Flavius Frasincar
Geert-Jan Houben
Richard Vdovjak (Eds.)

WISM 2005
Web Information Systems Modeling

Workshop at

ICWE 2005

The 5th International Conference on
Web Engineering
Sydney, Australia, 25-29 July, 2005

WISM 2005 Workshop Organization

Program committee

Tom Barrett	USA
Mario Cannataro	Italy
Olga De Troyer	Belgium
Martin Gaedke	Germany
Jaime Gomez	Spain
Maristella Matera	Italy
Oscar Pastor	Spain
Klaus-Dieter Schewe	New Zealand
Bernhard Thalheim	Germany

Workshop organizers

Flavius Frasincar	The Netherlands
Geert-Jan Houben	The Netherlands
Richard Vdovjak	The Netherlands
Peter Barna	The Netherlands

Preface

Modern Web Information Systems (WIS) need to satisfy a large number of requirements coming from different WIS stakeholders. Modeling WIS by focusing at one design aspect at-a-time helps the implementation of these requirements. During the last years several model-driven methodologies have been proposed to support the WIS design.

Strategic modeling is usually the first step in WIS design. It is a very general characterization of WIS which answers questions like: what is the purpose of the WIS?, which are the WIS users?, what functionality is provided by the WIS?, what is the content of the WIS?, what is the layout and atmosphere of the presentations provided by the WIS?, etc. It is only after answering the above questions at a high abstract level that the designer can proceed with the detailed specifications of the WIS.

Data integration is one of the most important characteristics of WIS. Some examples of domains in which data integration is present are: public services and bioinformatics. WIS need to support user interfaces that make a lot of data coming from different sources available to the user in a transparent way. The Semantic Web technologies seem to facilitate the data integration problem on the Web by providing the necessary languages to describe the data semantics.

Very often the Web user browses pages that he will like to view again at a later time. The present browsing history mechanisms included in Web browsers proved to be insufficient for an adequate retrieval of already seen information. A semantical organization of the previously visited pages can improve the process of retrieving previously seen data.

There is an increasing demand to make WIS personalizable so that these systems better deal with the user interests. WIS design methodologies do propose adaptation techniques in order to realize WIS personalization. Despite the fact that some of these adaptation techniques are very similar (or even the same) in different methodologies, the notations to specify WIS personalization aspects are quite different. By defining a reference model for specifying WIS personalization one could improve the reuse of the personalization specifications and also enable a seamless translation between different specific personalization specifications.

The above issues are some of the topics that are tackled in the workshop papers. We hope that we did raise the readers' interest so that they will have a close look at the papers and possibly contribute to the fascinating and challenging area of WIS modeling.

Flavius Frasincar
Geert-Jan Houben
Richard Vdovjak
(July, 2005)

Table of Contents

Invited paper

Regular papers

Strategic Modelling of Web Information Systems and its Impact on Visual Design Patterns*

Thomas Moritz
BTU Cottbus
Computer Science Institute
Postbox 101344
03013 Cottbus, Germany
moritz.th@t-online.de

Klaus-Dieter Schewe
Massey University
Department of Information Systems &
Information Science Research Centre
Private Bag 11 222
Palmerston North, New Zealand
k.d.schewe@massey.ac.nz

Bernhard Thalheim
Christian Albrechts University Kiel
Department of Computer Science
and Applied Mathematics
Olshausenstr. 40
24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Abstract

The development of web information systems (WISs) requires modelling on various layers of abstraction. Based on an abstract abstraction layer model (ALM) the work in this paper approaches the modelling on the highest layer dealing with strategic modelling. Strategic modelling addresses a very general characterisation of WISs in terms of its content, functionality, context, usage and presentation. The paper discusses branding, utilisation space modelling, utilisation portfolio modelling and atmosphere modelling as the major parts of a strategic model. In a second step it is then shown how the strategic model impacts on the formation of a WIS in terms of its layout and playout.

1. Introduction

The development of web information systems (WISs) requires modelling on various layers of abstraction. The co-design approach to WIS modelling in [12] emphasis a strategic layer, a business layer, a conceptual layer, a presentation layer, and an implementation layer. Other methods such as HERA [8], WSDM [5], OO-H [7], OOHDM [14], HDM [6], ARANEUS [1], WebML [2] and UML [9] agree on the use of abstraction layers as such, but differ in the concrete use of layers. In particular, higher layers dealing with strategic modelling and business-oriented modelling are often neglected.

The first goal of this paper is to discuss modelling on the strategic layer of a WIS (see Section 2). Strategic modelling addresses a very general characterisation of the system in terms of its content, functionality, context, usage and presentation. The strategic characterisation of a WIS starts from the very general question what the WIS is about, i.e. the purpose(s) of the system, and what are criteria for the WIS being successful. The general answer to these questions gives rise to an informal *mission statement*, and a characterisation of the *brand* of the WIS. The latter one will follow the general classification scheme for WISs.

Going more into details we first explore the kind of content that is to be presented in the WIS, and the kind of functionality, with which this content can be accessed, customised to the needs of particular WIS users, and updated. This defines the *utilisation space* of the WIS. We then explore the *utilisation portfolio* to gain even more details. This means to model the users (or actors) who will use WIS, their goals, i.e. why they are supposed to use the system, and the tasks that have to be performed to reach these goals.

The fourth and last part of a strategic WIS model are general principles for the formation of the WIS presentation. These address the layout, atmosphere and progression of the system based on knowledge about the cognitive perception of form, colour and other style elements. Taking these parts together we should keep in mind that the role of strategic modelling is to lay out the plan for the whole WIS without drifting into technical details. As a consequence, the techniques applied on this level will be rather informal, whereas formalisation will be achieved on lower levels of abstraction. Methods that are suitable for strategic anaysis such as linguistic analysis, the use of metaphors and communication analysis have been discussed in [11].

The rationale behind our approach is an observation made in theoretical linguistics [3]: whenever a complex construction has to be explained, humans first think in terms of concepts, which are then mapped to a linguistic construct and only finally translated into sentences. Carrying this idea

* The work in this paper was supported by MU/ABRF grant 57413 "Foundations of Conceptual Modelling".

over to WIS development means to first lay out the fundamental concepts that are to be captured by the system, then map them onto a conceptual model, before finally approaching an implementation using common available technology.

We will not go into the details of the conceptual modelling of WISs, as [12] contains a detailed account of storyboarding, and content and functionality modelling (see also [13]). We will, however, show how the strategic model impacts on the formation of a WIS in terms of its layout and playout (see Section 3). For this we start from the means that are available for visual communication such as visual ordering, partitioning, colouring and perspective. We then go into more details discussing grid models and colour selection in accordance with the layout, atmosphere and progression patterns identified on the strategic layer.

2. The Strategic Layer of WISs

We start with a brief account of WIS modelling on a strategic layer. For a discussion of the Abstraction Layer Model see [12].

2.1. Mission Statement and Brand

The *brand* of a WIS is based on a rough classification scheme for WISs. This classification scheme has the form $\mathcal{P}^{\mathcal{W}}2\mathcal{U}^{\mathcal{A}}$ and represents in an extremely terse form the following very general information:

- $\mathcal{P}$ stands for "provider", and thus indicates which role the system plays. This specifies very roughly what kind of content can be expected from the system. For instance, if the provider is a bank, the provided services will most likely center around accounts, investments, savings and loans.
- $\mathcal{W}$ stands for "what", and thus adds more detail to the kind of content offered by the WIS. For instance, if the provider is a bank, the provided content may just be accounts, investments, savings, loans, and mortgages.
- $\mathcal{U}$ stands for "user", and thus indicates to whom the services offered by the WIS are directed. For instance, if the provider is a bank, the users are probably just the customers, enterprises or other banks.
- $\mathcal{A}$ stands for "actions", and thus indicates the functionality of the WIS offered to its users. For our example of a bank offering accounts, investments, savings, loans, and mortgages possible actions can be apply_for_loan, apply_for_mortgage, set_up_account, buy_stock, etc.

The brand is usually the result of a brainstorming activity discussing the what, whom and for_whom of the WIS. The aim is to fill these general placeholders $\mathcal{P}$, $\mathcal{W}$, $\mathcal{U}$ and $\mathcal{A}$ with meaningful terms that describe the WIS in very general, terse terms. The brand gives a rough picture of the content and functionality of the WIS and its users using only descriptive keywords. This is, however, a valuable source of information for refinement using linguistic methods.

The *mission statement* complements the brand by an informal, textual description. The importance of having it was emphasised in [4]. Each of the actions in the brand are taken as the major tasks. For each of them the mission statement decribes, which types of users are involved, which activities they are supposed to execute, which content will be provided for them and requested from them, and what will be the results of these activities. However, no attempt is made to decompose the tasks or to refine them, as this is left to storyboarding [12].

Furthermore, the mission statement contains metaphors that turn out to be adequate for describing the activities associated with the WIS. These metaphors refer to the content and functionality keywords used in the brand.

In addition to its descriptive character the mission statement also has an explanatory character in the sense that it contains the reasons for setting up the WIS. That is, the mission statement will describe what the major and minor purpose of the system is, how each task will contribute to these purposes, and what the benefits of the system for the provider and the users will be.

EXAMPLE 2.1 Let us consider the example of a WIS that deals with loan applications. In the case the provider is a bank, and the content will be centered around (personal) loans and mortgages. The only users we think of are customers, and the tasks they execute are applications for loans and mortgages, respectively. This leads to the following brand:

$$\text{bank}^{\text{loan, mortgage}}2\text{customer}^{\text{apply\_for\_loan, apply\_for\_mortgage}}$$

We omit the mission statement, which would be just an informal explanation for the brand.

In general, it is sufficient to formulate the mission statement using free-form text, but it is also possible to use semi-formal structured text. In doing so the brand and mission statement take the following form:

`Content`:	⟨list of content items⟩
`Users`:	⟨list of users⟩
`Tasks`:	⟨list of tasks⟩
`Major Purpose`:	⟨textual description⟩
`Minor Purpose`:	⟨textual description⟩
`Benefits`:	⟨textual description⟩

Furthermore, we obtain the following informal description for each of the tasks:

`Task`:	⟨task name⟩
`Description`:	⟨textual description⟩
`Participants`:	⟨list of users⟩
`Required Content`:	⟨list of content items⟩
`Produced Content`:	⟨list of content items⟩
`Result`:	⟨textual description⟩

EXAMPLE 2.2 Using the tabular semi-formal description, we can rewrite the brand and mission from Example 2.1 in the following way:

`Content`:	loan, mortgage
`Users`:	customer
`Tasks`:	apply_for_loan, apply_for_mortgage
`Major Purpose`:	open an additional sales channel, for technology-experienced and informed, goal-oriented customers
`Minor Purpose`:	improve banking efficiency in loan sector
`Benefits`:	closer binding of customers attraction of new customers improvement of cost efficiency increased availability of bank services

2.2. Utilisation Space

The term "utilisation space" is used as a metaphor to characterise the WIS as a space, through which a human user can navigate. As such it has to cover mainly the content, functionality and context in general terms. The goal is to enable optimal orientation in the utilisation space, such that searching and finding information needed for certain tasks will be facilitated.

The type of content is already characterised by the brand, to be precise by its what-part. This gives a set of nouns describing the content in coarse terms. Similarly, the what_for-part of the brand gives verbs describing the functionality, i.e. what to do with the content.

The utilisation space will now add details to content and functionality, set the nouns and verbs used in the brand into relation, and place both into a utilisation context. This will be done in the following way:

- Refine the content keywords and place them in semantic relationships. These relationships can capture specialisation, part-of relationships, or associations of global context with details. They indicate navigation facilities and order principles among the content. Word fields are a valuable tool for the refinement [11].
- Refine the functionality keywords to discover various facets that can be placed in semantic relationships in the same way as the content. Again, word fields are a valuable tool for the refinement.
- Relate the functionality with the content, i.e. specify in which context a particular content is needed, i.e. which content is needed by which activity, which content is produced by which activity, in which order (if any) the content will be used by an activity. In doing so, we obtain a progression model for the functionality.

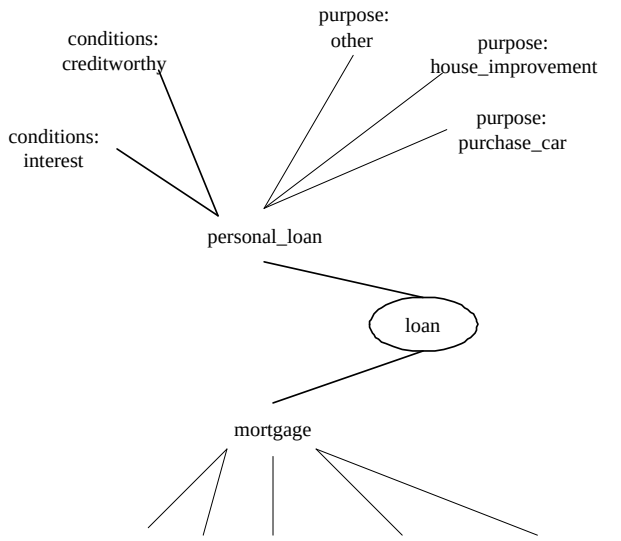

Figure 1. Semantic tree of content items in loan application

Semantic relationships for content – and similarly for functionality – can be represented by rooted trees, where the root is defined by a keyword taken from the brand. Thus, we can obtain the following more detailed description of content items:

`Content Item`:	⟨name⟩
`Derived From`:	⟨content item⟩
`Relationship`:	⟨description⟩
`Usage`:	⟨list of tasks⟩
`Description`:	⟨textual description⟩

In the same way we obtain more detailed description of tasks:

`Task`:	⟨task name⟩
`Derived From`:	⟨task name⟩
`Relationship`:	⟨description⟩
`Description`:	⟨textual description⟩
`Participants`:	⟨list of users⟩
`Required Content`:	⟨list of content items⟩
`Produced Content`:	⟨list of content items⟩
`Result`:	⟨textual description⟩

EXAMPLE 2.3 Figure 1 represents a tree of content items with root loan, which is specialised by personal_loan and mortgage. Details for mortgages have been omitted, but for personal loans the three decisice facets – conditions for obtaining the loan, i.e. creditworthiness, conditions for loan repayment, i.e. life span, principal and interest, and loan purpose have been indicated.

Similarly, Figure 2 represents a task tree with root apply_for_loan, which again is specialised by apply_for_personal_loan and apply_for_mortgage. For the latter one further details have been omitted. For apply_for_personal_loan the components contributing to selecting terms and conditions, outlining personal finances, declaring the purpose of the loan, and entering customer details are shown in the tree.

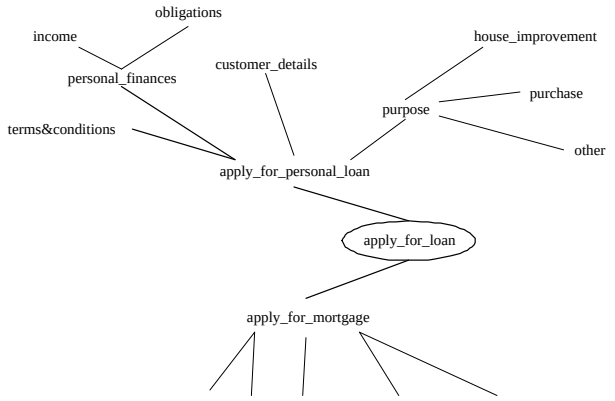

Figure 2. Semantic tree of tasks in loan application

The decisive criterion for orientation in the utilisation space is the preservation of context. This can be achieved, if for each task a mental model can be constructed. This model captures the progression of content perception over a time axis. It represents a system map for the user showing information that has already been used and processed.

For strategic modelling it is important to reflect, whether the stages that represent perception of content are logically connected. It is also important to see that later stages enable the possibility to be redirected to information that was already processed earlier.

If available, metaphors may help to set up this mental model. For instance, the "desktop" metaphor is a well-established tool that has significantly contributed to ease the use of computer technology by technical laypersons. Similarly, the "shop" metaphor is frequently used in commerce applications.

EXAMPLE 2.4 We may regard the task of personal loan application in the loan application system from Example 2.1 as a matching problem. We have to match the needs of a customer with the purposes og the available loans, and the payments arising from loan conditions with the payment latitude of the customer, which is determined by creditworthiness conditions set by the bank and the personal finances, i.e. income and obligations, of the customer.

Thus, a suitable mental model consists of the set of loan options, i.e. loan type, conditions and purpose, and the implications of each option with respect to the payments. This model develops over time in a way that non-suitable options are deleted first, then a selection is made, and finally organisational data are added to turn the selected loan option into a full loan application.

2.3. Utilisation Portfolio

The utilisation portfolio complements the utilisation space emphasising the whom-part of the brand. Thus, it is mainly concerned with the WIS users, for which we will later adopt the term "actor", their goals and the tasks that have to be executed to achieve these goals.

Tasks correspond to the actions in the brand and their refinement in the utilisation space. So we can assume a task hierarchy emphasising specialisation between tasks and decomposition of tasks into subtasks, as long as these can be described in a simple way.

The users used in the brand and mission statement will be roughly classified according to roles they have with respect to the WIS. Each role has particular goals, and each of these goals corresponds to a task that is meant to achieve this goal. This does not mean that the task has to be executed by the user in this role; it may well refer to tasks executed by users in other roles. Tasks are broken down into subtasks to a level that elementary tasks can be associated with a single role. In addition, subtasks should refer to subgoals.

Furthermore, we obtain dependencies between goals, e.g. being a subgoal, a specialisation, or any other kind of dependency. Thus, we complement the informal description of the system by adding goals:

`Goal`:	⟨goal name⟩
`Derived From`:	⟨goal name⟩
`Relationship`:	⟨description⟩
`User`:	⟨role name⟩
`Description`:	⟨textual description⟩

The relationship between tasks, roles and goals can be represented in a graph, which we call a *task-goal graph*. In these graphs we have three different types of vertices for actors, tasks and goals, respectively. Furthermore, we have five different kinds of edges for task-goal relationships, in-

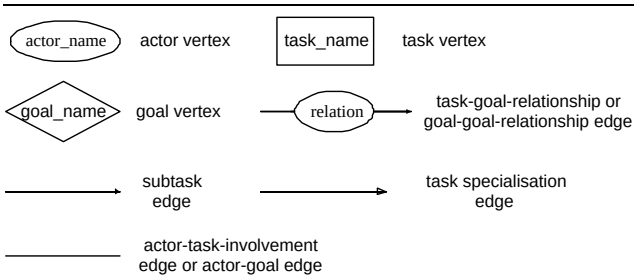

Figure 3. Legend for Task-Goal graphs

volvement of an actor in a task, goal-goal-relationships, as well as for subtasks and task specialisation. Figure 3 shows the legend for such graphs.

EXAMPLE 2.5 The task-goal graph in Figure 4 illustrates goals, tasks and actors in a loan application. In this case the goal buy_new_car depends on obtaining a personal loan, i.e. on the goal personal_loan. Sufficient for achieving this goal is the successful execution of task approve_personal_loan, which has to be done by a bank clerk. However, this task will only be triggered by a task apply_for_personal_loan to be executed by the customer. This task is thus necessary for the goal. Furthermore, the task decomposes into four subtasks select_conditions, enter_customer_details, declare_purpose and set_up_budget.

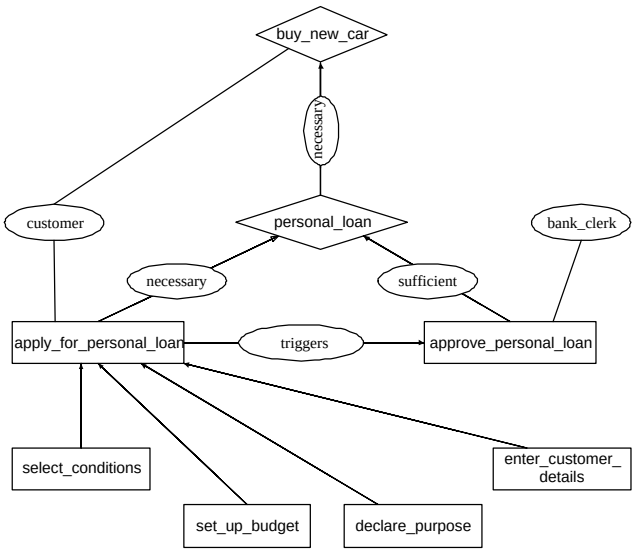

Figure 4. Task-Goal graph for loan application

A valuable tool for setting up the utilisation portfolio is communication analysis, which addresses how a user will communicate with the WIS and why this communication is the best for the provider and the user.

2.4. The Atmosphere of a WIS

While the brand, mission statement, utilisation space and utilisation portfolio aim at the characterisation of content, functionality and usage of the WIS in strategic terms, the atmosphere addresses the gestalt of the WIS, i.e. how the WIS should be configured. At the end the WIS will be implemented by and presented through web pages, which should convey a uniform impression to the WIS users.

Categories characterising the impression of pictures can be used such as energetic, romantic, elegant, refreshing, harmonic, or stimulating. Each of these categories will have implications on the choice of form and colour [10]. On the strategic level the choice of one of these categories corresponds to the question which impression the WIS shall convey. The question is, which atmosphere is best suited for the envisioned content and functionality.

