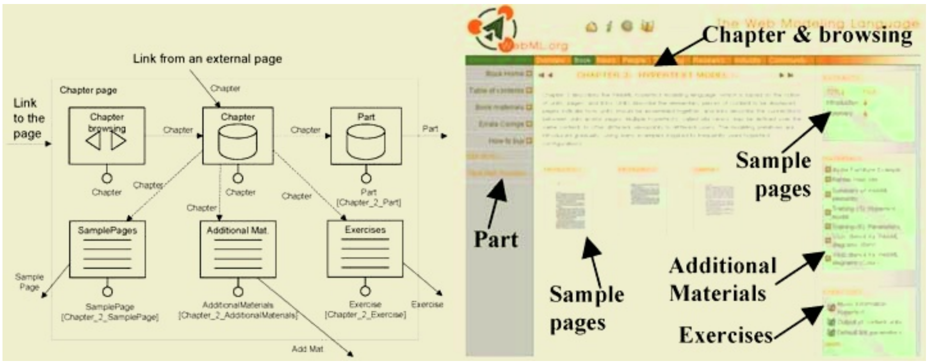

Fig. 5. The WebML Book chapter page hypertext model and its rendering.

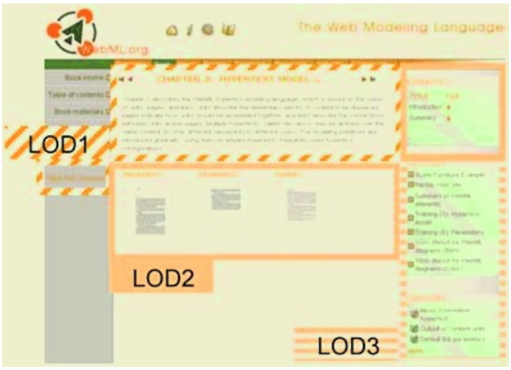

Fig. 6. Example of a possible LoD definition for the WebML Book chapter page

3.2 Links and Content of a Unit

The LoD operators may also be applied to links activating operations or leading to other pages[3], or to the content of each unit: e.g., among a set of links exiting the page it is possible to filter the less important, or among a set of attributes displayed by a unit, some priorities could be defined. In this latter case, the approach is similar to the data stratification proposed in [6], and does not require particular rules in the priority assignments. However, also at this level the same kind unit in different pages or two different kinds of units defined on the same entity may require different criteria for selecting the attributes to which the LoD operators are applied. Therefore, for each unit/page priorities can be assigned to the attributes and links.

3.3 Implementation

The proof of concepts of the proposed approach has been implemented using tclhttpd [9], a Web server built entirely in Tcl [8]. Tclhttpd provides a Tcl+HTML template

[3] Notice that intra-page links connecting two units are treated by the rules defined in Section 3.1, since a cluster contains a set of units and the links connecting them

facility that allows to mix Tcl commands with HTML. We have built a Tcl package linkable from tclhttpd implementing the WebML units. The Tcl+HTML template we have defined is composed of two parts: the first part computes the content of the units contained in the page; unit computation is performed through a call to a Tcl procedure implementing the unit. WebML units are instantiated with a designer-defined priority created according to the rules presented in Section 3.1. In the second part, the content of the unit is inserted into the markup describing the page. For storing the application data we have used Sqlite [7], an embeddable relational database library, supporting SQL92 and providing a Tcl interface. The WebML units interact with the database through such interface.

The rules for specifying the LoDs and their computation have been proven on the most complex pages of several real Web applications specified with WebML such as our department Web site (http://www.elet.polimi.it), a complex application including a set of B2B portals (http://www.metalc.it), and other business sites (http://www.acer-euro.it, http://www.webratio.com).

Currently we are mapping these rules on the WebML specification supported by the WebRatio tool [12], a development environment for the visual specification of applications in WebML and the automatic generation of code for the J2EE and Microsoft .NET platforms, in order to test it on the complete applications.