EXAMPLE 2.6 In Example 2.4 we characterised the application for loans as a matching problem. Choosing a harmonic presentation for the WIS would suggest the intention to find an "optimal" solution for the customer and the bank. Choosing a stimulating atmosphere might suggest to encourage the customer in his/her application. An elegant atmosphere of the WIS may be chosen to convey confidence to the customer, i.e. that his/her financial affairs will be dealt with in the best way.

In addition, the atmosphere of a WIS is concerned with the progression patterns for the tasks. These patterns reflect the logical connection of information revealed to the user during the execution of a task. We distinguish between the following *progression patterns*:

- A *circular* progression pattern is centered around a particular content item, the phases of which form the core of the content to be delivered. At each stage details are added.
- A *loop* progression pattern emphasises the iteration of content, each time taking a different perspective. This is similar to a circular pattern, however puts more emphasis to changes.
- An *incremental* progression pattern emphasises the development of several content items over time. At each stage some of the items may be completed.
- An *evolutionary* progression pattern emphasises the stages of the content items, in particular those that are used in the result of tasks. At each stage content item may still be incomplete.

- A *network* progression pattern emphasises the flexible treatment of content items during the development of tasks and the logical connection between various such objects.

EXAMPLE 2.7 If we choose a circular progression pattern for a personal loan application, the presentation of information will be centered around the loan, each time gaining a clearer picture of the result. In Example 2.4 we explained that it would be a good idea to successively discard possible loan options. This is in accordance with circular progression.

An incremental progression pattern would emphasise the various components of a loan application, i.e. the customer data, the conditions, and the budget. Each of these components would be treated as a separate item. While this is common in many WISs that offer form-oriented access, it may not be the best choices, because it does not support well the interaction between a tentative choice of conditions, the calculation of repayment costs and the personal financial situation of the customer.

An evolutionary progression would be very similar to an incremental one. However, each component could be left incomplete. This would help with the problem of changing conditions.

A loop progression would be similar to a circular progression. The difference is that the circular progression emphasises more the narrowing of options, while a loop progression would permit returning to options that have already been discarded.

Finally, a network progression is again similar to an incremental one, but leaves much more flexibility, as to fixed order of the components is presumed.

Progression patterns have a direct impact on the placement of content on web pages, thus on the tiling of pages and the mapping of content to the tiles. Such a mapping of patterns to web page grids is discussed in detail in [10].

3. Visual Design Patterns

We will now discuss how the decisions made on the strategic level impact on the formation of the WIS presentation. We may assume that the result of conceptual WIS modelling is available, i.e. there is a clear specification, which content is to be presented, and how the different content units are logically linked together. This is captured by the *media schema* in [12]. So our focus now in on how the WIS content and functionality is to be presented to its users. We concentrate on the visual design, though audio design could be tackled as well.

3.1. Basic Principles

Visual design patterns are composed of visual and functional building blocks. The visual building blocks correspond to the geometric partitioning of the screen, whereas the functional building blocks realise the access to the presented content. Thus, the functional building blocks correspond directly to the represented content and its organisation along the strategic progression patterns. They order the content, whereas the visual building blocks place the content on the screen using a flexible graphical structure with constant colour coding and repeating elements that reflect the functional order.

Within the limitations of this paper we concentrate on the visual building blocks.In doing so, we have to consider the following three aspects:

- the visual alignment and partitioning of the screen;
- the colouring with respect to funcionality and aesthetics;
- and the perspective perception of the whole screen.

The visual alignment is based on a tiling of the screen as a two-dimensional surface. In general, we can divide the horizontal and vertical axes using grid points $x_{\min} = x_0 < x_1 < \cdots < x_k = x_{\max}$ and $y_{\min} = y_0 < y_1 < \cdots < y_\ell = y_{\max}$. A *tile* is defined by a rectangular region $[x_i, x_j] \times [y_{i'}, y_{j'}]$. Then use a partition of the whole screen into tiles.

EXAMPLE 3.1 A very common tiling is obtained by using just 4 horizontal grid points $x_0 < x_1 < x_2 < x_3$, and only 3 vertical grid points $y_0 < y_1 < y_2$. Then define four tiles

$$\begin{aligned} \text{up} &= [x_0, x_3] \times [y_1, y_2] & \text{left} &= [x_0, x_1] \times [y_0, y_1] \\ \text{middle} &= [x_1, x_2] \times [y_0, y_1] & \text{right} &= [x_2, x_3] \times [y_0, y_1] \end{aligned}$$

Usually, the “up” tile is used for some menu bar, the “left” tile for navigation links, the “middle” tile for the major content, and the “right” tile for side options.

The colouring scheme will be the major instrument to achieve the desired atmosphere as specified in the strategic model. For the perception of the presentation by a user the interaction of colours is decisive. The basis can be a *colour chord* consisting of n colours ($n \in \{2, 3, 4, 6\}$) that form a regular polygon in the colour circle. These are complemented by adding grey tones enabling to achieve contrast between light and dark colours. A *quality contrast* results from brightening the coulours of a chosen ground chord in the same way.

From the theory of colour harmonics it is known that the choice of colour chord impacts directly on the perception

leading to the sensations such as warm, cool, cold, intensive, hot, light, dark, etc. Conversely, the desired sensation indicates guidelines for the choice of the colour scheme.

The colouring scheme also impacts on achieving a three-dimensional impression (if desired) or not. The technical means for achieving a perspective perception are contrast, colour perspective, depth of sharpness, and the differentiation of motif [10].

3.2. Grid Geometry

Following the general principles the grid geometry addresses the visual alignment. We now leave the simple tiling into several columns and rows aside and concentrate on grids, in which the visual building blocks have sizes following a rhythmic structure that can be expressed by a sequence of positive integers. Such grid models can then be combined with colour schemes in a way that the rhythmic proportions of the colours conform to the desired atmosphere.

We are particularly interested in the Fibonacci sequence, which is defined by the recurrence $f_{n+2} = f_n + f_{n+1}$ with the starting values $f_1 = f_2 = 1$. This gives the well-known sequence $1, 1, 2, 3, 5, 8, 13, \ldots$ Solving the recurrence equation gives

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

involving the two roots of the equation $x^2 - x - 1 = 0$. The positive root is known as the *number of the golden section*, which played an important role in art and architecture, and appears naturally in nature.

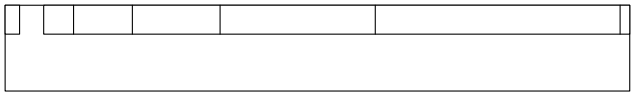

Figure 5. The grid model of visual flags

A simple use of the sequence leads to the model of *visual flags*, which in fact dates back to Leonardo da Vinci. This is illustrated in Figure 5. The flags enable the selection of sections that are ordered according to some functional criteria.

In terms of tiling we simply use the Fibonacci numbers as the horizontal grid points, and do not bother about vertical grid points at all. Multiplying the Fibonacci numbers with a constant can be used to define a partition of the screen width.

Another use of the Fibonacci sequence is to place squares with increasing side length f_i along a spiral as shown in the lower part of Figure 6. It is therefore called the *Fibonacci gid model*. In doing so we obtain

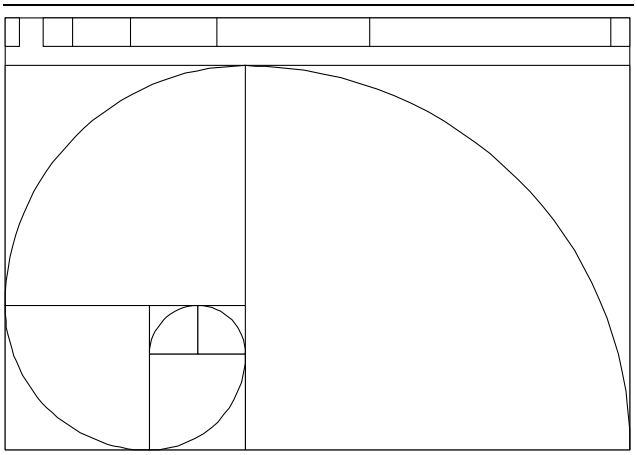

Figure 6. The Fibonacci grid model with flags

a very nice tiling of the screen, in which the proportions of the square tiles are determined by the Fibonacci sequence. If each tile is associated with a colour of a well chosen colour scheme this enables to achieve the desired atmospheric effects as specified in the strategic model. The thesis [10] shows this combination of the Fibonacci grid with various colour schemes in different web application projects[1].

The Fibonacci grid can be combined with visual flags as shown in Figure 6. In doing so we obtain a visual alignment with the following characteristics:

- It can be combined with a harmonic colouring due to the proportions of the tiles that are defined by the Fibonacci sequence. This has been often exploited in art.
- Its visual flags can be exploited for global navigation according to the functional specification.
- The implicitly given Fibonacci spiral can be used as a line along which the content progresses.

3.3. Atmospheric Effect of Colour Schemes

On the strategic level we specified the desired atmosphere of a WIS. For this we used categories such as energetic, romantic, elegant, refreshing, harmonic and stimulating. All these categories refer to a desired sensation that the WIS presentation should convey to the users. The question is, which colour schemes can be identified to support these desired effects.

For each category we first need a ground colour chord, which is extended to a quality contrast. In particular, colours

1 Unfortunately, it is difficult to show the effect of colour schemes in a black and white text.

will be brightened uniformly. Based on these base decisions we develop an ordering of colours. The colour ordering in combination with the association to grid tiles determines the overall effect of the colour scheme. The colour ordering is based on formal and functional criteria:

- Formally, the order is determined by the placement of the colours in the colour circle. It is known that equidistant colours harmonise better.
- Functionally, the order of colours is determined by the subjective sensation and associations of a viewer. This is taken from psychological studies and centuries of experience in art.

An energetic or powerful atmosphere can be achieved using a three-colour chord with bright and high-croma colours such as a blueish purple with green and orange. The blueish purple colour with increasing brightness can be used for the visual flags. Orange with increasing brightness can be applied to the initial squares on the Fibonacci spiral, where green is reserved for the largest tile. However, a tile will always inherit the colour of its corresponding flag.

A romantic atmosphere can be achieved using a three-colour chord with pastell colours such as (light) blue, pink and yellow. Similarly to the energetic atmosphere yellow and blue in increasing brightness would be used for the flags and the initial tiles along the Fibonacci spiral, where pink would be reserved for the largest tile.

An elegant atmosphere can be achieved using a two-colour chord, e.g. red and green, in combination with grey. As before, grey would be reserved for the largest tile in the Fibonacci grid, while the other colours in increasing brightness would be used for the initial tiles and the flags, respectively.

A refreshing atmosphere can be achieved using a three-colour chord with light colours in a "temperature contrast", i.e. the individual colours show opposite effects with respect to associated temperature. For instance, we might choose (light) purple, yellow-orange, and a blueish green such as cyan. The use on the Fibonacci is as before with yellow-orange for the largest tile.

A harmonic or balanced atmosphere can be achieved using a two-colour chord, e.g. dark blue and ochre, in combination with grey with grey being used for the largest tile. Similarly, a stimulating atmosphere can also be achieved using a two-colour chord, e.g. yellow-green with red-purple, in combination with grey.

4. Conclusion

In this paper we discussed the strategic modelling of web information systems on the basis of the abstraction layer model from [12]. The starting point is an informal mission statement and a characterisation of the brand of the WIS, the latter one following a general classification scheme. Both together describe what the WIS is about, i.e. the purpose(s) of the system, and both result from brainstorming.

Going more into details we presented models of utilisation space and utilisation portfolio, which describe in very general terms the content that is to be presented in the WIS, the functionality, with which this content can be accessed, as well as the users of the WIS, their goals, and the tasks that have to be performed to reach these goals. The fourth and last part of a strategic WIS model are general rules for the layout, atmosphere and progression of the system based on knowledge about the cognitive perception of form, colour and other style elements. We then showed how the strategic model impacts on the formation of a WIS in terms of its layout and playout. Based on general principles for visual design patterns we discussed grid models and colour selection in accordance with the layout, atmosphere and progression patterns identified on the strategic layer.

The strategic model complements the conceptual model of WISs that was presented in [12]. Furthermore, it shifts interface design to a higher-level of abstraction in a way that permits intentions, metaphors and context information to be exploited from the very beginning, whereas it is usually left only to the generation of pages using style patterns. This enables a much tighter coupling of the abstraction layers in WIS development.

References

[1] P. Atzeni, A. Gupta, and S. Sarawagi. Design and maintenance of data-intensive web-sites. In *Proceeding EDBT'98*, volume 1377 of *LNCS*, pages 436–450. Springer-Verlag, Berlin, 1998.

[2] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, San Francisco, 2003.

[3] N. Chomsky. *Lectures on Government and Binding*. Mouton de Gruyter, Berlin, 1993.

[4] O. De Troyer. Designing well-structured websites: Lessons learned from database schema methodology. In T. Ling, S. Ram, and M. Lee, editors, *Conceptual Modeling- ER'98*, volume 1507 of *LNCS*, pages 51–64. Springer-Verlag, 1998.

[5] O. De Troyer and C. Leune. WSDM: A user-centered design method for web sites. In *Computer Networks and ISDN Systems – Proceedings of the 7th International WWW Conference*, pages 85–94. Elsevier, 1998.

[6] F. Garzotto, P. Paolini, and D. Schwabe. HDM - a model-based approach to hypertext application design. *ACM ToIS*, 11(1):1–26, 1993.

[7] J. Gómez, C. Cachero, and O. Pastor. Modelling dynamic personalization in web applications. In *Third International Conference on Web Engineering – ICWE 2003*, volume 2722 of *LNCS*, pages 472–475. Springer-Verlag, 2003.

[8] G.-J. Houben, P. Barna, F. Frasincar, and R. Vdovjak. HERA: Development of semantic web information sys-

tems. In *Third International Conference on Web Engineering – ICWE 2003*, volume 2722 of *LNCS*, pages 529–538. Springer-Verlag, 2003.

[9] D. Lowe, B. Henderson-Sellers, and A. Gu. Web extensions to UML: Using the MVC triad. In S. Spaccapietra, S. T. March, and Y. Kambayashi, editors, *Conceptual Modeling – ER 2002*, volume 2503 of *LNCS*, pages 105–119. Springer-Verlag, 2002.

[10] T. Moritz. *Visuelle Gestaltungsraster interaktiver Informationssysteme*. PhD thesis, BTU Cottbus, Cottbus, Germany, 2005. to appear.

[11] K.-D. Schewe, R. Kaschek, C. Wallace, and C. Matthews. Emphasizing the communication aspects for the successful development of electronic business systems. *Information Systems and E-Business Management*, 3(1):71–100, 2005.

[12] K.-D. Schewe and B. Thalheim. Conceptual modelling of web information systems. *Data and Knowledge Engineering*, 54(2):147–188, 2005.

[13] K.-D. Schewe and B. Thalheim. Engineering web information systems. Tutorial Notes, ICWE 2005, Sydney University of Technology, 2005.

[14] D. Schwabe and G. Rossi. An object oriented approach to web-based application design. *TAPOS*, 4(4):207–225, 1998.

Database Integration and Querying in the Bioinformatics Domain

Bob Myers[1,2,5], Trevor Dix[1,2], Ross Coppel[1,3], David Green[1,4]
1:Victorian Bioinformatics Consortium.
2:School of Computer Science and Software Engineering, Monash University, Australia
3:Dept. of Microbiology, Monash University, Australia.
4:Faculty of IT, Monash University, Australia.
*5:*bob.myers@med.monash.edu.au

Abstract

Given the exponential growth in the amount of genetic data being produced, it is more important than ever for researchers to have effective tools to help them manage this data. This paper describes a system that enables users, generally biologists, to construct components to answer specific questions in their field. The system allows the creation of modules and submodules via top-down decomposition. Concepts and terms can be defined through conversation. These are then used when composing base-level functions to produce code for modules and for interfacing modules.

1. Introduction

For more than a decade the quantity of genetic data produced each year has been growing exponentially. GenBank's database has grown from just over 217 million base pairs in 1994 to more than 44 billion in 2004 [1], increasing on average by a factor of 1.7 each year. PubMed now contains over 15 million citations [2] and is growing by approximately 450,000 each year. Not only is the quantity of data increasing but so too is the number of databases, web-sites and other information sources that a researcher must cope with. Also, given the ever increasing number and speed of modern DNA sequencing machines, micro-arrays and other devices, it seems unlikely that the flood will abate any time soon. Faced with such an information overload, researchers need computerized tools to assist them in making best use of the available data.

The basic problem is to build a system that enables users, generally biologists, to construct components to answer specific questions in their field. The following list provides examples of questions in the context of the malaria parasite *Plasmodium falciparum*:

- Find a list of all proteins that have predicted trans-membrane anchor sequences and are expressed in asexual blood stages.
- Find secreted proteins that have disulphide bonds.
- Find proteins with repeats.
- Find allelic variants of this surface protein that have been identified in Thailand.
- Find homologs in other species and home-in on conserved sequences.

The authors created a prototype of a system intended to fulfill some of these needs. Based on a workflow style graphical interface where the user drags modules from a palette of tools onto a drawing frame and then connects the outputs of some modules to the inputs of other, as shown in figure below.

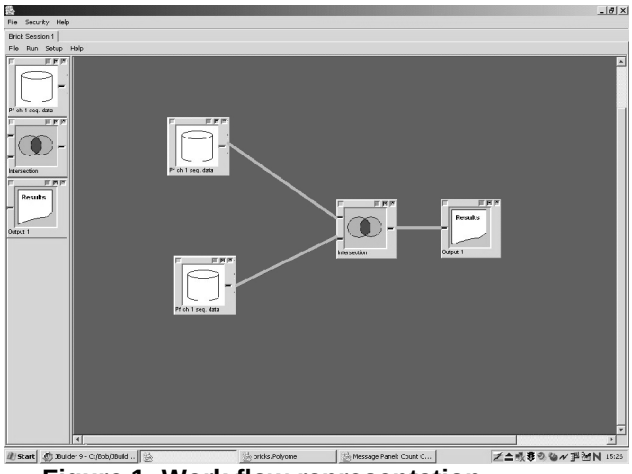

Figure 1: Work flow representation

Sufficient modules were programmed to enable the system to answer some real-life biological questions in the domain mentioned above. In the course of developing the

system some significant problems were encountered including:

- Modules tend to be extremely complex, requiring much time and expertise to build.
- Some of the modules were very fragile. A slight change in a data source, for example, could cause the module to fail.
- A new module has to be created for each new data source.

This prototype is described in more detail in section 2.

To overcome these problems it was decided to break the large modules down into many small units, each one representing a very basic piece of functionality; for example, read an HTML page, send an SQL query to a relational database, find a substring of some text etc. Because each module is now trying to do much less, it requires fewer lines of code and is less complex. There are fewer ways in which a small module can go wrong, making it easier to trap errors. There will of course be more modules. However, the system allows decomposition of modules into submodules. Indeed most common workflow modules will be sequential, permitting a functional definition for each in terms of input(s) and output(s).

The creation of modules and submodules is typical of top-down decomposition. This allows consideration of (sub)modules in isolation, thus keeping complexity manageable. The coding within (sub)modules is a mixture of top-down and bottom-up. The system allows concepts, terms etc to be defined through conversation. However the base-level functions are composed in a bottom-up manner applying to the defined terms. Section 3 presents the conversational process within the system.

2. Related Methods

The problem of data integration has been approached in many ways, some of which are described below. There is a degree of overlap between some of these approaches, and many solutions to the problem of data integration consist of combinations of these approaches.

- Data Warehousing: Data is extracted from its original location and added to a common, centralized database. The main problems with this approach is that the quantity of data is so large that it would require massive amounts of computing power to handle it, and there are so many different types of information sources that a vast programming effort would be required to write the extraction routines.

- Standardised Databases: Several attempts have been made to come up with a standard (relational) database structure to suit all purposes within the Bioinformatics field. Examples of this are GUS[3] (used for PlasmoDB, AllGenes, EPConDB, GeneDB etc.), ACEDB[4] (used for WormBase, DictyDB etc) and GMOD[5] (used for WormBase, FlyBase, MGI, SGD, Gramene, Rat Genome Database, EcoCyc, and TAIR). The main problem with this approach is finding a format that suits everyone, otherwise leading to a proliferation of standards, defeating the original purpose.

- Federated Databases: Data is not amalgamated into a data warehouse, instead it remains in its original location and is retrieved on demand. This approach requires interfaces to each data source to be constructed. This is manageable if all the data sources are of a similar type, e.g. relational databases, but it can become impractical if the data sources vary wildly in type and structure, requiring expertise in both data analysis and the subject matter of the data sources.

- Web Services: Applications are made available for automatic use by other systems via a network. The main problem here is that each data source would need to provide its own services and make them generally available.

- Database Wrappers: Similar to the idea of federated databases, this consists of a piece of software that interfaces between the user and the target database. This software receives queries and returns replies in a standard format, enabling the user to query multiple data sources in the same way regardless of the target's type. This concept requires the software to be specifically built for each target, requiring expertise in data analysis and the subject matter.

- Semantic Web: Consists of annotating data sources in a standardized, machine readable way, so that systems can extract meaning from them (e.g. the types of entity represented in the data and how they relate to each other). Contrast this to the World Wide Web, where the data sources (web pages) have content, but little machine readable meaning. Few bioinformatic data sources have done this so far, although a few do provide data in XML as well as HTML.

Our initial model for an information system was based on a combination of the above approaches, and was developed from the observation that people often draw flow diagrams to describe the information flow and processing that they wish to perform. The system interface consists of a palette of tools and a drawing pane, as shown

in figure 1 above. This work-flow type of interface has been used in a number of other systems, for example, DiscoveryNet[6].

Each tool is written as a separate module to perform a specific task, like read from a specific database or combine two sets of data etc. Each tool may have data-streams as inputs and outputs, and a set of parameter type inputs used to tailor the specifics of its functionality. To perform some work the user drags tools from the palette onto the drawing frame, sets the values of any parameters and then joins the output data-streams of modules to the input data-streams of others to produce a work-flow type diagram. When the work-flow is executed, all modules with no input data-streams (generally functions like reading from data sources) are run first, each in a separate execution thread, and their output data-streams are passed onto the appropriate modules as indicated by the arcs in the diagram. When a module has received a sufficient amount of its input data-streams, it too runs and so on until all modules have been executed. The terminal modules of the work-flow are generally those that produce some sort of output for the user (writing a file, producing a printout etc) and so do not pass their output data streams onto another module. The example of a simple work-flow in figure 1 shows two databases being read, their outputs combined and displayed on the screen.

The initial system was set up with several representative modules, including:

- Run a TMPRED query (Prediction of Transmembrane Regions and Orientation) at EMBNet in Switzerland (http://www.ch.embnet.org/software/TMPRED_form.html).
- Run a SignalP query (presence and location of signal peptide cleavage site prediction in amino acid sequences) at the Center for Biological Sequence Analysis at the Technical University of Denmark (http://www.cbs.dtu.dk/services/SignalP-2.0/).
- Count amino acids in a protein and selecting proteins with counts in specified ranges.
- Create the intersection of two sets of proteins.
- Read a number of protein sequences from a FASTA format file.
- Format protein data for screen output.

The size and functionality of the modules were chosen based on the level of breakdown that a biologist would use when thinking of a real-world problem. For example, a question might be: *"Which secreted proteins have disulphide bonds?"*, and a biologist could break this down into the following parts:

a) Get protein sequences from file x.fasta
b) Count the number of Cysteine amino acids in each protein, keep only the proteins with 2 or more.
c) Feed those proteins into the SignalP service with parameters:
   i. Truncate at 50 residues
   ii. Organism group = Eukaryotes
   iii. Method = HMM

   and keep only those proteins that return a positive result.
d) Feed those proteins into TMPRED with parameters:
   i. Minimum length of hydrophobic part of trans-membrane helix = 17
   ii. Maximum length of hydrophobic part of trans-membrane helix = 33

   and keep only those that return a negative result.
e) Display those proteins.

It is easy to see how the modules listed above correspond to the steps in the process, which would be represented graphically as in figure 2:

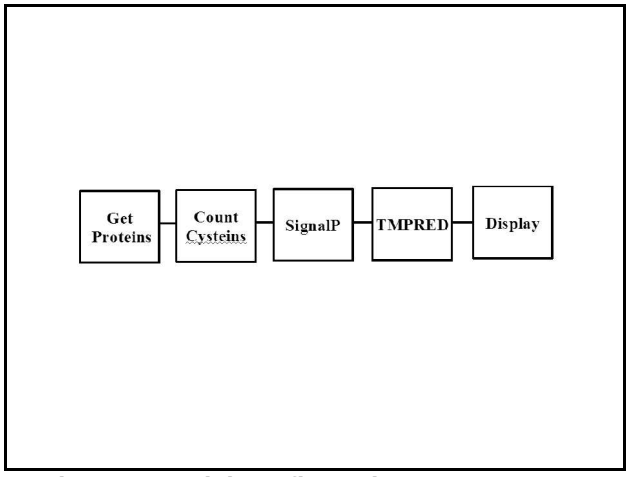

Figure 2: Serial configuration

It might also be advantageous to perform some of the steps in parallel to take advantage of distributed processing if tasks run on remote processors (as in the case of SignalP and TMPRED), in which case the process could be represented as in figure 3:

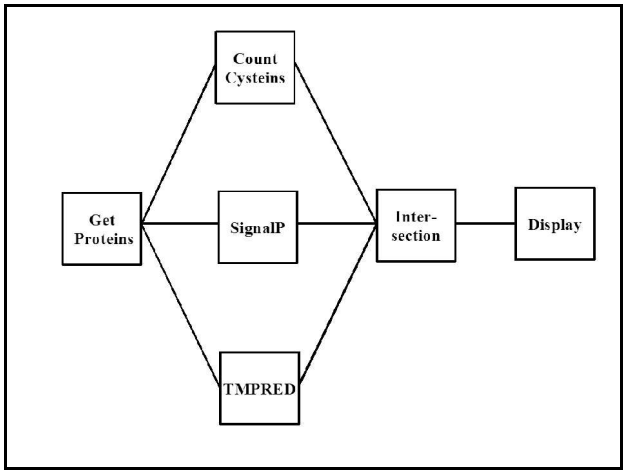

Figure 3: Parallel configuration

Although the system worked well, producing the expected results in a reasonable time-frame, a number of problems were found:

- **Complexity**: Many of the modules were very complex, requiring much time, effort and knowledge about the biological subject matter and computing techniques, making them too difficult for a biologist to produce.

- **Fragility**: Some of the modules (particularly those that interfaced with external systems) were very susceptible to minor changes in the target system, causing it to either fail or yield incorrect results. Because of the complexity of the module they also contain many possible points of failure. The number and variety of ways that a module could fail made trapping the errors and taking appropriate action difficult.

- **Variability**: A new module would need to be created for each new data source, and even sometimes for different questions posed to the same data source. Given the explosion in the number and types of bioinformatic data sources it would require a very significant effort to keep up.

These problems indicate that the development and ongoing maintenance load required for such a system would be impractical to maintain.

3. A Solution: PolyOme

To overcome the problems listed in section 2, it was decided to breakdown the high level modules, like "Run a SignalP query..." into base-level modules like "Get an HTML page" or "Run a remote CGI script". Smaller, simpler modules with limited functionality have the advantage of being:

- easier, quicker and cheaper to build,

- reusable, only need to be written once but can be used in many combinations with other small modules, and

- more robust, simpler modules have fewer ways in which they can go wrong, making error trapping easier.

However, having many small modules introduces another problem. It became correspondingly more difficult for the user to join together a large number of small modules compared to a small number of large ones. Essentially, the skill that the developer of the large module had used in writing it, now had to be shown by the user in linking the smaller modules together. Clearly another method of linking the modules was required.

After considering a number of GUI type interfaces, it was decided to use a text based interface, with English words as input. A text based interface would be superior in several ways:

a) It would be more flexible, allowing the system to cope with a wide variety of questions, statements and data sources.
b) Details of the workings of the functional modules could be hidden from the user.
c) The system could resolve ambiguities or request further information from the user via the same interface, establishing a conversation with the user.
d) The system could use the same process to handle non English type inputs such as HTML, XML, comma delimited data etc. enabling it to not only handle user input, but also to process the information returned from a data source.

It was decided not to use a full-blown Natural Language Processing (NLP) system because it would be unnecessarily complicated. The domain is simpler, not needing full NLP. Also, full NLP would not permit the handling of HTML, XML and other non English information.

PolyOme is essentailly a mechanism by which the user can compose submodules from base-level units and higher level modules from submodules by means of a conversational process. The general architecture of PolyOme consists of the user interface, a database, a

library of basic functions and a set of processes, see figure 4.

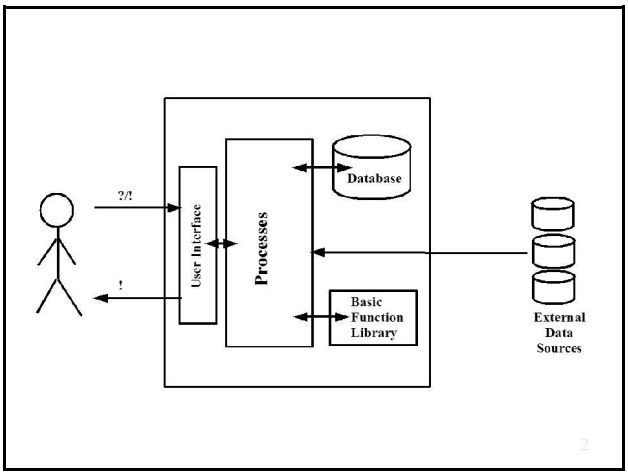

Figure 4: PolyOme general architecture.

The system is implemented as a single user, single database structure for the sake of simplicity.

- **User Interface:** The user interface is implemented as a simple two part screen, the user types input at the bottom and the system's reply appears at the top, as shown in figure 5.

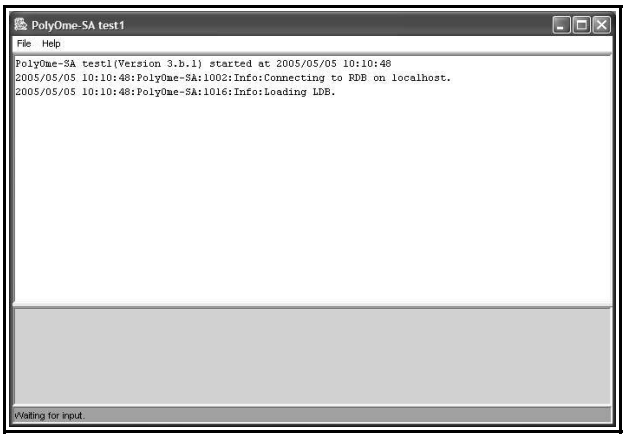

Figure 5: User interface

- **Database:** The database consists of two parts, a relational database (RDB) and a logic database (LDB). The LDB is used to contain facts about the subject matter and the RDB is used to record part-of speech tags and translation rules. Although the database could have been constructed using either a LDB or a RDB alone, a combination of the two was chosen so that each could be used for the tasks to which it is best suited. A LDB (implemented here in Prolog) is good for storing a wide variety of information, having no strict field/table structure, and the logic engine is good at inferring information from the stored facts. Both of these are difficult to do in a relational database. However, Prolog tends to be slow, so tasks that require little or no use of the inferencing engine and have well defined data structures would be better performed by the RDB.

- **Basic Function Library:** The basic function library consists of a number of Prolog functions, and provides the basic capabilities of the system, such as reading relational databases, reading HTML pages etc. Some of these are programmed as Java classes, which are then registered with the Prolog engine as 'built-in' functions, and some are written as pure Prolog.

  These functions are purely for internal use within the system. They are hidden from the user by the User Interface and the process ProcessMsg, which is described below.

  The following two Java functions provide the system with the ability to read any ODBC data source and to add that information into the LDB.

  readODBC/4, which reads an ODBC data source. Given the data source name, table name and a list of fields it returns a list of values corresponding to the rows and columns extracted from the table.

  factise/3, which converts a list of row and column values (as returned by readODBC/4 for example) into facts which are added to the LDB.

  An example of a pure Prolog function is xisa/2:

  ```
  xisa(X,Y):-isa(X,Y).

  xisa(X,Y):-isa(X,Z),xisa(Z,Y)
  ```

  The function isa(X,Y) corresponds to the English statement "X is a Y". xisa(X,Y) extends this to say that if "X is a Z" and "Z is a Y" then "X is a Y". For example, if it is known that isa(human,mammal) and isa(mammal,vertebrate) (i.e. "human is a mammal and a mammal is a vertebrate) then the question "isa (human,vertebrate)" gets the answer "No", however asking "xisa(human,vertebrate)" gets the answer "Yes".

  Several other functions have been written in Java to assist in the system's processing. These include:

checkPoint/1, to take a snapshot of the LDB.

rollBack/1, to restore the LDB from a previous checkPoint snapshot.

These functions are required to enable the system to test sets of updates to the LDB and back them out if required. checkPoint/1 is also used to ensure LDB persistence from one user session to the next.

- **Processes:** The main process, Converse, is responsible for conducting a dialog with the user, see figure 6 below.

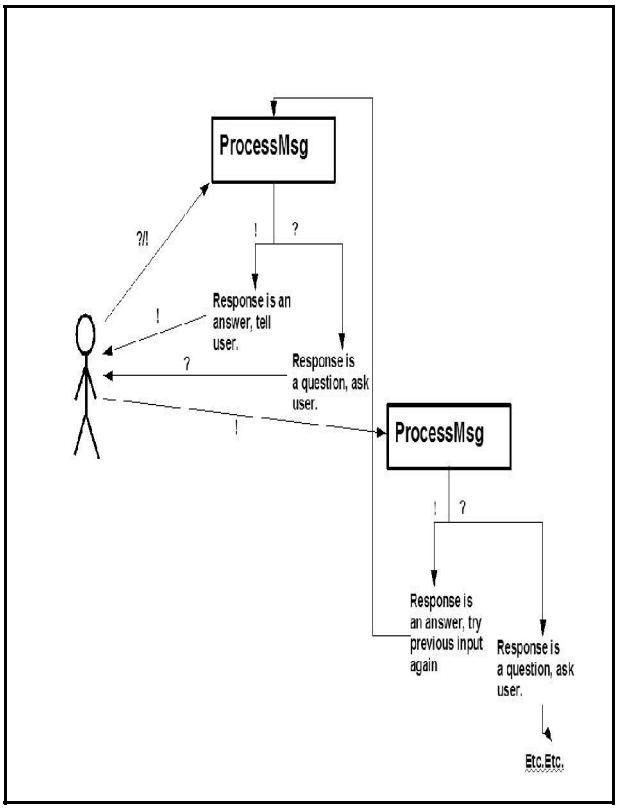

Figure 6: Converse.

The user enters some input, which may be a question or a statement, and this is passed to a sub-process; ProcessMsg (the workings of which are discussed below). Having processed the input, ProcessMsg returns either a statement or a question. If it is a statement, it is passed to the user and the conversation ends. If it is a question, it is also passed back to the user, but in this case the system places the original input on a stack and waits for a reply from the user. When the system receives a reply it is passed to a new invocation of ProcessMsg. This second invocation of ProcessMsg can also return either a statement or a question. If a question is returned, the (second) input is stacked, the question is sent to the user, etc on down through another level. If a statement is returned, this too is sent to the user and the system pulls the top message from the stack and passes it to ProcessMsg where it is processed again from scratch.

In this way the system conducts a conversation with the user, successively seeking clarification to anything that it does not know how to handle.

The sub-process ProcessMsg is responsible for taking one input at a time, converting it into a set of calls to functions from the Basic Function Library and executing these functions. This is done in the following steps:

- The words are split up and associated with all known part-of-speech tags from the RDB. For example, the input:

  "rip is a protein"

  gives the combinations of patterns:

  (rip,noun),(is,verb),(protein,noun)

  (rip,verb),(is,verb),(protein,noun)

- Each pattern is then matched, word by word, to a set of translation rules from the RDB. Each translation rule associates a pattern of words to one or more basic functions. E.g:

  (%sub%,noun),(is,verb),(%obj%,noun) = assert(isa(%sub%,%obj%))

  The left side of the rule specifies the word patterns to which it matches (note the use of named variables, %sub% and %obj%) and the right side specifies the corresponding basic functions.

  It is possible to have several patterns for an input, and for each of these to match to several translation rules. Hence an algorithm is applied to all the possible interpretations of the input to decide which is the most appropriate. For example, the second pattern would not match this rule. In fact it is unlikely that a rule for a verb/verb/noun pattern would exist, so this interpretation would be discarded.

After matching, the actual values from the pattern are substituted for the appropriate variables in the right side of the rule, giving a prolog statement:

assert(isa(rip,protein))

Note that for some patterns it is possible that a series of more than one rule may apply, i.e. rule1 matches the first 4 words of the pattern and rule2 matches the next 5 and so on.

- The Prolog clauses are then passed to the Prolog engine for processing. In this example it adds one fact to the LDB.

Note that the system is not attempting to understand the input. It is merely determining the most likely association between the input and a set of basic functions based on the rules it has in its database. The system does not impose a specific syntax on the user. It is also possible for the system to accept other forms of input, for example, structured text such as XML or HTML, by adding appropriate translation rules to the database.

4. Future Work

The essence of the converstaional engine has been presented. The following are yet to be investigated fully:

- Scaling: It is not yet clear how well the Prolog engine will cope with a significant increase in the size of the LDB.
- Scoring algorithm: As more and more translation rules are added to the database the scoring algorithm may need to be changed in order to adequately differentiate between them.
- Automatic generation of translation rules: For the system to 'grow and learn' it will be necessary for ProcessMsg to be able to add appropriate translation rules to the database.
- Fault tolerance: Improve the mechanism for handling failure within a basic function.
- Bulk load part-of-speech tables and LDB: To give the system a head start it is intended to bulk-load several parts of the database from various online dictionaries and ontologies.
- Build primitive function library: What the system is able to do is limited mainly by the contents of the basic function library. It will be necessary to write an extensive set of routines.
- Make the system available to a number of biologists to assess its usefulness.

5. Conclusion

The outline of a system for the integration and querying of data sources in the bioinformatic domain has been given. It has a simple, intuitive user interface, and is capable of being extended to cope with new data sources. A mechanism for the composition of high level modules from base-level units using a conversational process has also been presented. The full system is still under development. It already performs reasonably well on simple tasks and statements like learning facts from conversation to add to the LDB where appropriate, being able to read data from a (relational) data source and integrate it into the LDB, and to answer simple user queries on that data. Further development is required to demonstrate the system's full potential and to investigate the effects of increases of scale.

6. References

[1] NCBI, "GenBank Growth", http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html, 2005

[2] NCBI, "Entrez PubMed", http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed, 2005

[3] CBIL and others, "The GUS Platform", http://www.gusdb.org, 2005

[4] Sanger-Institute, "The Sanger Institute: AceDB Database", http://www.acedb.org, 2005

[5] GMOD, "Generic Model Organism Database Construction Set", http://www.gmod.org/, 2005

[6] A. Rowe, D. Kalaitzopoulos, M. Osmond, M. Ghanem, and Y. Guo, "The discovery net system for high throughput bioinformatics," Bioinformatics, vol. 19, pp. 225i-231, 2003.

A Semantic Web Service-based Architecture for the Interoperability of E-government Services

Alessio Gugliotta
University of Udine
Department of Computer Science
via delle Scienze 206, 33100 Udine, Italy,
gugliott@dimi.uniud.it

Liliana Cabral
The Open University
Knowledge Media Institute
Walton Hall, Milton Keynes, MK7 6AA, UK
l.s.cabral@open.ac.uk

John Domingue
The Open University
Knowledge Media Institute
Walton Hall, Milton Keynes, MK7 6AA, UK
j.b.domingue@open.ac.uk

Vito Roberto
University of Udine
Department of Computer Science
via delle Scienze 206, 33100 Udine, Italy,
roberto@dimi.uniud.it

Mary Rowlatt
Essex County Council
Strategic Information Manager
PO Box 11, County Hall Chelmsford Essex, CM1 1LX
maryr@essexcc.gov.uk

Rob Davies
MDR Partners
11 Sanders Drive Colchester, Essex, UK CO3 3SE
rob.davies@mdrpartners.com

Abstract

We propose a semantically-enhanced architecture to address the issues of interoperability and service integration in e-government web information systems. An architecture for a life event portal based on Semantic Web Services (SWS) is described. The architecture includes loosely-coupled modules organized in three distinct layers: User Interaction, Middleware and Web Services. The Middleware provides the semantic infrastructure for ontologies and SWS. In particular a conceptual model for integrating domain knowledge (Life Event Ontology), application knowledge (E-government Ontology) and service description (Service Ontology) is defined. The model has been applied to a use case scenario in e-government and the results of a system prototype have been reported to demonstrate some relevant features of the proposed approach.

1. Introduction

The current trend in e-government applications calls for joined-up services that are effective, simple to use, shaped around and responding to the needs of the citizen, and not merely arranged for the provider's convenience. In this way, the users need have no knowledge of – nor direct interaction with – the government entities involved. Thus, services need to be interoperable in order to allow for data and information to be exchanged and processed seamlessly across government.

Interoperability is a key issue in the development of current e-government services. A recent working paper by the Commission of European Communities [14] emphasized its role, not only as a technical issue concerned with linking up computer networks, but also as a fundamental requirement to share and re-use knowledge between networks, and reorganize administrative processes to better support the services themselves.

Still in ref.[14], three levels of interoperability were in-

dividuated: *technical*, *semantic* and *organizational*. The first one refers to the topics of connecting systems, defining standard protocols and data formats. The second one concerns the exchange of information in an understandable way, whether within and between administrations, either locally or across countries and with the enterprise sector. The third one refers to enabling processes to co-operate, by rewriting rules for how Public Administrations (PAs) work internally, interact with their customers, use Information and Communication Technologies (ICT).

On practical grounds, the *integration* of services is a basic requirement of PA portals, which aim at gathering and transforming processes – needed for a particular citizen's life event – into one single service and the corresponding back-office practices. A promising solution, which we extend in this paper, is offered by the *one-stop government portals* [23], [17], that are unified on-line access points, where various PAs collaborate for the provision of integrated services.

Another technological solution adopted for integration purposes are the Web Services (WS) [10] [3] and Semantic Web Services (SWS) [12], which enable the standardized description, retrieval, invocation and combined use of pre-existing applications.