4 Related Work

Adaptive hypermedia systems (surveyed and described in [1], [3]) and adaptive Web approaches [5] also consider the removal of objects from pages, e.g. by introducing conditional links, conditional fragments, and hidden links. Therefore also adaptivity applies to navigational structures, and page content, and also in the adaptive context operators like filtering/summarization/zooming are used. However, adaptivity is based on the user's interests and *knowledge*, given by the information already visited by the user. When we filter some information our aim is not that of removing useless information for the user (all the data of the page are always available and can be seen by setting the maximum LoD), but to assign different priorities to the different pieces of information, so that the user can select to explore the site with the desired LoD and increase or decrease it according to her interests. Our approach is *orthogonal* to adaptive systems: it can be applied also to the results of an adapted page, to assign different priorities to the information that could be interesting for the user. On the other hand, the information to which the LoD operators apply could be determined with an adaptive approach: this would allow to apply sophisticated heuristics in the assignment of the priorities to the units, attributes and links shown in a page. Another important difference is that we do not assume a priori knowlegde about the interests of the user and therefore our approach can be applied to any kind of Web application. Moreover, with our approach it is the user (at client-side) that selects the amount of information to be displayed according to her needs, and the decision about the amount of detail is not taken by the system.

To our knowledge, the only work providing a stratification of the information of a Web Information System with the aim of assigning different priorities is that presented in [6]. In this work the authors apply a stratification to the information space, described by a model called Progressive Access Model (PAM), possibly connected to functional, hypermedia and user models. The proposed approach is *bottom-up*, since the stratification is defined on the *data model* and then the other models use such stratification. We also face the problem of managing different LoDs to provide a progressive access to the information, but our approach is *top-down*: the stratification is applied to the *hypertext* and then, for each object contained in the hypertext, a stratification can be defined on the displayed attributes or links. This allows us to consider *complex Web applications*, typically providing different point of views of the same underlying data and different access facilities to the same data.

5 Conclusions and Future Work

In this paper we have presented an approach for applying different LoDs to the concepts of a Web application, focussing in particular on the content of a page. We have shown through some examples, that complex Web applications built on several interrelated data and providing different access paths to such data, require the assignment of different LoDs, and a top-down approach starting from the hypertextual layer. We have provided a set of basic rules that must be satisfied to produce significant priority rules for guaranteeing the computation of the page.

Such an approach can be extended to be effectively applied also in adaptive Web applications, by extending users profiles with the information about the priorities associated with the Web concepts.

Moreover, also accessibility [10] could take advantage from this approach, since different LoDs can be assigned to the information of a Web application taking into account their importance for the different classes of users. For example, someone who cannot use the mouse needs shortcuts, which can be filtered to all the other classes of users; images can be filtered to blind users; and so on.

As future work we plan to automatize the phase for assigning the priorities to the content of the Web pages, by defining heuristics for the identification of the key units and for inferencing the priorities of the other units contained in a page. Moreover, we plan to study more sophisticated clustering algorithms involving multiple stratifications, and, finally, we would like to investigate the possibility to remove the assumption of preserving the site topology, by applying merging/splitting operators at the page level.

Acknowledgements

I wish to thank Alberto Allara for helpful discussions and contributions to the implementation of the proposed approach.

References

1. Peter Brusilovsky Methods and Techniques of Adaptive Hypermedia. *User Modeling and User Adapted Interaction*, 1996, V. 6, N. 2-3, 87-129
2. Clark, J. H. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 1976, 19:547-554
3. Paul De Bra, Peter Brusilovsky, Geert-Jan Houben. Adaptive Hypermedia: From Systems to Framework.. *ACM Computing Surveys*, Vol. 31(4), 12, 1999, ACM Press

4. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, M. Matera: *Designing Data-Intensive Web Applications*, Morgan-Kaufmann, December 2002.
5. Mike Perkowitz, Oren Etzioni. Towards adaptive Web sites: Conceptual framework and Case Study. *Artificial Intelligence*, 118 (2000), 245-275
6. M. Villanova-Oliver, J. Gensel, H. Martin. A Progressive Access Approach fro Web-Based Information Systems. *Journal of Web Engineering*, Vol. 2, (1&2) (2003) 027-057
7. Sqlite. http://www.sqlite.org/
8. Tcl. http://www.tcl.tk/
9. Tclhttpd - Tcl Web server. http://www.tcl.tk/software/tclhttpd/
10. W3C. Web Content Accessibility Guidelines 2.0, Working Draft 30 July 2004. http://www.w3.org/TR/2004/WD-WCAG20-20040730/
11. WebML Site. http://www.webml.org
12. WebRatio. http://www.webratio.com