The present paper addresses the issues of semantic interoperability and service integration, by adopting knowledge management techniques. In particular, ontologies are employed [2],[1] in support of the following activities: systematic and standard description of information resources (documents, processes and their relations); support to the automation of services, systems and infrastructures involving PAs; supply of added-value services, such as selected information retrieval and personalization of contents.

We describe the architecture of a one-stop government portal based on a SWS infrastructure, which we have implemented an experimental testbed. The portal provides common services from government organizations without affecting their autonomy, with flexible solutions to enhance and include additional functionalities. We use the IRS-III [15] framework that supports the creation and management of SWS according to the WSMO [25] ontology.

The project also involves the development of a domain ontology that represents the semantic structure of life events underlying the service supply.

Advantages of the proposed solution are: providing a single access point to government services via the web, providing citizen-oriented services by means of the life event metaphor, providing the tools for collecting information from autonomous Public Administrations (PAs), while keeping their internal processes and legacy systems intact.

The paper is organized as follows: in Section 3 we introduce the system architecture; in Sections 5 and 4 we describe the middleware layer; in the next we present typical system operations and a case study implemented using the architecture. The final section contains our conclusions.

2 Related Work

2.1 Web Information Systems in E-Government

Many e-government projects are being developed and various approaches have been proposed for the design and the development of an architecture to deliver e-government services to citizens.

The *eGOV* project [6] proposes an architecture to enable 'one-stop government'. In order to describe services a markup language (GovML) has been developed [11]. GovML defines a set of metadata to describe public administration services and life events.

The *FASME* project [9] focuses on supporting citizen mobility across European countries by the integration of administrative process. In order to satisfy this objective a smart card is provided to citizens for the storage of all personal information and documents. Services are delivered through dedicated kiosks.

The *EU-PUBLI.com* project [8] defines a Unitary European Network Architecture. It proposes a middleware solution to connect heterogeneous systems of different public administration and to enable a service-based cooperation between public administrations.

The *eGovSM* project [17] supports the automation of administrative process involving several administrations and allowing the reuse of data. The eGovSM is formalized using a set of XML Schema models in order to support the realization of an interoperable system.

Unlike our approach, no one of such projects takes into account the use of SWS technology as the base for developing a government portal nor the use of ontologies for describing life events, services and e-government knowledge.

2.2 Semantic Technologies in E-Government

The e-government scenario is an obvious and promising application field for ontologies, since legislative knowledge is by nature formal to a large extent and its definition is shared by many stakeholders. In fact there are other e-government projects where the semantic technologies are involved.

The *ONTOGOV* project [19] is developing a platform that will facilitate the consistent composition, reconfiguration and evolution of e-government services.

The *e-POWER* project [7] has employed knowledge modeling techniques for inferences like consistency check, harmonisation or consistency enforcement in legislation.

The *SmartGov* project [21] developed a knowledge based platform for assisting public sector employees to generate on-line transaction services.

The *ICTE-PAN* project [13] developed a methodology for modeling PA operations and tools to transform these models into design specification for government portals.

Such projects have demonstrated the feasibility of semantic technologies in e-government, but they did not explore the possibility of using a Semantic Web Services infrastructure for the interoperability and integration of different public administration services.

3 The proposed E-Government Portal Architecture

We define here the basic structure of a generic e-government one-stop portal based on a SWS infrastructure. This architecture extends the one defined in [16], wherein the concept and the architecture of an *active life event portal* were illustrated. The core component of such portal is a knowledge-based system. Which is a program based on inference mechanisms to solve a problem by employing the relevant knowledge. Its primary goals are identifying a life event applicable to the user's requirements, identifying the services needed to solve a given event and matching the user request, and identifying an instance of each service in the list. In our approach, the role of knowledge-based system is played by a semantically-enhanced architecture, which is composed of the loosely-coupled modules outlined in Figure 1.

The modules are organized in three layers.

User Interaction supports the user to identify a life event and collects information for service execution.

Middleware allows the semantic description, publishing and updating of life events in order to provide citizens with an up-to-date and personalized list of available services and allows the description, identification, instantiation and invocation of services.

Service Layer is responsible for the execution of services for a life event. Each PA supplies services through the WS technology. Each one is connected to the back-office and semantically described via the IRS-III module of the Middleware layer.

The core of the architecture is the Middleware, the semantically-enhanced layer responsible for the interoperability and service integration. The main issues addressed in the Middleware layer are follows[12].

Infrastructure for semantic interoperability: enables the automated interpretation and paves a common ground for services.

The ontologies: knowledge models for defining the concepts of the e-government domain and the semantic structure of the life events involved in the service supply.

Both issues will be detailed in the forthcoming sections.

4 The Infrastructure for Semantic Interoperability

We use a Semantic Web Services infrastructure for the semantic interoperability of e-government portal services. Our approach uses IRS-III [15], which is a framework allowing the publication, configuration and execution of multiple, heterogeneous web services, compliant with WSMO [25]. The architecture of IRS-III includes the following components: Server, Publisher and Client. The components communicate through a SOAP-based protocol. Publishing with IRS-III entails associating a specific web service to a WSMO description. IRS-III contains platforms to support the publishing of web services as well as standalone Java and Lisp code. Web applications accessible via HTTP GET methods are handled internally by the IRS-III server. The IRS-III Client supports a goal-centric invocation mechanism. The user simply asks for a goal to be solved; the IRS-III broker locates the appropriate semantic description, and then invokes the deployed service.

The *Web Service Modeling Ontology (WSMO)* [25] is a formal ontology for describing the various aspects of services in order to enable the automation of Web Service discovery, composition, mediation and invocation. Its main components are Ontologies, Goals, Web Services and Mediators. *Goals* represent the objectives that users would like to achieve via the WSs. The WSMO definition of goal describes the state of the desired information space and the desired state of the world after the execution of a given WS. A goal can import existing concepts and relations defined elsewhere, by either extending or simply re-using them as appropriate. *Web Service* descriptions describe the functional behavior of an actual WS. The description also outlines how Web Services communicate (choreography) and how they are composed (orchestration). *Mediators* define mappings between components: for instance, a goal can be related to one or more web services through mediators. They facilitate the clear-cut separation of different interoperability mechanisms. *Ontologies* provide the basic glue for semantic interoperability and are used by the three other components.

5 The Conceptual Model

Both PAs and citizens can benefit from a standard conceptual model for describing public services and life events. PAs will have a shared description structure, thus production and management of government information would be eased, while interoperability with other agencies would be fostered. Ontologies can also be used to represent the viewpoint of citizens in the application, making it easier for them to navigate through different services and administrations.

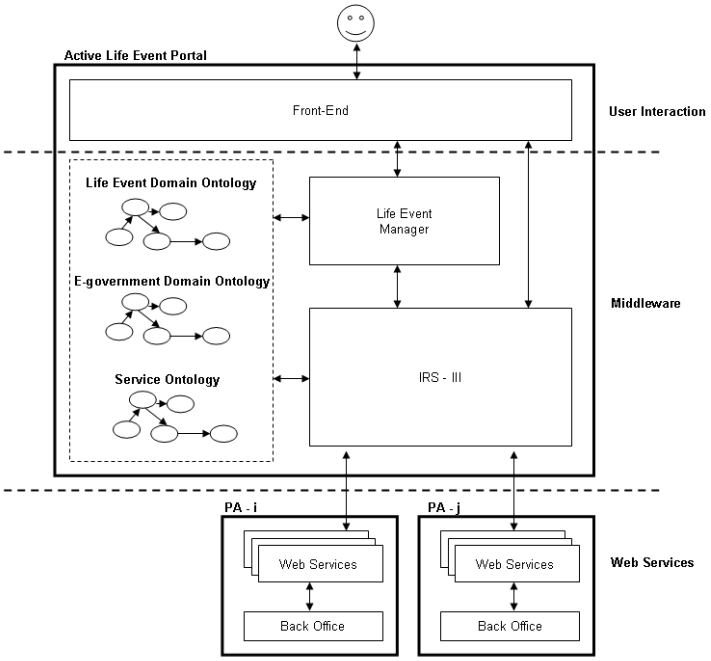

Figure 1. The semantically-enhanced infrastructure of a portal.

Ontologies enable the use of vocabulary about a certain domain in a coherent and consistent manner [20]. In particular, ontologies are the tools for formalizing knowledge and encoding higher-level data models, such as life events, procedures and services.

We use OCML (Operational Conceptual Modeling Language) [18] for describing a conceptual model for the e-government portal based on three ontologies: the E-Government Domain Ontology, the Life Event Domain Ontology, and the Service Ontology.

In the design of the ontologies above, we followed a deductive approach based on existing upper and specialized ontologies, with the assistance of domain experts. In particular we used the Description & Situations (D&S) [5] – a module of the DOLCE ontology [4]. D&S is a theory to describe context elements (non-physical situations, plans, beliefs,...) as entities. It features a philosophically concise axiomatization.

The *E-Government Domain Ontology* encodes concepts in the PA domain: organizational, legal, economic, business, information technology and end-user concepts. Starting from the D&S ontology we have built a domain ontology where all the PA concepts refer to (subclasses of) D&S main concepts. The formal descriptions of the PA-related concepts are the building blocks for the descriptions of the two other ontologies. Part of this ontology is shown in Figure 2.

The *Life Event Domain Ontology* defines a hierarchy of topics. Each life event can branch into sub-life events. It describes a life event in terms of norms that define it, information objects that describe it, parameters, involved agents (actor, applicant and provider), involved objects, involved procedures, and results (effects) of the life event. Moreover, for each Life Event, it is possible to associate one or more Goals, a concept of the WSMO ontology, and Entitlements, which include services and benefits. We show in Figure 3 the UML diagram of the Life Event model. All the classes describing life events – e.g. someone-move-in, getting-married, getting-divorced, moving-house, etc.– are subclasses of the life event class model.

The following ontology (OCML code) defines the *Someone-Move-In* life event and the related *Someone-Move-In-Description*:

```
(def-class Someone-Move-In
            (Manage-Family-Related-Life-Event) ?x
((defined-by :type Someone-Move-In-Description
            :min-cardinality 1)
 (has-associated-entitlement
            :value Child-benefit-someone-gets-for-you
            :value Guardian-Allowance)
 (has-associated-goal
            :value moving-in-person-change-of-domicile
            :value moving-in-person-change-of-residence
            :value family-change-of-circumstance)
 (moving-in-person-change-of-domicile
            :type notify-change-of-domicile)
 (moving-in-person-change-of-residence
            :type notify-change-of-residence)
 (family-change-of-circumstance
            :type notify-change-of-family))
 :constraint (and (defined-by ?x ?d)
```

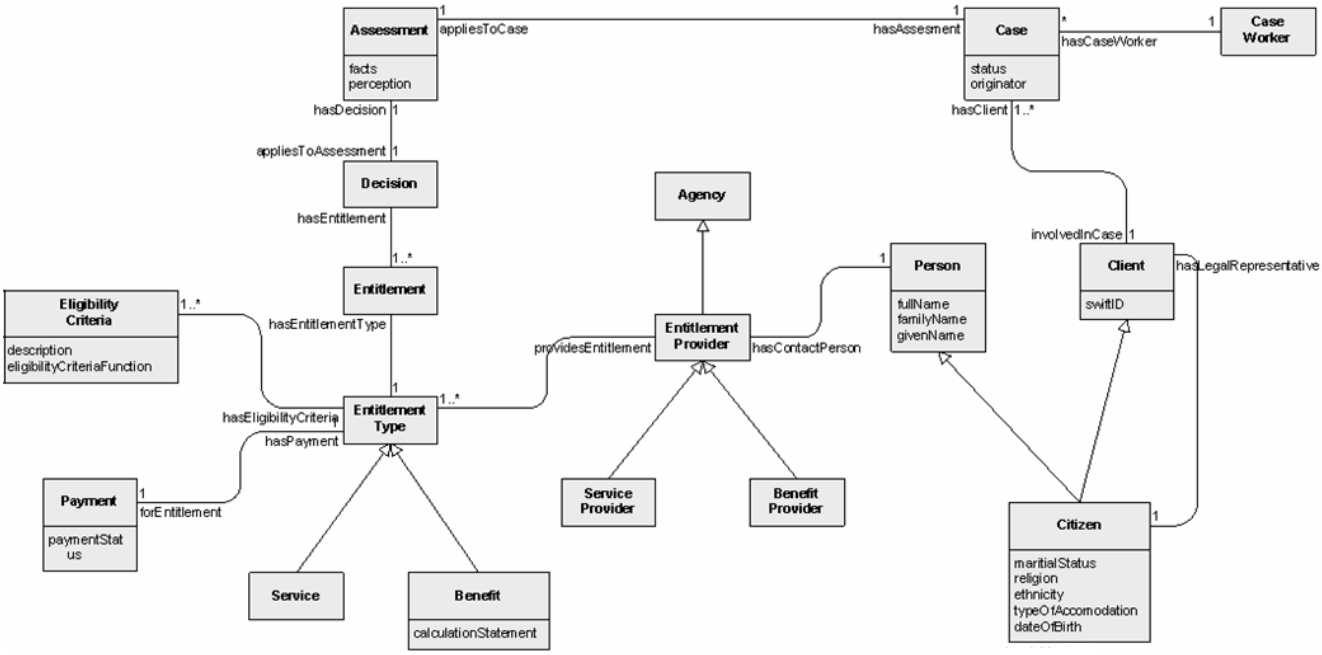

Figure 2. The UML diagram showing a small part of the whole E-Government Domain Ontology, which specifically models the 'Change of Circumstance' case study scenario (Section 6.1).

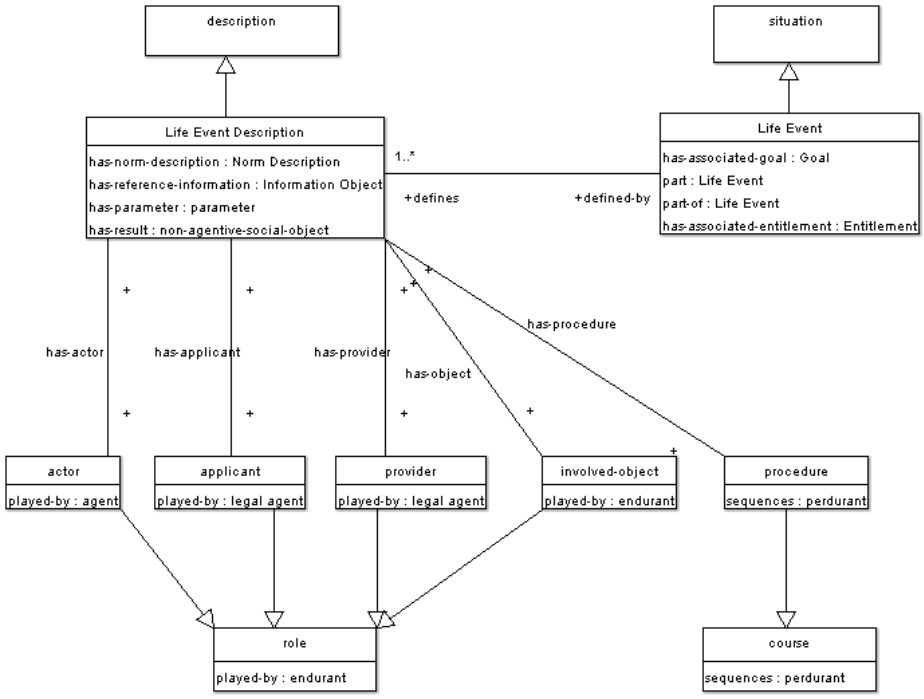

Figure 3. The UML diagram showing the generic description of a life event. A Life Event is a Situation (D&S concept) that satisfies one or more descriptions (different points of view: citizen, provider, PA, . . .). A Life Event Description is a Description (D&S concept). A description is composed by different role and courses (D&S concepts)

```
                (has-moving-in-person ?d ?p)
                (played-by ?p ?c)
                (or (exists ?g

                    (and (moving-in-person-
                          change-of-domicile ?x ?g)
                         (notify-
                          change-of-domicile ?g)
                         (has-citizen ?g ?c)))
                    (exists ?gr
                    (and (moving-in-person-
                          change-of-residence ?x ?gr)
                         (notify-
                          change-of-residence ?gr)
                         (has-citizen ?gr ?c))))))


(def-class Someone-Move-In-Description
           (Life-Event-Description) ?x
((defines :type Someone-Move-In
          :min-cardinality 1
          :max-cardinality 1)
 (has-parameter
          :value has-moving-date)
 (has-moving-date
          :type date-parameter)

 (has-applicant
          :value has-citizen-applicant)
 (has-citizen-applicant
          :type citizen-applicant)

 (has-provider
          :value has-government-provider)
 (has-government-provider
          :type government-provider)

 (has-actor :value has-moving-in-person
            :value has-destination-family-group
            :value has-origin-family-group)
 (has-moving-in-person
            :type moving-in-person)
 (has-destination-family-group
            :type destination-family-group)
 (has-origin-living-unit
            :type origin-living-unit)

 (has-result :value has-modified-living-unit)
 (has-modified-living-unit :type modified-living-unit))
 :constraint (and (exists (?md ?f ?lu)
                   (and (has-modified-living-unit ?x ?md)
                        (played-by ?md ?f)
                        (living-unit ?f)
                        (has-origin-living-unit ?x ?lu)
                        (played-by ?lu ?f)))
                  (exists (?md2 ?f2 ?d)
                   (and (has-modified-
                         living-unit ?x ?md2)
                        (played-by ?md2 ?f2)
                        (family-group ?f2)
                        (has-destination-
                         family-group ?x ?d)
                        (played-by ?d ?f2)))
                  (exists (?p ?c ?o)
                   (and (has-moving-in-person ?x ?p)
                        (played-by ?p ?c)
                        (has-origin-living-unit ?x ?o)
                        (member ?o ?c)))
                  (exists (?a ?c ?fg)
                   (and (has-citizen-applicant ?x ?a)
                        (played-by ?a ?c)
                        (has-destination-
                         family-group ?x ?fg)
                        (member ?fg ?c)))))
```

The *Service ontology* contains the SWS definitions. They correspond to instances of the Goal, Web Service and Mediator classes used in the IRS-III module (Section 4), following the WSMO definitions (Section 4). The following OCML code defines the *notify-change-of-address-goal* and the description of the *county-council-provider-notify-change-of-address* capability:

```
(def-class notify-change-of-address-goal (GOAL) ?goal
  ((has-input-role :value has-new-address
                   :value has-old-address
                   :value has-client-name
                   :value has-client-id
                   :value has-source-provider
                   :value has-target-provider)
   (has-output-role :value has-confirmation)
   (has-new-address :type string)
   (has-old-address :type string)
   (has-client-name :type string)
   (has-client-id :type integer)
   (has-source-provider :type service-provider)
   (has-target-provider :type service-provider)
   (has-confirmation :type string)))

(def-class county-council-provider-
           notify-change-of-address-ws-capability
  (capability) ?capability
  ((used-mediator
          :value notify-change-of-address-mediator
    has-assumption
          :value
    (kappa (?psm)
     (and (unit-of-organization
             (role-value ?psm 'has-target-provider)
             ?agency)
          (county-council-organization ?agency)))))
```

6 E-Government Portal Implementation

By using the infrastructure described previously, the application (portal) developer will use tools for describing, publishing and invoking services. Figure 4 shows some snapshots for the prototype scenario explained in next section.

Managing Services A developer creates a new WS for supplying a service through the portal. He provides a Goal description which represents the objectives that citizens would like to achieve via WS – and associates it to a Life Event. The developer might also refer to an already existing Goal instead of defining a new one. Then, the developer semantically describes its WS and associates it to the Goal. Dedicated interfaces and the IRS-III module are used for describing Goals and Web Services. Descriptions are maintained in the Service Ontology. Finally, through the publisher interface of the IRS-III module, the developer publishes the SWS, associating the semantic description to the developed WS.

Consistency between the Middleware layer and the Web Services is maintained by means of the Service Ontology. If changes are made to a web service or if a web service is removed/replaced, the developer updates only the correspondent SWS description. Goals, Mediators and other parts of the system (domain ontologies, infrastructure) are not affected.

Invoking a Goal A request presented by the user through the portal interface is satisfied by a goal achievement. The request is processed by the Life Event Manager module, which discovers all related life events, allowing the user to select the appropriate Life Event (e.g. Notify change of address). Information is described through the E-Government Domain Ontology, while the Goals are described via the Service Ontology. When the user invokes one of the goals, the Life Event Manager calls the IRS-III module, which retrieves the semantic description of the goal. Then, it creates an instance with specific data items, and identifies and invokes the web services addressing the user needs by means of their semantic description. Finally, the web service is executed by the PA information system and the result is presented to the user.

6.1 Prototype Scenario: Change of Circumstance

We illustrate the implementation of our e-government portal through an application scenario.

The prototype is a portal for the Essex County Council based on the infrastructure reported in Section 3. In this scenario the end users are the caseworkers of the Community Care department which are helping the citizen to report his/her change of circumstance to the different agencies involved in that process. In this way, the citizen only has to inform the county once about his/her change, and the government agency (Community Care unit) automatically notifies all the agencies involved. An example Community Care service scenario might be when a disabled Mother Moves In to her daughter's home. The change of circumstance provokes a change in which services and benefits – health, housing, etc. – the citizens are eligible to receive. Multiple service-providing agencies need to be informed and interact.

The main objective is that a citizen should only notify his/her change of circumstance to one single local authority. Then, all changes (Post Office, Treasury, National Health Service, etc.) will be automatically notified.

For instance, the mother notifies a case worker at Community Care department that she is moving. The case workers have a coordination role, which are frequently centred on tracking changes of the living address of the client.

We have developed the E-government Domain Ontology for describing the main concepts related to the change of circumstance scenario (Figure 2). The concepts describe the process of defining a *Case* for a particular *Client*. The *Case Worker* does an *Assessment* about the *Citizen* situation and takes a *Decision* about the *Benefits* and *Services* the Client is entitled to. Every Entitlement Type has specific Eligibility Criteria, described by a function.

The portal is associated with the Life Event Domain Ontology that can represent events related to the E-government Domain Ontology for Change of Circumstance. These include getting married, going into hospital, someone move in, going into residential/nursing home, inheriting money, winning a lottery, retiring, and death.

In addition, the Life Event Domain Ontology associates events with Semantic Web Services. In particular we refer to the *Someone move in* life event and its associated goal *Change of address*.

The prototype portal administers a network of agencies – service/benefit providers – that can register declaring which services/benefits they supply. Every registered agency publishes one or more SWS, which have to be based on the agreed E-Government Domain Ontology. There are a number of fixed SWS Goals (e.g. change of address) to which agencies could subscribe for publishing services.

Agencies can also define and make their own SWS Goals available through the portal. For instance, the change of address goal is defined by the Community Care department, but different agencies can create their own SWS for managing the change of address on their systems.

The case worker can register a new client, search or update the information of an existing one through the portal. He has to fill in several fields about the citizen's information. This information will be stored and related in the E-Government domain ontology, as a new instance of the class *Client*.

The same procedure is followed to register a new agency. This time it is not the case worker, but the agency, the one that registers itself. It also has to fill in the name of the service/benefit it provides (in the form of SWS), accompanied with the URL of the server where the SWS is published. This information will be stored and related in the E-Government domain ontology, as a new instance of the class *Agency*.

In Figure 5, we present the user interface for invoking the change of address goal. The case worker chooses the agency he wants to notify, he can also choose the option for automatically detecting the agency to notify. He then inserts the data of the client and activates the 'notify' button. With this simple form the case worker shares change of address details with relevant partner organizations (Housing, Pension Service, etc.) and providers of external commissioned services (e.g. meals on wheels and nursing support).

When doing this, the change of address goal is invoked and the IRS Server detects and calls the web services that match the data. A matching web service could be composed of different integrated web services that realize the change of address (updating different databases in different PAs). In our example, the Vulnerable Citizen Change of Address WS is detected (the web service published by the Community Care department). The client's address is updated in the Community Care legacy database and the user receives a confirmation on what happened. Figure 6 shows what happens in the IRS-module when the 'notify' button is pressed.

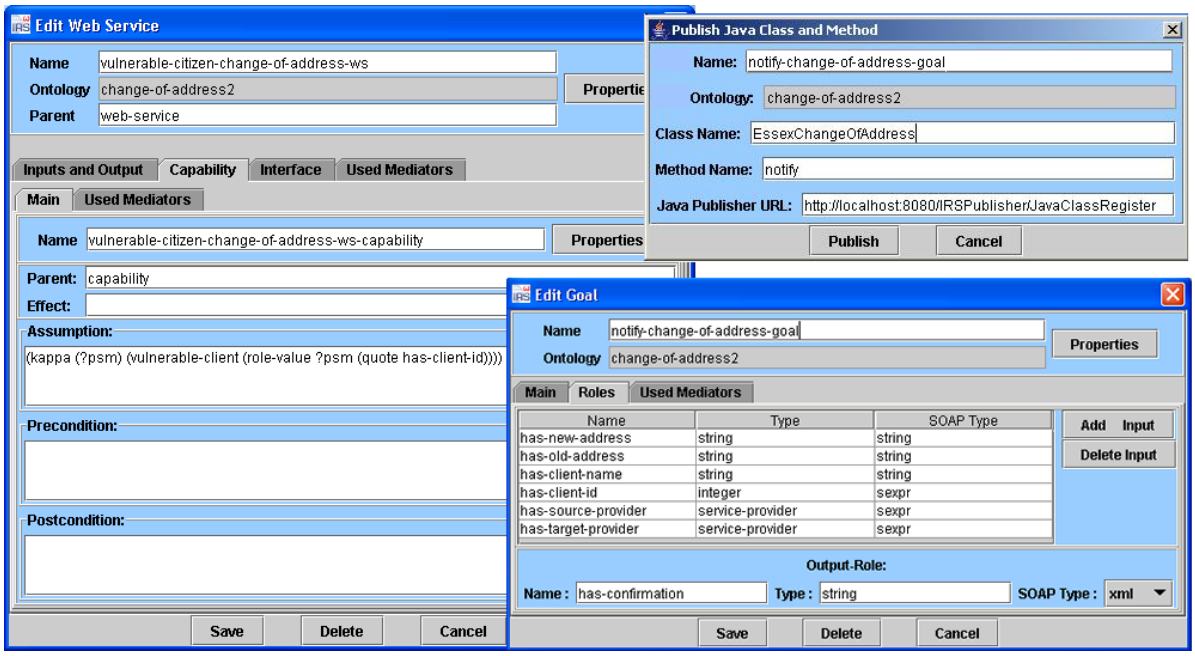

Figure 4. User interfaces for defining Goal and Web Service descriptions according to the WSMO ontology and the publisher interface for publishing SWS.

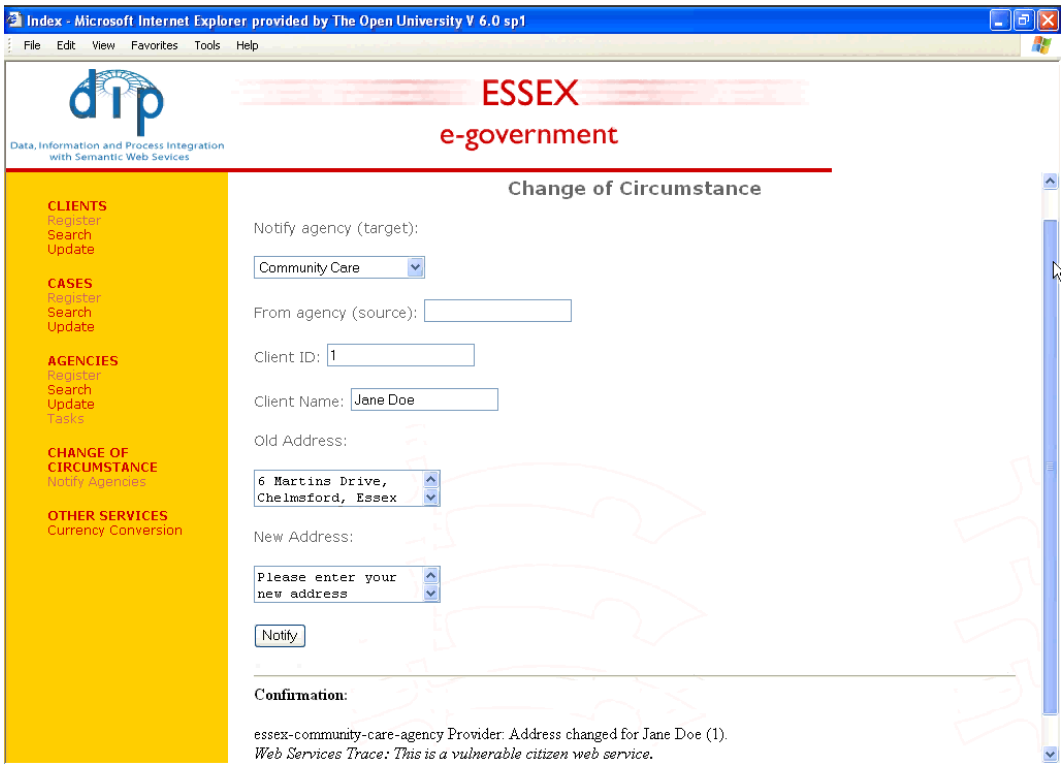

Figure 5. Web Page to invoke *the change of address* **goal**

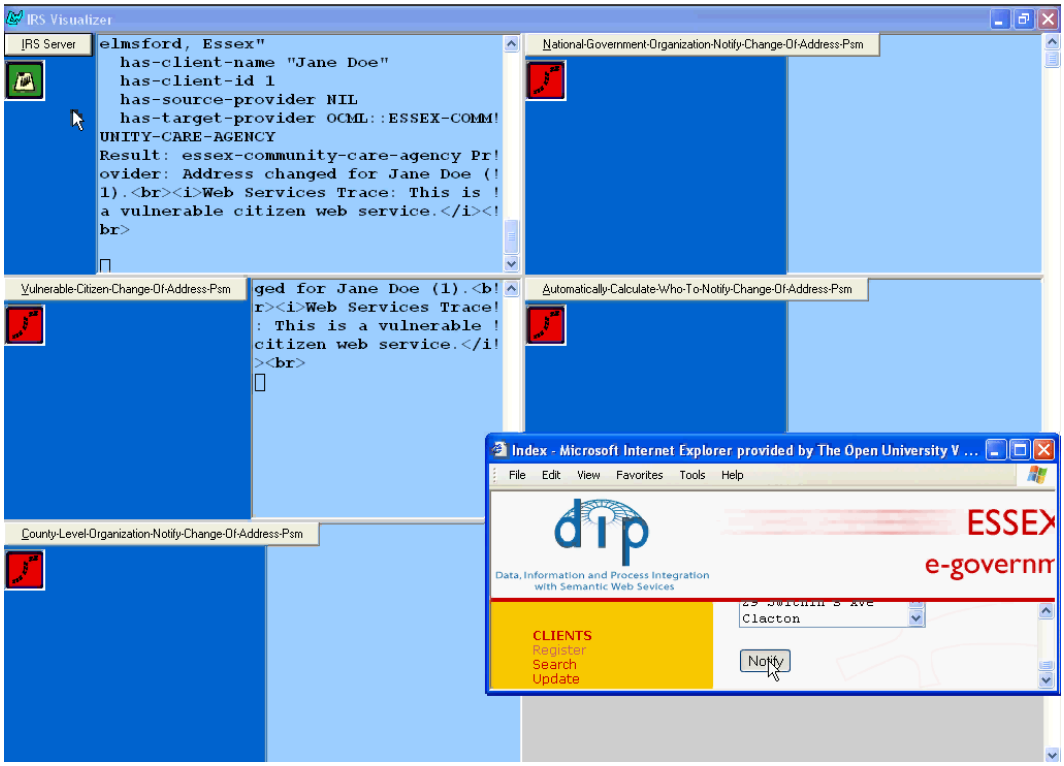

Figure 6. The IRS visualizer interface. It shows which web services are activated, among all published web services. Each box in the IRS visualizer represents a published web service. When a published web services is activated, its behavior (inputs, output, etc.) is traced in the respective box.

7 Conclusions and future work

The aim of our research effort is developing a semantic based architecture of a portal, that helps the user – citizen and business – to find the information and services that best fit his/her needs, and enables the interoperability between government agencies and service providers, as well as agencies to integrate existing service for creating new ones.

The proposed architecture is composed of a front-end, a middleware and a service layer; we focused on the second layer which defines an explicit conceptual model in terms of three domain ontologies: the E-Government, the Life Event and the Service Ontology, each of which is grounded on the upper ontology D&S, and an infrastructure for interoperability and integration in terms of Semantic Web Services, based on the IRS-III framework.

Our architecture applies semantic web technology at the data and service level.

A prototype of a portal realizing the proposed architecture has been implemented with a scenario about the 'Change of Circumstance' of citizens for illustrating the advantages of the proposed architecture. The difference between our prototype and the 'one stop portal' [23] is that end users are not citizens, but the main aim was to test the advantages of SWS in term of interoperability between different PAs and integration of services.

Future work includes the extension of the ontologies for capturing more concepts about the e-government domain and life events. Further life event and services descriptions will be integrated into the portal and a real one stop portal will be developed.

Acknowledgement

This work is supported by the Data, Information and Process Integration with Semantic Web Services (DIP) – http://dip.semanticweb.org.

References

[1] Apostolou D., Stojanovic L., Lobo T.P., Thoenssen B.: Towards a Semantically-Driven Software Engineering Environment for eGovernment. M. Bohlen et al. (Eds.): TCGOV 2005, LNAI 3416, pp.157-168, 2005.

[2] Bercic B., Vintar M.: Ontologies, Web Services, and Intelligent Agents: Ideas for Further Development of Life-Events Portals. R. Traunmuller (Ed.): EGOV 2003, LNCS 2739, pp. 329-334, 2003.

[3] Bouguettaya A., Medjahed B., Rezgui A., Ouzzani M., Liu X., Yu Q.: WebDG - A Platform for E-Government Web Services. S. Wang et al. (Eds.): ER Workshops 2004, LNCS 3289, pp. 553-565, 2004.

[4] Oltramari A., Gangemi A., Guarino N., Masolo C.: Sweeting ontologies with DOLCE. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 2002.

[5] Gangemi A., Mika P.: Understanding the semantic web through description and situations. In DOA/CoopIS/ODBASE 2003 Confederated International Conference DOA, CoopIS and ODBASE, Proceedings, LNCS. Springer., 2003.

[6] The eGOV Project Website: http://www.egov-project.org

[7] Van Engers T., Patries J.M., Kordelaar J., Den Hartog J., Glasseee E. (2002). Available at http://lri.jur.uva.nl/epower/

[8] The EU-PUBLI.con Project Website: http://www.eu-publi.com

[9] The FASME Project Website: http://www.fasme.org/index-org.html

[10] Gamper J., Augsten N.: The Role of Web Services in Digital Government. R. Traunmuller (Ed.): EGOV 2003, LNCS 2739, pp. 161-166, 2003.

[11] Kavadias G., Tambouris E.: GovML: A Markup Language for Describing Public Services and Life Events. Working Conference on Knowledge Management in Electronic Government, 2003.

[12] Klischewski R.: Semantic Web for E-Government. R. Traunmuller (Ed.): EGOV 2003, LNCS 2739, pp. 288-295, 2003.

[13] ICTE-PAN Project Website: http://www.eurodyn.com/icte-pan

[14] Commission of the European Communities, 'Linking up Europe: the Importance of Interoperability for e-Government Services', Commission Staff Working Paper SEC(2003) 801, July 3, 2003.

[15] Domingue J., Cabral L., Hakimpour F., Sell D., Motta E.: IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. Proc. of the Workshop on WSMO implementation, Frankfurt, Germany, September 2004.

[16] Leben A., Bohanec M.: Architecture of an Active Life-Event Portal: A Knowledge-Based Approach. 5th IFIP International Conference, KMGov2004, Krems, Austria, May 17-19, 2004. Proceedings.

[17] Mugellini E., Pettenati M.C., Khaled O.A., Pirri F.: eGovernment Service Marketplace: Architecture and Implementation. M. Bohlen et al. (Eds.): TCGOV 2005, LNAI 3416, pp.193-204, 2005.

[18] Motta E.: Reusable Components for Knowledge Modelling. IOS Press, Amsterdam, The Netherlands, 1999.

[19] OntoGov Project Website: http://www.ontogov.com

[20] Gomez-Perez A., Fernandez-Lopez M., Corcho O.: Ontological Engineering: with examples from the areas of Knowledge Management, e-commerce and the Semantic Web. Springer-Verlag, 2004.

[21] SmartGov Project Website: http://www.smartgov-project.org

[22] SOAP Version 1.2 Part 0: Primer. http://www.w3.org/TR/soap12-part0/

[23] Wimmer M.A.: European Development towards Online One-stop Government: The 'eGOV' Project. ICEC2001 Conference, 31/10 - 4/11/2001, Vienna. Proceedings.

[24] Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[25] Web Service Modeling Ontology – Standard. http://www.wsmo.org/2004/d2/

The xMem Project: Semantic Memory of Web Interactions

Stefano Ceri, Florian Daniel, Maristella Matera, Francesca Rizzo
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Via Ponzio 35/a, 20133 Milano, Italy
{ceri,daniel,matera,rizzo}@elet.polimi.it

Abstract

Finding a previously visited page during a Web navigation is a very common and important kind of interaction. Although most commercial browsers incorporate history mechanisms, such mechanisms are typically very simple indexes of visited pages, sorted according to the time dimension. They are not very effective and are quite far from giving users the impression of a semantically aware, long-term memory, as it is available to the human brain. In particular they lack associative, semantic-based mechanisms that are essential for supporting information retrieval.

This paper introduces the xMem (Extended Memory Navigation) as a new method to access users' navigational history, based upon semantic-based and associative accesses. Its aim is to imitate some of the features of the human memory, so as to give users a better understanding of the context of their searches, by exploiting time-based ordering and semantic clues.

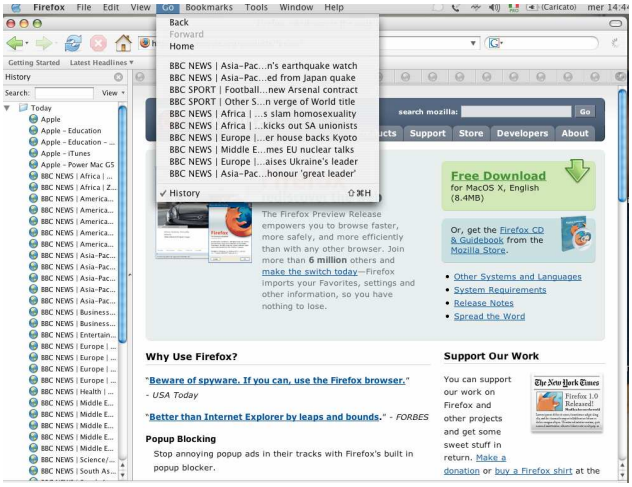

Figure 1. FireFox history mechanism. The "go" menu displays the last 10 visited URLs in a stack from the most recent to the oldest. FireFox also offers a history window that organizes information in chunks per days.

1. Introduction

As the amount of information on the World Wide Web continues to grow, efficient hypertext navigation mechanisms are becoming crucial. Among them, special attention must be devoted to effective history mechanisms enabling users to get back to information already met [18].

History tools are common components of Web site interfaces. They aim to support users to go back to the previously used information. Three factors motivate the introduction of these mechanisms in commercial browsers: (1) they can help users to navigate through the huge quantity of information provided by the Web, thus providing access to information previously visited; (2) they can substitute search engines for finding old pages and avoid the replication of navigations along intermediate pages to the destination one; (3) they can positively affect users' activities by reducing cognitive and physical navigation burdens: pages can be retrieved with little effort, and users can easily locate where they have been in the past.

Virtually, all commercial and experimental browsers provide users with methods to revisit pages of interest. We categorize them as *passive* and *active*. Passive history mechanisms (see Figure 1) are syntactic in nature, resulting from the navigation actions taken by the user. Active re-visitation mechanisms, such as bookmarks or the "back" button, have a more semantic quality and are explicitly created by users based on their interest level in the page [16].

Most of the results coming from the field of HCI [18, 12, 6, 13] show that more of the 30% of users' activities

on the Web are based on the use of the "back" button or of favorites, but they also show that reverse browsing mechanisms are time consuming and cognitively difficult to use, organize and envision. These reasons have lead us to introduce a new method to access navigational history.

This paper introduces the xMem (Extended Memory Navigation) project, which aims at providing advanced memory mechanisms to Web users, intermixing semantic aspects with the temporal dimension. The xMem solution requires the classification of information being shown on the Web according to predefined semantic categories, which are next used for indexing and retrieving. Such classification and indexing are effective for those Web applications that are previously described to xMem, and in particular those modeled at a high level of abstraction, by means of primitives of a conceptual model. However, the xMem architecture presented in this paper is quite general and applies as well to arbitrary Web pages, provided that these pages being browsed by users are also analyzed in parallel by the xMem tool by means of suitable wrappers.

The paper is structured as follows. Section 2 describes the main idea underlying xMem and outlines the goals of our research. Section 3 then introduces the functional architecture of xMem, while Section 4 discusses some issues concerning design and implementation of the current prototype tool. Section 5 provides insight into related work and describes already collected experiences within the context of history mechanisms in the HCI field. Finally Section 6 presents some conclusions and gives an outlook over ongoing and future work.

2. The *xMem* Web Interaction Memory

The xMem project offers to users a specialized Web site hosting a repository containing the individual user's memory. The key idea of xMem is the "transparent" transmission to the memory Web site of the URLs of the pages being browsed by the users. Then, suitable rules extract the information from the URLs and populate the memory with semantic content. In the light of the previous considerations, the aim of our work becomes twofold:

(i) Providing easier and more intuitive navigation-history retrieving mechanisms. Web navigation can be aided by novel navigation history tools providing enriched memory access mechanisms that build on semantics-based search criterions categorizing available pieces of information. The xMem approach further puts special focus on accessing the long-term memory and thus fosters the active use of history data.

(ii) Fostering ubiquitous accessibility of the navigation history by making it accessible over the Internet. The gathered and classified information must be always reachable in order to become an active help facility that can be integrated within the user's browsing environment.

In order to provide users with some added value (with respect to usual history mechanisms), besides chronologically ordered lists of URLs, xMem also supports further semantics in presenting history data. So-called semantic classes or categories are associated to groups of URLs in order to provide high-level meanings for the contents of the related Web sites. For associating URLs and categories, proper filters can be defined, which allow specifying syntactic rules describing relationships between specific URL structures and categories.

xMem also provides a mechanism for automatically deriving semantic classes starting from application specifications expressed through WebML conceptual models, and a set of generic filters that apply to any WebML application. WebML is a conceptual modeling language that provides a set of primitives for the design of information content and hypertext front-end of Web applications [17, 7].

Rules are particularly powerful when the pages being browsed by the user have a known design, because the content can simply be derived from the URL; when instead the design is not known, pages can be re-materialized and then analyzed by suitable rules, which are capable of detecting the main concepts being displayed by the pages themselves. In any case, xMem maintains remote log data (with respect to users and 3rd party Web servers) about navigation actions of users, by tracking the requested URLs. We adopt an online logging mechanism that allows accessing history data, classifying navigated contents and calculating related statistics at runtime.

3. *xMem* Architecture

The software architecture of the xMem tool prototype is primarily influenced by two goals of our approach: (i) adopting remote logging mechanisms for (ii) providing online access to logged data. Remote logging builds on the client-server paradigm, online log access suggests splitting the overall tool into two logical components, one for each communication direction. Figure 2 graphically depicts the resulting functional architecture, roughly divided into *Client PC*, *3rd party Web applications* and *xMem Server*.

Users navigate the Internet by means of a common Web browser by continuously requesting pages of Web sites or applications residing on 3rd party Web servers. In order to take advantage of the semantic navigation history facilities of xMem, users must register themselves as new xMem users and install the so-called xMem *Tracker Client* on their client PCs. This component is in charge of monitoring user navigation actions and communicating them to the xMem *Tracker Server*, which is responsible for feeding the incoming messages into the *URL Repository*. The *Classifier* component parses syntactic filters with respect to logged page

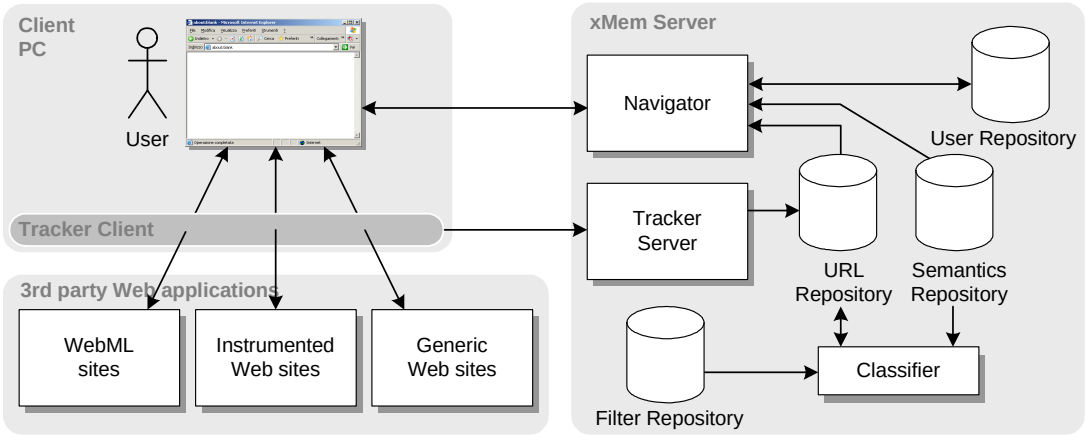

Figure 2. xMem functional Architecture.

requests and associates them to semantic classes stored in the *Semantics Repository*. By means of the so-called *xMem Navigator* Web application, users can then access their personalized navigation history over the Internet and interactively browse logged data at a comprehensible level of abstraction.

The box 3rd party Web applications is divided into *Model-based Web sites*, *Instrumented Web sites* and *Generic Web sites* for representing the different "semantic expressiveness" they enclose with respect to the xMem infrastructure. We will deepen this aspect later on.

3.1. Data Repositories

The xMem project consists of several components that share the same data source. The implementation of the correspondent database depends on the expected load at runtime and can consist either in a single database on the xMem server itself or in a freely distributed server architecture. However, at a conceptual level, we distinguish mainly four logical "repositories", each dedicated to collect functionally similar data:

- **User Repository**: gathers user profiles and preferences for registered xMem users and divide users into different user groups for managing access rights. These data are the basis for personalizing contents and history data and allow fulfilling different functional requirements with respect to user groups. The User Repository must implement at least the following user groups: *xMem User*, *Site Developer* and *Administrator*. xMem users are those that use the history tool, developers are those providing xMem support for their Web sites and administrators manage the overall xMem tool.

- **URL Repository**: contains the actual log data for each registered user; visited Web pages are stored as URL strings. The xMem Tracker accesses the URL Repository for adding HTTP requests, but also Navigator and Classifier make active use of it. The URL Repository masters the highest workload.

- **Semantics Repository**: collects the so-called semantic classes, which provide high-level meanings for classifying the contents of Web pages. Such classes can be defined either automatically or manually by 3rd party site designers. Each adhering Web application is provided with its own set of classes. For WebML applications, a subset of the application's schema is stored within the Semantics Repository for deriving semantically relevant concepts.

- **Filter repository**: contains the set of filters that allow matching URLs and semantic classes by means of syntactic similarities. Each application has its own set of filters. A default set of filters applying to WebML applications is provided by xMem, for all other applications filters can be specified manually or automatically, as explained in section 3.6.

3.2. The xMem Tracker

The history tracking mechanism of xMem comprises two components, one residing on the *xMem Server*, the other running (as background process) on the client PC. At each navigation action, the *Tracker Client* automatically sends a message to its server counterpart communicating xMem user identifier, and requested HTTP URL. The *Tracker Server* inserts the request into the underlying URL Repository, which terminates the tracking process for that URL.

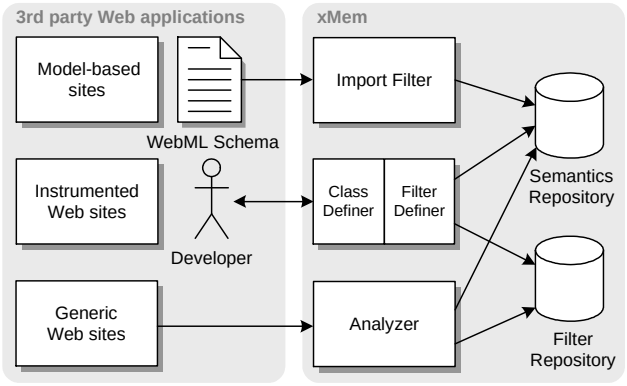

Figure 3. Defining semantic classes for Web sites.

3.3. The xMem Navigator

The interactive component of xMem, the *Navigator*, is a Web application developed through WebML; as such, it is accessible over the Internet, and can be deployed on arbitrary Web servers. Hence, for accessing their history data, users do not need to get used to new interaction paradigms.

The Navigator provides the interface toward history data. It allows users to choose between several mechanisms for accessing navigated pages. Besides chronological and alphabetical lists of URLs, it also shows a hierarchical list of the concepts (semantic classes) that summarize the contents visited by a user. The finest granularity is given by the tracked URLs themselves and allows users to comfortably recall the actual page searched. Also, the Navigator allows users to modify their user profiles.

3.4. The Classifier

One of the core components of xMem consists in the *Classifier*. Its inputs are the tracked URLs and the set of filters of the respective Web application. The Classifier runs those filters over the tracked URLs for constructing semantic associations between *URL Repository* and *Semantics Repository*. It therefore acts as an interpreter of syntactic filters, according to their definition in section 3.6.

3.5. Defining Semantics for Web Pages

For providing users with semantic clues about their past navigation actions, xMem makes use of collection of semantic classes contained in the *Semantics Repository* and describing the navigated pages by means of a natural language, easily comprehensible by users. In this section we show how semantic classes are specified in xMem during subscription of adhering applications.

Subscription in this context means providing, through an appropriate user interface for Web designers, the description of core pieces of information about the application to be added: a unique base URL identifying the application, a set of semantic classes in an automated or manual way, and the qualifying filters for each class. Designers need to subscribe their Web sites in order to gather full xMem support.

For the purpose of defining semantic classes, we distinguish several kinds of Web site implementations. Web applications can be developed using conceptual models, such as WebML, and even automatically generated [19], or they may be freely hand-coded without any particular design method. The former are characterized by a certain degree of already intrinsically modeled high-level semantics, whereas the latter do not provide any additional information besides plain HTML code. Thus, according to the availability of additional semantics and the way this can be fed into xMem, we distinguish three families of Web sites (see Figure 3):

- **Model-based sites**: are those designed by means of WebML and implemented according to WebML design rules. Intrinsically, the WebML design primitives already contain additional information with respect to plain HTML (e.g. the name of content units within pages, such as *Book Details* or *Author*) that can be fed into the Semantics Repository and thus can be interpreted as semantic classes. In addition to the semantics provided by the schemas, WebML applications are characterized by coherent URL formats throughout the whole application that easily can be translated into syntactic filters, which can be generalized and applied to any WebML application. Starting from WebML schemas, it is thus possible to provide a basic set of universal filters (permanently stored within the *Filter Repository*) and to automatically extract semantic classes associated to them. In Figure 3 this process is expressed by means of the *Import Filter* block, which takes in input a WebML schema and provides in output the set of semantic classes describing the site's contents.

- **Instrumented Web sites**: are those sites that do not rely on WebML schemas or other modeling languages, but whose developers adhere to the xMem project anyway. For this purpose, xMem provides the *Class Definer* and *Filter Definer* tools, which allow providing the additionally needed semantics manually by means of proper semantic classes and specifying a set of filters matching the peculiarities of the Web site's URL encoding conventions. Depending on the conventions adopted within the site, filters may differ in their complexity. Hence, the full features of xMem can also be exploited by any Web site whose designer adheres to xMem.

Target	Type	Condition
Schema	string	contains, doesn't contain
User-Info	string	contains, doesn't contain
Host	string	contains, doesn't contain
Port	int	equals, differs from
Path	string	contains, doesn't contain
Query	string	num. parameters, exists parameter
Fragment	string	exists, contains, doesn't contain
Parameter	string, float*	contains, doesn't contain, equals, differs from, greater than, less than

* Depending on the adopted condition expression, parameter types are assumed being either string or float.

Table 1. Target components and available conditions.

- **Generic Web sites**: are those applications that do not know about the xMem project and thus neither provide WebML schemas nor other forms of additional information that could help classifying recorded navigations. Tracked URLs of this category are rematerialized by a special *Analyzer* module (cf. Figure 3), which parses the respective HTML encoding for automatically extracting significant "keyworks" summarizing the contained contents as accurate as possible. Extracted keywords represent *semantic classes*, and are associated to the analyzed URL by means of proper filters, as well generated automatically by the Analyzer. The keyword extraction algorithm developed so far is still based on pure syntactic heuristics; semantic or ontology-based ones are under investigation and promise to enhance the accuracy of the keyword extraction algorithm on the one hand, and to provide means for clustering and categorizing keywords on the other one.

Although a good design of Web sites should allow covering with proper semantic classes most of their pages and contents, we do not believe in the usefulness of a complete coverage of all pages and contents by means of semantic classes. A semantic classification of error pages, for example, would be rather confusing to users. Therefore, even within a subscribed application, there may be tracked pages without any further associated semantics.

3.6. xMem Filters

According to their scope within xMem, we distinguish two kinds of filters, global filters and local filters. *Global*

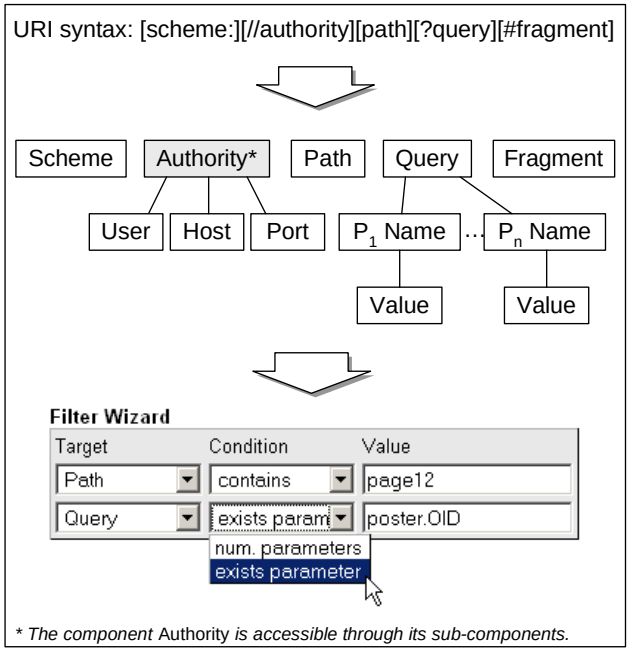

Figure 4. Decomposing URL strings and defining rules over its components.

filters are built upon the URLs of applications and check whether incoming URLs belong to one of the subscribed applications. Their scope is global within xMem and thus they are applied to all tracked requests. Suitable global filters are automatically created during the subscription of an application.

Conversely, *local filters* are only applied, once a URL has been associated to a particular application. The mechanisms mentioned in the previous section are adopted for deriving semantic classes covering the significant pages of an application. For each of these classes proper filters must be specified. Due to the fact that more different URL structures could lead to the same concept, several filters may be defined for the same class. At least one of them must match a particular URL in order to associate URL and class.

Filters consist of one or more rules and one action. For executing the action, which consists in associating URLs and semantic classes, all rules of the filter must evaluate to true. A rule is composed of target, condition and value; Table 1 shows the possible instances for the first two, the *value* is provided by designers (cf. Figure 4). Target elements are derived from URLs according to [4], which classifies URLs registered by the *xMem Tracker* as absolute, hierarchical and server-based URIs (Uniform Resource Identifiers). Figure 4 graphically illustrates the applied decomposition of URLs and also shows how, based on the resulting atomic components, rules are defined by means of a proper Filter

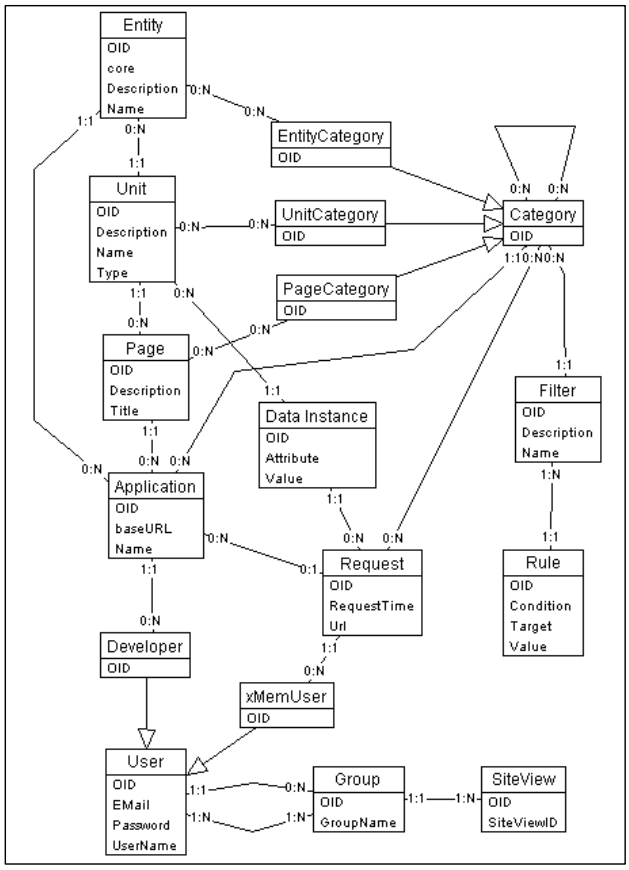

Figure 5. The xMem prototype data source.

Wizard. The *Filter Wizard* provides an interactive user interface for defining rules for filters. The so collected user inputs are translated at runtime into suitable regular expressions, one for each rule of the filter, and evaluated for the respective target component.

4. Prototype

We are currently working on a prototype tool for assessing the usability and usefulness of the xMem concepts. The following subsections provide insight into the state of the art of xMem. For presentation purposes, the proposed Web interface and the structure of the underlying data source are a simplification of the actual prototype.

4.1. Data Model

Figure 5 shows the Entity-Relationship diagram of the prototype data source containing the repositories outlined in section 3.1 within a single database. The database is shared among the various xMem components, but its design is particularly tailored to the requirements of the *Navigator* WebML application.

Entities and repositories can be matched as follows: The entities *User*, *Group* and *SiteView* implement the user model within WebML applications [17] and thus respond to the needs of the *User Repository*. The entities *xMemUser* and *Developer* further refine the repository, representing specific properties of the two user classes. The *URL Repository* can be associated to the entity *Request*, that collects navigated pages and request times. In addition, in case of WebML applications, also references to the specific data instances visited can be stored (entity *Data Instance*). The *Semantics Repository* comprises the entities *Application*, *Entity*, *Page* and *Unit* for collecting the data extracted from submitted WebML schemas. The entities *Category*, *EntityCategory*, *UnitCategory* and *PageCategory* then build the proper *Semantics Repository*. Finally, the *Filter Repository* consists of the entities *Filter* and *Rule*. A filter can have one ore more rules and is associated to its specific category. The rule comprises the attributes *Target*, *Condition* and *Value*, which reflect the rule structure introduced in section 3.6. Each application has its own set of categories and thus its own set of associated filters.

4.2. The xMem Tracker

The Tracker is a small Java HTTP sniffer program to be installed on the client PC. Once launched, it operates in a completely transparent manner in the background. Users have the option to turn the tracker on and off, thus activating and de-activating xMem. While running, it is accessible by means of a small tray icon in the operating systems' taskbar. A mouse click on the icon opens the context menu and provides the user with facilities for managing his authentication data and for starting and stopping the tracking service. No data is sent without the explicit permission of the user. Tracked page requests and user identifier are encoded and sent to the xMem Server via the Internet as common HTTP requests.

The Tracker Server consists of a single Java servlet that listens for incoming messages and, after successfully authenticating the user, adds the request properly formatted to the user's log data.

4.3. The xMem Navigator

Figure 6 gives an idea of the Web interface of xMem and outlines the basic functionalities offered by the Navigator prototype. Two landmark areas (reachable through the application's main menu) are available, one concerning user profile management, the other providing proper history browsing facilities. The home page *User Details* of the site view (labeled with an "H") also is the default page

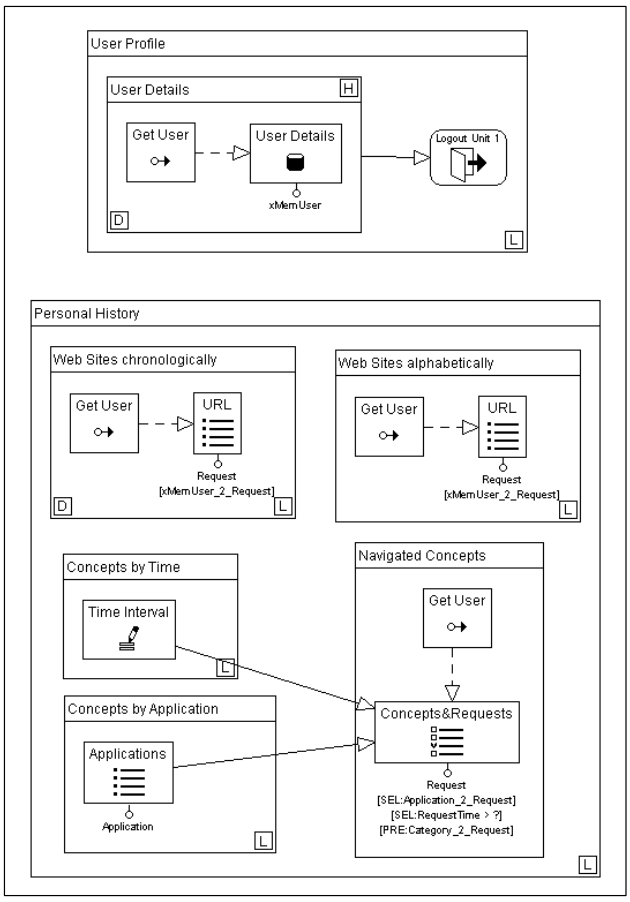

Figure 6. WebML schema for the site view of xMem users.

(label "D") of the *User Profile* area; thus it is presented to users as first page once they enter the area. The area *Personal History* has as default page the page *Web Sites chronologically*. That page and the page *Web Sites alphabetically* respectively show chronological and alphabetical lists of navigated requests, personalized with respect to the current user. The page *Navigated Concepts*, at the other hand, exploits the additional semantics stored in the data source in form of different categories and associated by the wrapper mechanism to the visited pages. This results in a page that provides a hierarchical list (in form of a so- called *multidata unit*) of navigated Web sites structured by meaningful categories. From this point on, the user can browse the concepts according to several criteria (similar concepts, concepts seen during previous or subsequent visits, etc.).

Once a user has successfully retrieved the concept he was searching for, the leaves of the hierarchical list show all URLs that match the interactively specified access conditions. By clicking on one of the URLs the user is forwarded to the actual page (within a new browser window).

4.4. The xMem Administration Tools

Besides the site view for xMem users, the Navigator also provides proper site views for 3rd party application designers, administrators and a public site view for logging in. The administrator's site view serves mainly the purpose of managing users, groups and applications. The designer's site view allows subscribing Web applications, being they WebML applications or not. A detailed presentation of this two site vies is out of the scope of this paper; however, section 3.5 and Figure 3 provide little insight.

4.5. The Classifier

The Classifier is currently implemented as integrated component of the Tracker Server. The URL decomposition mechanism is based on the two Java classes `URL` and `URI`, whereas the filter parsing mechanism makes extensive use of the Java `regex` package. Incoming URLs are instantly categorized and stored into the database, by extracting semantic information such as entity names or object instance identities from them.

5. Related Work

Most commercial browsers incorporate history mechanisms. However, users still do not extensively use them. Catledge and Pitkow [6] state that these complex history mechanisms are often not used; and the 40.6% of the user actions involve the browser's back button. In [12] results show that "0,1% of page accesses were through the history list, 42% of page accesses used the back button" for all of the pages navigated by subjects.

These and many other findings suggest that current browsers lack an efficient method for revisiting Web pages, and that "current interface for browsing on the WWW are still primitive... They do not aid users in accessing already visited pages without much cognitive demand" [13].

History mechanisms can be distinguished in *passive* and *active*. The first class works by maintaining some kind of information about a particular Web session. The browser stores some subset of the pages visited by the user, typically in a list. However, since most history mechanisms store only the links on the last spoke traversed (i.e., the current path in a depth-first traversal), by using this mechanism the user may only reach a small portion of previously visited pages. If, when deep in a search tree, the user finds an interesting page (i.e., one to which she/he may wish to return), it may be difficult to preserve that link while continuing with the search.

Active mechanisms, such as bookmarks, provide users with a method to mark interesting Web pages they would

like to return to. For most people, however, bookmarks create a new difficulty: as users find a large number of pages interesting, bookmark lists quickly become long and unwieldy. Although most browsers provide mechanisms for editing and maintaining bookmark lists, they are not an ideal mechanism for maintaining context within a single browsing session. In general, it seems that active history mechanisms such as the back button and bookmarks function are more effective to support short and long term information memories retrieval than what the passive mechanisms do. In fact, if a user is navigating a Web site it is more easy to him to go back to the pages he wants to revisit using the back button then opening the history mechanism window, analyzing the information displayed as URL, choosing one among them and click on it.

On the other side, users experience frustration in retrieving already visited pages when a certain amount of time is passed from the first visit. The reason is that the context in which that page was being viewed is lost. The path-following method (on which the most part of the history mechanisms are based) for retrieving long term history memories imposes users to traverse in reverse order their previously visited pages. This method relies on users remembering their navigational behaviors, either because they must recall the page visited and their sequence or because they must realized that they can return to a page by retracing a particular pathway. Indeed, the lack of efficient mechanisms for maintaining context may be partially responsible for this phenomenon.

Few researchers have considered new metaphors for browsing and collecting information on the Web; we next briefly describe their work. Most of the methods introduced in the following use a mix of graphical representations and a sense of context. Some of them place on the end users an extra request of cognitive effort. Most of them are features of experimental browsers and are only able to show syntactic information, not semantic information.

IBM's Web Browser Intelligence (WBI) tool [1] is a browser companion with personal history functions intended to make Web browsing more efficient by annotating hyperlinks on all Web pages with traffic signals, and this performs well for functions such as: remembering visited pages, providing a keyword search through the text of pages already visited, noticing patterns in the Web browsing behavior and suggesting shortcuts, and automatically checking favorite Web pages for change.

Some types of systems present graphical maps through which a user may navigate. Some are like HTML frames in that they are created once by the content provider. Others are created on the fly by the Web browser. For example, WebTOC [15] is an automated system for creating table of contents (TOC) frames for sets of Web pages. The TOC frame can be quite useful, and having that frame actually run a Java program allows it to more dynamically present the desired information. Its main drawback is that it occupies a large portion of the screen. WebNet [8] is a browser extension that displays a graphical representation of the hyperspace being explored. It does so dynamically, and independent of the content provider. In fact it can do so across many sites. It is a challenge, however, to present the graph in such a way that the contextual information is highlighted.

DeckScape [5] is an experimental browser based on the concept of deck. Each deck is a linear stack of pages that the user can leaf through. As with history mechanisms, if a user starts from page A and goes to B, B is added to that stack or deck. However, unlike history mechanisms, if the user goes back to A and then traverses a link to C, B is not lost; it remains in the deck of pages. Also, the user can always revisit any page since no page is ever lost. However, users are cognitively loaded with the responsibility of maintaining pages logically in different stacks unless decks are pruned regularly.

Elastic Windows [13] also provides a mechanism that illustrates graphically the hyperspace in which a user is navigating, but it does so more interactively. If the user selects a link using this system, the contents of the corresponding page do not replace the currently displayed page; instead, the new page is displayed alongside its parent. Selecting multiple links from a page results in all the new pages being displayed alongside the parent, but in a smaller size. The same operations may be performed on any window in the browser. Users are provided with functionality to manage the windows by collapsing some sections of the hierarchy, while maximizing the size of others. Since the complete hierarchy is visible at anytime, users can easily move in the hierarchy while not losing their search context. The simultaneous display of multiple pages again places a management burden on the user; screen real estate can easily become cluttered if not managed efficiently. Cyclic links are not dealt with very well; the same page may be displayed multiple times, further wasting screen real estate. Elastic windows enables users to look at many Web pages at once, whereas a breadth first navigation technique provides a linear order for web visitations.

Scratchpad is a prototype incorporated in Chimera 1.70 [14], an experimental browser. Scratchpad introduces two mechanisms to support web navigation: a re-visitation mechanism called Dogears serves as a set of temporary bookmarks, and a breadth-first traversal mechanism is adopted as alternative navigation strategy to depth-first traversal. Scratchpad is not a history mechanism, but a navigational planning tool, applied to support users' visits of new pages. The information gathered by the breadth-first search and regarding possible future page visitations opens new ways for intelligent pre-fetching of those pages, thereby allowing faster navigation. Both mechanisms may

be added to existing browser technologies [16].

More recently [11] the Microsoft research center has worked on the system *tuff Ive Seen* (SIS), which is able to support users in retrieving and reuse information already seen locally on the PC. The system aim at facilitating the information seeking behavior by providing an index of information that a user has seen (email, web page, document, appointment) and, in addition a set of rich contextual cues about the searched information, made available from the previous accesses.

All solutions described above respond to the need to record users' navigational history (URL lists) for allowing the successive re- access of visited pages. xMem works beyond these mechanisms analyzing and interpreting the structure of recorded URLs and making available the results of this process to end users. This is obtained independently from the browser in use.

6. Conclusions

We have argued that current browsing functionalities do not adequately support retrieving information from a user's navigation history. xMem improves passive history mechanisms by making use of new semantic criteria to organize and show the navigational history instead of simply exploiting time-sorted history mechanisms that prevail today. This retrieving strategy makes navigation to already visited information easier and more effective because it provides a much richer notion of semantic context in which information has been seen. Context-dependent history mechanisms sustain users' episodic memory of the visited web pages.

Further investigations are necessary to refine our prototype, especially for what concerns the analysis of generic Web applications. We are currently studying wrapping technologies such as LIXTO [2, 3] and RoadRunner [9, 10] to see if their functionalities apply to the xMem context. We will also extensively experiment the use of the prototype, so as to assess its usefulness and usability. We also are interested to gain insight into the aspects of the history mechanisms that are most frustrating to users in an effort to suggest improvements for such features for future implementations.

References

[1] R. Barrett, P. Maglio, and D. Kellem. Web browser intelligence: Opening up the web. In *Proc. of COMPCON'97*, 1997.

[2] S. Baumgartner, S. Flesca, and G. Gottlob. Supervised wrapper generation with lixto. In *Proc. of VLDB'01, Rome, Italy*, 2001.

[3] S. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with lixto. In *Proc. of VLDB'01, Rome, Italy*, 2001.

[4] T. Berners-Lee. Uniform resource identifiers (URI): Generic syntax. RFC 2396, 1998.

[5] M. Brown and R. Shillner. Deckscape: An experimental web browser. Proc. of WWW'95: Technology, Tools and Applications, April 1995.

[6] L. Catledge and J. Pitkow. Characterizing browsing strategies in the world-wide web. Proc. of WWW'95: Technology, Tools and Applications, April 1995.

[7] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kauffmann, 2002.

[8] A. Cockburn and S. Jones. Which way now? analysing and easing inadequacies in www navigation. *Int. J. of Human-Computer Studies*, 45(1):105–129, July 1996.

[9] V. Crescenzi, G. Mecca, and P. Merialdo. Automatic web information extraction in the roadrunner system. Proc. of the DASWIS'01 ER Workshop, 2001.

[10] V. Crescenzi, G. Mecca, and P. Merialdo. Wrapper-oriented classification of web pages. Proc. of ACM SAC'02), 2002.

[11] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. Robbins. Stuff ive seen: A system for personal information retrieval and re-use. In *Proc. of SIGIR03, July 28 August 1, 2003, Toronto, Canada*, 2003.

[12] GVU's WWW Surveying Team. Gvu's 10th www user survey, 1998.

[13] E. Kandogan and B. Shnaiderman. Elastic window: A hierarchical multi-window world wide web browser. In *Proceedings of UIST'97*. ACM, 1997.

[14] J. Kilburg, G. Niklasch, and W. Edmond. Chimera-1.70, x11/athena world-wide web client. http://www.unlv.edu/chimera.

[15] D. Nation, C. Plaisant, G. Marchionini, and A. Komlodi. Visualizing websites using a hierarchical table of contents browser: Webtoc. Proc. of Human Factors and the Web'97, Denver, June 1997.

[16] D. Newfield, B. Sethi, and K. Rayll. Scratchpad: Mechanisms for better navigation in directed web serarching. In *Proc. of UIST'98*. ACM, 1998.

[17] Politecnico di Milano. The web modeling language. http://www.webml.org.

[18] L. Tauscher and S. Greenberg. How people revisit web pages: Empirical findings and implications for the design of history systems. *Int. J. Human Computer Studies*, 1997.

[19] WebModels. Webratio site development studio. http://www.webratio.com.

A Reusable Personalization Model in Web Application Design

Irene Garrigós, Jaime Gómez
Universidad de Alicante
Campus de San Vicente del
Raspeig, Apartado 99 03080
Alicante, Spain
{igarrigos,jgomez}@dlsi.ua.es

Peter Barna, Geert-Jan Houben
Techniche Universiteit
Eindhoven
PO Box 513, NL-5600 MB
Eindhoven, The Netherlands
*{*P.Barna, G.J.Houben *}@tue.nl*

Abstract

Personalization of web sites has become an important issue in web modeling methods. Many of these methods use similar approaches to specify personalization. However, even with similar (personalization) requirements the resulting implementations differ. The consequence is that we cannot reuse the same (personalization) specification. This paper presents a generic extension for existing conceptual modeling approaches to address the particulars associated with the design and specification of dynamic personalization. We allow the designer to specify, at design time, how the website will be personalized (at runtime). We claim that the personalization specification can be mapped to different web modeling methods so we can define this specification once and reuse it for different (development) environments. For this purpose, a high level (generic) language based on rules (Personalization Rules Modeling Language) is presented. Finally, we describe how PRML rules can be easily mapped to the rule representation for a commercial rule engine.

1. Introduction

Web idiosyncrasy introduces new challenges for the design process that go further than the specification of the navigation map, and that include aspects like the need for continuous evolution, and the treatment of a heterogeneous audience (that implies different user's needs, goals, preferences, capabilities etc. in our website). In this context the personalization of websites has become an important issue in web modeling methods [6] [8] [9] [11] [12] [15], due also to the effect of the diversity of personalization policies over all the development cycle of applications. Many of these methods tackle very similar problems and use conceptually similar techniques (personalization model based on rules, user model, etc). Despite of this, even with the same (personalization) requirements the final implementations differ. This is why we cannot reuse the same (personalization) specification. When adaptation rules (from personalization model) are specifically defined for an application environment the re-usability of rules is a big challenge [14]. Only when rules management is separated from application development personalization will achieve maximum flexibility.

It is obvious that many existing approaches offer powerful specification means and tools for personalization. However, in most cases they are not portable, typically use different specification techniques and are aimed and implemented for different design methods. In this paper we offer a solution that preserves the advantages of the existing approaches, and in our opinion it also shows a good balance of expressive power (aimed to personalization) and portability – its ability to be (automatically) mapped to rather different conceptual modeling methods using different specification techniques. This solution is based on a high level rule language called PRML (Personalization Rules Modeling Language) that the designer can use to specify (at design time) the personalization to be performed at runtime. The specification of personalization is separated from any other application functional specification, so it can be specified once and used for different (development) environments. The personalization specification can be mapped (implemented) to different existing web design approaches. To demonstrate our claims, we discuss its mapping to OO-H [7] [8] and Hera [9] [18] representing rather different approaches to web design.

The paper begins describing backgound in section 2. Section 3 explains how, in a web design model context, personalization is applied. An example specified in the methods studied in the paper demonstrates two different

modeling techniques where personalization is modeled. In section 4 the PRML language is presented and the mapping from PRML to OO-H and Hera is described. In the next section we show how PRML rules can be easily mapped to the rule representation for a commercial rule engine. Finally we present conclusions and further work.

2. Background

Adaptation rules are widely used in the literature for specifying adaptive websites. There are different kinds of adaptation rules depending on the object of the adaptation: we focus on the user-related adaptation rules (personalization rules). A personalization rule will state when concrete users see certain pieces of content and how specific functionality is presented. Some modeling approaches have their own language for defining rules.

For example, for describing navigation goals WebML [6] uses the WBM formalism based on a timed state-transition automata. This formalism combined with WebML form a high-level Event-Condition-Action paradigm. In WebML an event consists of a page request, the condition is expressed as a WBM script and represents a set of requeriments on the user navigations, and the action is expressed as a WebML operation chain and can specify adaptivity actions. In WSDM [3][4], in the context of audience driven design, a language is provided to help the designer to define adaptive behavior, focusing on the adaptation for a group of users by adapting the structure of and the navigation possibilities in the website. WSDM doesn't support personalization, but this language could be used to personalize. UWE [11] in the context of an OO method, define adaptation rules as OCL expressions. A rule is modeled as a class Rule that consists of one condition, one action and attributes. Rules are classified according to their objectives into: construction rules, acquisition rules and adaptation rules.

In WUML[10], in the context of ubiquitous computing, customization rules allow the designer to specify the adaptation to perform when a certain context is detected. These rules are specified within an UML annotation, using the stereotype <<CustomizationRule>>. The specification of such a rule comprises a unique name, a reference to one or more requirements, and an ECA triplet. The annotation is attached to those models elements being subject to customization. Thus the customization rules can be attached to any web application modeling using UML as the basic formalism.

However, when personalization rules are specifically defined for an application environment we find some problems: the re-usability (in different approaches) of the personalization strategy is (almost) impossible. Moreover, due to the heterogeneous ways to personalize in the different methods, it is difficult to compare them by the (level, kind of) personalization they support.

Also, there is a big number of commercial tools (e.g. ILog JRules, LikeMinds, WebSphere, Rainbow..) that make easier the use of the personalization techniques and strategies and give support to many personalized web applications. These tools are oriented to the implementation of personalization strategies. Some of them also allow to personalize in basis of rules The main problem of these approaches is the low abstraction level that causes reuse problems and a difficult maintenance and scalability of the resultant personalized applications.

3. Contextualization of Personalization in Web Design Methods

In the context of web design methods, the existence of conceptual models independent of the final implementation language makes easier the personalization (adaptation) of the web applications to the environment changes, reusing the information captured during the process of analysis and design. Focusing on the conceptual modeling phase, every design method has three main layers: a Domain model, in which the structure of the domain data is defined, a Navigation model which defines the structure and behavior of the navigation view over the domain data. Finally, the Presentation Model defines the layout of generated hypermedia presentation. To be able to model adaptation/personalization at design time two new models are added (to the set of models): (1) A Personalization Model, in which personalization rules are defined to store information needed to personalize and to specify different personalization policies. (2) A User Model, which allows to store data needed for personalization besides the beliefs and knowledge the system has about the user. This information builds the user profile and it is needed to base the personalization on.

As already mentioned, we propose to create a reusable personalization model. In this paper we are going to focus on two modeling methods different enough to show the generality of our proposal. The OO-H Method [7] [8] is based on the OO paradigm that provides the designer with the semantics and notation necessary for the development of web-based interfaces. Hera [9] [18] is a model-driven methodology for designing web applications. Both methods have three layers for designing a web site. In this paper we (only) focus on the personalization of the structure of the website (not in presentation details). In the next subsections we define the models of the conceptual modeling phase, and compare these two methodologies.

We study a (simplified) book store shopping basket system as a running example. Consider the following personalization requirement: *we want the user to be offered a list of books written by authors in which user has certain interest degree.*

3.1 Domain Model and User Model

In figure 1 we see the DM and the UM for OO-H for the running example. The UM is represented as part of the DM. These models are defined (in OO-H) as UML compliant class diagrams. Concepts are represented by classes with attributes. There are associations between them representing Concepts relationships. We can store or update the information of this model by means of ECA [5] rules as we will show in section 4.

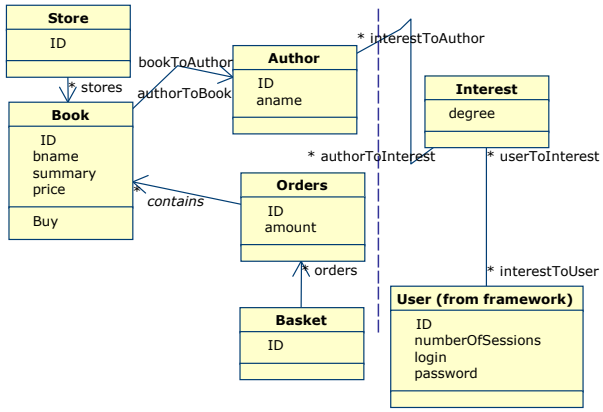

Figure 1 DM-UM for OO-H (expressed in UML)

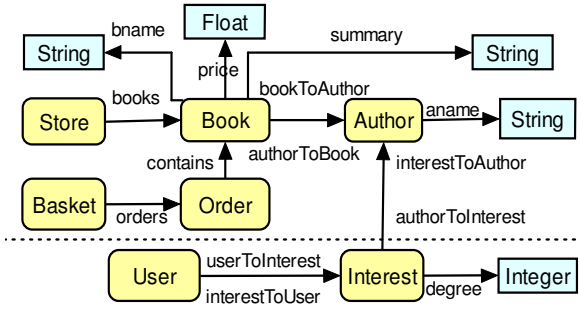

Figure 2 DM-UM for Hera (expressed in RDFS)

In figure 2 we can see the Hera DM and UM for the same example. In Hera, all models are represented with RDFS[2]. Ovals represent Concepts and rectangles represent literal Attributes. In both methods, for the purpose of the personalization requirement defined, we store in the updatable UM the user's interests on authors.

3.2 Navigation Model

A generic Navigation Model (NM) will be composed of Nodes and Links. Nodes are (restricted) views of domain Concepts. Each Node has associated a (owner) Root Concept from the DM. Browsing Events are associated to the Links of the NM. A NM describes a navigation view on data specified by DM (and possibly also UM). Moreover it also captures the functionality (personalization) of the designed application.

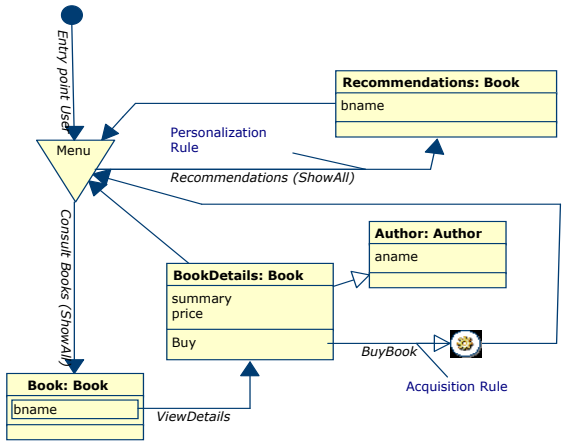

Figure 3 OO-H Navigation Model

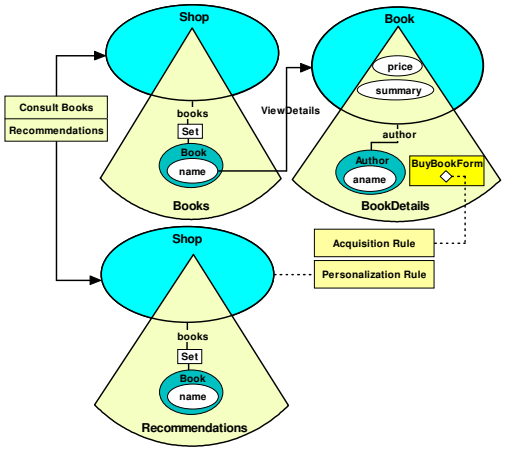

Figure 4 Hera Navigation Model

In figure 3 we can see the NM for OO-H, and in figure 4 for Hera for the running example. In both cases, the final website will have as a starting point: the home page where we can find a collection of links: "Consult Books" and "Recommendations" in this example. If the user navigates through the first link (Consult Books), he will find a list with all the books in a new page. This set of books is an indexed list, so the user can click in one of the books names to view the details of each of the books. When the user clicks on "Recommendations"

personalized book suggestions should appear (based on the information from UM). To fulfill this personalization requirement (Users will see recommendations of books based on their interests in authors) we need to define a personalization model to adapt this NM. It will be explained in section 4.

The OO-H NM is composed by navigational classes, which represent views of the classes of the domain model (the above defined Nodes). In each Node we have defined the Root Concept from DM (or UM) attached to it by the notation: "RootConcept:Node". In the Hera NM the ovals with the slice shapes represent navigation Nodes. Every slice is based on a DM (or UM) Concept (Root Concept) indicated by the title on appropriate oval. Overlaying areas of slices and ovals contain attributes of the (root) concept (e.g. price in Books.Detail) and attributes of related concepts are located in the slice areas not overlapping with ovals (e.g. aname in Book.Details). Arrows represent links that have attributes as anchors and slices as targets. For a more extensive description of OO-H and Hera NM we refer the reader to [8] and [9] respectively.

In the next section we introduce the Personalization Rules Modeling Language to specify a personalization strategy and explain how it can be mapped to OO-H and Hera methods.

4. Personalization Rules Modeling Language

The rules of the personalization model will be defined using an effective, simple and easy to learn language. As already mentioned, the purpose of this language is to help the web designers to define all the rules required to implement a personalization strategy. The basic structure of a rule defined with this language is the following:

When *event* **do**
    If *condition* **then** *action* **endIf**
endWhen

The rules we can define with this language are ECA [5] rules. A rule is formed by an event, the body of the rule, a condition (optional) and an action to be performed. For satisfying a personalization requirement we have to define how to obtain the required knowledge to personalize (acquisition rule). This information will be stored in the UM. Also, we have to define the effects this personalization causes in the system (personalization rules). These rules use the information from the UM to describe adaptation actions. Depending on the object of the adaptation two kinds of personalization rules are considered: navigation rules (to alter navigation), and content rules (to add/remove/adapt content). Due to space limitations we can only show an excerpt of the BNF specification of PRML. Symbols and words in bold denote keywords. (In the paper for clarity reasons rules have been simplified omitting the features of the rule, e.g. priority, expiration date, etc.).

```
<rule>::= <features> When <event>do <body>
endWhen
<body>::= [<periocity>] [If ( <condition>) then]
<action> [endIf]
<event>::= (Start | <navigation> |
<methodInvocation> | End )
<navigation>::= Navigation. <link>
<methodInvocation>::= MethodInvocation. <link>
<periocity>::= SetPeriocity <operator> (always |
everyday | eachmonth | <userdefined>) [1]

<condition>::=(<relexpression> [<booloperator>]
{<relexpression>} )
<relexpression>::= <operand> <operator>
(<value>|<operand>) | not <relexpression>
<operator>::= < | > | = | <= | >= | <>

<action>::= <SetContent> | <Show> | <Hide> |
<Sort> | <SortLinks>
<SetContent>::= SetContent(<attribute>,
<content>)
<attribute>::=<attname>
<attname>::=<class> . <attrib>
<content>::=<attname> | <value>
<Show>::= Show(<attname>)
<Show>::= Show(<link>)
<Hide>::= Hide(<link>)
<Sort>::= Sort <class>  orderBy ASC | DESC |
<attname>
<SortLink>::= SortLinks <link> orderBy ASC |
DESC | <link>
```

For the requirement considered in the running example we need the following rules: We need an acquisition rule to update the user's interests (in the UM) when the user consults a specific book. We also need a personalization rule that will do the following: "When the user clicks on the link 'Recommendations' we show the books of authors with an interest degree > 100" (checking the updatable value from UM).

The three parts that form a rule (event, condition, action) are explained in next subsections, by means of the running example, to illustrate the use of the PRML

4.1 PRML: Event Specification

First, we consider the events. We want an active system that monitors situations of interest to personalize and to trigger the proper response when one of these

[1] Periocity states a certain interval of time in which the rule condition has to be checked.

situations occurs. This response is specified by an action in a rule, so to trigger that action we need events. Each time the user activates a link (activelink), the associated event is launched causing the evaluation of a rule. When a link is clicked a node is also activated (activenode). We consider the following types of browsing events:

- *Navigation event:* implies the activation of a navigational link and the activation of a node (i.e. the new page resulting of the navigation). Links have a parameter indicating the instance or set of instances that are going to be shown (and the navigation pattern, i.e. *showall, index*..) in the activated node. When the link is activated the rule/s attached to this event is/are triggered passing to the rule/s this parameter to build, in basis of this information, the active node. We need the data before the node resulting from navigation is generated (in order to use those data in the rules and build the adaptive web page). This information is passed in the navigation event (when possible and needed) as a parameter[2]. This parameter can be a simple parameter (when the data on the webpage is the instance of a concept of the domain model) or a complex parameter (when the data of the webpage is a set of instances of a concept of the domain model). In PRML we can write: **When** *Navigation.activelink(NM.activenode parametername)* **do** to express this event passing a single parameter, and **When** *Navigation.activelink(NM.activenode* parametername)* **do** when we pass a complex parameter. In the running example we could express: **When** *Navigation.ViewDetails(NM.Book book)* **do** to specify the event of a rule triggered when the user browses the link "ViewDetails". As a parameter we add the active node "Book". In the following case we pass a complex parameter when the user clicks on the "ConsultBooks" link: **When** *Navigation.ConsultBooks(NM.Book* bookset)* **do.**
- *Method Invocation event* implies the invocation of a method defined in the website. To express this event in PRML we can write **When** MethodInvocation.*activelink(NM.activenode)* **do.** In the *Method Invocation* event we also pass a parameter containing the instance of the root concept of the node containing the invocation to the method, this parameter can be also simple (i.e. an instance) or either complex (i.e. set of instances). If we would like to add a rule when the user invokes the service *Buy Book* in PRML we would express it as follows (simple parameter): **When** MethodInvocation.*BuyBook(NM.Book)* **do.**
- *Start event* is associated with the entrance of the user in the website (i.e. the start of a session). In PRML this event is expressed as follows: **When** *SessionStart* **do**.
- *End event* implies the exit of the user from the website (i.e. end of a session). In PRML we can express this event as follows: **When** *SessionEnd* **do .**

Mapping *Events* to OO-H and Hera

In OO-H the events are associated with the different types of link in the NM. *Navigation events* are associated with navigational links in the navigation diagram (see fig. 2 , e.g. View Recommendations, Consult Books). *Method invocation events* are launched when the user clicks on links that invoke a service (*service* links*,* e.g. *Buy Book*). The *Start event* is associated with the *Entry point* link in the navigation diagram. The *End event* is triggered after certain inactivity of the user in the system or when the user clicks activates an *exit* link.

In Hera events are associated either with links or with processing of forms (user inputs). A concrete link event corresponds to the instantiation of its target slice, and a form processing event corresponds to the execution of a data (manipulation) query. Another events are associated to the instantiation of a browsing session (for a single user), and its expiration. Hence, we can consider the following event types analogous:

- *Start event* is *Session instantiation* in Hera,
- *End* event is Session expiration in Hera,
- *Navigation* event is Slice instantiation, here the type and data instance of the source slice (the slice where the link anchor was activated) are default session parameters (sliceid, conceptid), and
- *Method invocation* event is Form processing in Hera, since the application logic is in Hera provided by form processing queries.

4.2 PRML: Condition Specification

Once rules are triggered, only if a condition is satisfied an action is performed. In the case of personalization rules the condition is based on information stored in the UM. In PRML to indicate that we are accessing data of that model, we add the prefix "UM" before the path to access that information. Conditions can be based on user-specific information (independent of the domain, e.g. we consider user characteristics, user context, etc) or on domain dependent information (e.g. number of clicks on a certain link). In the example, in the condition of the personalization rule (once the rule has been triggered) we check the interest degree in a certain author (i.e. greater than 100, the condition is based on domain dependent information). For this purpose we need to check all the instances of the UM concept "interest". In PRML it is expressed as follows "**If** (UM.userToInterest.degree>100)

[2] All the parameters have the prefix NM to indicate that they come from the Navigation Model.

then"[3]. We use path expressions to access the data of the condition of a rule of the type: "ConceptRelationship[4].Attribute". (In the case of accessing the information of the UM, the path expression will start with the User concept, if we have to start from any other class the path expression will start with the name of that class).

Mapping *Conditions* to OO-H and Hera

The mapping of the *conditions* to OO-H and Hera methods will be done in conjunction with the mapping of the actions. The reason for this is that PRML rules may contain implicit bindings of expressions in the condition and action parts. For instance in the excerpt of a rule:

```
If (UM.userToInterest.degree>100) then
    Show(UM.
userToInterest.interestToAuthor.authorToBook.bname)
endIf
```

only these instances of the Interest class given by the condition expression are taken into account in the action part (are shown). This is a powerful mechanism, but it also means that we need to consider mapping of whole condition-action parts into different methods rather than to map the conditions and actions separately. We will see it at the end of the next subsection.

4.3 PRML: Action Specification

The action part of a rule specifies the operations to be performed when the rule is triggered and its condition is satisfied. The action, in case of the personalization rules, will specify the adaptation action to perform. In the acquisition rules, we will specify which information is stored in the UM. We show now how to adapt the content and the navigation using this language in the context of the running example.

- **CONTENT**

For adapting the content we can perform several actions on it. We can, for example, set new content to an attribute. For instance, the acquisition rule needed for the shopping basket example is going to change the content of an attribute from the UM.

```
When navigation.ViewDetails(NM.Book book) do
If(book.bookToAuthor.aname=UM.userToInterest.Ito
Author.aname) then
SetContent(UM.userToInterest.degree,UM.userToInt
erest.degree+10)
  endIf
endWhen
```

This rule is triggered when the user activates the link "ViewDetails". It updates the interest degree on the author of the consulted book using the SetContent statement in which we specify the attribute that we want to modify, and the value or formula that calculates the new value. This rule compares implicitly each of the instances of the class interest with the consulted instance (to properly update the value).

As explained before, we need to keep the data from the navigation previous to trigger the event, to use that information in the rules. For this purpose, in this example in the event of the rule we pass as a parameter the visited instance of the class Book (from the NM). That is why this parameter has the prefix "NM". The rest of the data of this rule have the "UM" prefix, indicating that information is stored in the UM.

- To adapt the content, we could also select the content to show from a class, or depending on a certain condition. The personalization rule of our running example shows this type of content personalization.

```
When navigation.Recommendations(NM.Book* books)
do
    If
(books.booktoAuthor.authortoInterest.degree>100)
and (UM.userToInterest.interestToAuthor.ID=
books.booksToAuthor.ID) then
            Show(books.bname)
    EndIf
endWhen
```

This rule is triggered also by a navigation event, when the user clicks on the link "Recommendations". When the user activates that link, book titles of the authors with interest degree greater than 100 (specified condition) are shown.

We could also sort the content by a certain value, (specifying if the order is ascendant or descendent), as an example we could sort the books by book name:

```
When navigation.ConsultBooks(NM.Book* books) do
    Sort books orderBy ASC books.name
endWhen
```

The rule is triggered when the user goes through the link "ConsultBooks" and sorts the books ordered by book name in an ascendant way. This rule has no condition so when the user activates that link, book titles are sorted.

- **NAVIGATION**

For adapting the navigation we can perform diverse actions over the links. We can sort the links on a page, and we can show or hide links. For example we could

[3] The loop to check all the instances is not expressed in the rule, it is done implicitly.

[4] The ConceptRelationship statement can be repeated 1 or more times depending on what we need to access.

hide the link "ViewRecommendations" for users that have not consulted any book yet.

```
When navigation.ViewRecommendations(NM.Book*
books) do
    If (NM.ViewDetails.nclicks>0) then
        Show(NM.ViewRecommendations)
    endIf
endWhen
```

Mapping *Conditions* and *Actions* to OO-H and Hera

As already explained in previous section, the mapping of the conditions and actions is done together. The specification of conditions and actions in PRML can be seen as a declarative query language designed for personalization purposes. This approach allows for transformation of PRML rules to different query languages ranging from OO (like OCL) to relational (like SQL), or Semantic Web query languages (like RQL, SeRQL).

To avoid possible problems with automatic translation of PRML rules (condition and action parts) to methods relying on declarative query languages (OO-H and Hera among others) we apply to PRML rules the following restrictions:

- *Action* parts can contain only one operation, because is not always possible to implement an arbitrary sequence of operations by a single query or automatically derive a postcondition of such a sequence, and
- *Action* parts cannot activate other rules (they don't trigger events); this restriction would facilitate the deployment of PRML and would avoid risks of infinite loops and deadlocks.

In general, PRML actions contain instance selection (filtering), instance sorting, and instance modifications. Let's see how PRML mapping is done to OO-H and Hera methods. In case of OO-H, conditions are going to be mapped to OCL [17] expressions. PRML actions are translated into methods of OO-H action classes of the Personalization Model (represented as a class diagram). In this diagram rules are implemented as subclasses of the Rule class, including specializations for different kinds of actions like *SetContent* for modification, or *Sort*, *SortLinks* for sorting. All these OO-H PM classes inherit of the main class *Rule*, which has an abstract *action()* method which must implement the PRML actions. The concrete PRML operations specified by the designer are then converted to OO-H (plus OCL) specification as it is described in Table 1.

Table 1 Overview of implementing selected PRML operations in OO-H and OCL

PRML	Implementation in OO-H
Operation	
Show	• PRML condition and action: applies to precondition of the *action()* method of the *Show* class.
Sort	• PRML condition: precondition of the *action()* method of the *Sort* (*SortLink*) class. • PRML action: postcondtion of the *action()* method of the *Sort* (*SortLink*) class.
SetContent	• PRML condition: precondition of the *action()* method of the *SetContent* class. • PRML action: postcondtion of the *action()* method of the *SetContent* class.

An example of a more complete personalization rule for presenting books in authors, in which the user is most interested is next:

```
When navigation.Recommendations(NM.Book* books)
do
If (UM.userToInterest.degree > 100) and
    (UM.userToInterest.ItoAuthor.ID =
books.booksToAuthor.ID) then
Show(UM.userToInterest.interestToAuthor.authorTo
Book.bname)
 endIf
endWhen
```

The mapping to OCL code for this rule (action and condition) is:

```
navigation.Recommendations(b:
Collection(NM.Book))
Context Recommendations_Show::action()
pre: b->select( i : UM.userToInterest | i.degree
> 100 and
i.interestToAuthor.authorToBook.bname-
>includes(b.bname))
```

Each OCL expression is written in the context of a specific instance, concretely of instances of UM/NM classes appearing in expressions in condition and action parts. In an OCL expression, the reserved word self is used to refer to the contextual instance.

In this example, the precondition determines the selection of the books instances satisfying the PRML rule condition (based on the user interest degree). It applies to the *action()* method of the appropriate Rule subclass. Let's see now an example of how to map to OCL an acquisition rule:

```
When navigation.ViewDetails(NM.Book book) do
If(book.bookToAuthor.aname=UM.userToInterest.int
erestToAuthor.aname) then
SetContent(UM.userToInterest.degree,UM.userToInt
erest.degree+10)
```

```
endIf
endWhen
```

the OCL code would be as follows:

```
navigation.ViewDetails(b: Book)
Context SetContent_VD::action()
pre: b.author.aname =
UM.userToInterest.interestToAuthor.aname
post: UM.userToInterests.degree =
UM.userToInterest.degree@pre + 10
```

The precondition of the appropriate subclass of the Rule class (representing the concrete rule) constrains the *action()* method by applying the original PRML rule condition, and the postcondition determines how the user interest is updated.

In Hera PRML rules are implemented as (SeRQL[13]) queries that can manipulate or select data. Table 2 shows an overview of few selected PRML operations and their implementation in Hera.

Table 2 Implementation of selected PRML operations in Hera

PRML Operation	Implementation in Hera
Show	A selection query associated with an appropriate slice.
Sort	A selection query associated with an appropriate slice containing an ordering directive.
SetContent	A data manipulation query attached to a form submission. Most typically a couple of REMOVE and CONSTRUCT queries (SeRQL does not support UPDATE yet).

In general,,in SeRQL, queries have the form:

SELECT | CONSTRUCT | REMOVE action-variables
FROM Path-expresions-with-variable-bindings
WHERE Transformed-condition

The first clause depends on the action part (e.g. SELECT for filtering, for changing values first REMOVE query followed by CONSTRUCT query, etc.) and the action-variables part contains variables bound with path expressions appearing in the action part of rules. The Path-expresions-with-variable-bindings part contains path expression appearing in action and condition parts and variables. The WHERE clause provides the evaluation of the condition part.

Let's see a mapping to Hera for the following rule:

```
When navigation.Recommendations(NM.Book* books)
do
    If (UM.userToInterest.degree > 100) and
       (UM.userToInterest.ItoAuthor.ID =
books.booksToAuthor.ID) then
Show(UM.userToInterest.interestToAuthor.authorTo
Book.bname)
     endIf
endWhen
```

in SeRQL the query is as follows:

```
SELECT BN
FROM
    {UM:User}userToInterest{}degree{D};
     interestToAuthor{}authorToBook{B}bname{BN}
WHERE
    D > 100 AND B IN SELECT * from session:books
```

This simple query filters books from given selection (stored in session variable session:books and retrieved by the inner sub-query) that are written by authors in which the user degree of interest is greater than 100.

5. Deploying PRML rules under rule engines

Rule engines (RE from now on) provide a comprehensive set of tools for developing, deploying and managing rules (resolve conflicts…). They provide for clear separation of rules from the application code. We claim that this is a good strategy to abstract our personalization specification because we keep the personalization orthogonal to the prime functionality of the application. The idea of developing a RE perfectly adapted to the whole system can seem an attractive option, but the down side of this choice is the time needed to develop it. Moreover, efficient REs rely on extremely complex expertise and algorithms. To reduce efforts and risks the most convenient option would be to use an existing RE like [1] [10] to support personalization rules. The only requisite for using them is to use the rule language they process.

We claim that PRML rules can be easily translated into the rule representation for commercial REs using XSLT scripts. Every web design method with a proprietary rule language would have to map its language to the rule language of the RE. Specifying the personalization with PRML also is an advantage in that sense.

For our purposes we have chosen the Simple Rule Mark-up Language (SRML) [16], an XML-based rule language format initiated by the ILog software vendor company. This language can easily be executed on any conforming RE. In a preliminary attempt to do a mapping to SRML we have achieved the following conclusions:

- *Events* can be mapped to SRML only regarding event parameters. Rules are identified by their names that are transferred to SRML. The proper rule to be triggered when an event arises is selected by the identifier of the

rule's event in SRML that matches the identifier in an executable script. The script invokes the rule engine method passing in the rule name and event parameters.

- *PRML Conditions* can be mapped to SRML conditions which are composed of test expressions, and can be simple conditions or not conditions. *Simple conditions* can be bound to variables while *not conditions* cannot.
- *PRML Actions* for updating the content (*SetContent* statement) can be easily mapped to the *modify* statement in SRML. The mapping to SRML of the actions for updating the presentation and navigation in PRML (filtering, sorting) are still under consideration.

As an example to map to SRML a content update we consider the following PRML rule:

```
When navigation.Recommendations(NM.Book book) do
If (UM.userToInterest.degree>100) then
SetContent(UM.userToInterest.degree,10)
```

This rule can be mapped to the following SRML code:

```
<rule name="NRecommendations">
<conditionPart>
  <simpleCondition className="Interest"
objectVariable="i">
     <binaryExp operator="gt">
       <field name="degree"/>
       <constant type="integer"
        value="100"/>
       </binaryExp>
   </simpleCondition>
 </conditionPart>
 <actionPart>
 <modify>
    <variable name="i"/>
      <assignment>
        <field name="degree"/>
        <constant type="integer" value="10"/>
       </assignment>
 </modify>
  </actionPart>
  </rule>
```

Mapping from PRML to SRML (or to other languages) is still a subject of research and has still to be improved and studied deeply.

6. Conclusions

In this paper we offer a general solution to the lack of reusability of personalization specifications. This solution is based on a high level rule language called PRML (Personalization Rules Modeling Language) to specify at design time the personalization. This specification can be mapped (implemented) to different existing web design approaches. The specification is independent from any other application functional specification, so you can re-use it for different (development) environments. Future directions include the implementation of PRML in context of other web design methods and situations of personalization. Also a complete study of the integration of a rule engine component with the development platform is intended.

7. References

[1] Blaze Advisor. http://www.blazesoft.com/product/advisor

[2] Brickley, D., Guha, R..: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004

[3] Casteleyn, S., De Troyer, O., Brockmans, S.: "Design Time Support for Adaptive Behaviour in Web Sites", In Proceedings of the 18th ACM Symposium on Applied Computing, ISBN 1-58113-624-2, Melbourne, USA (2003), pp. 1222 – 1228.

[4] Casteleyn S., Garrigós I., De Troyer O.: "Automatic Runtime Validation and Correction of the Navigational Design of Web Sites", In Web Technologies Research and Development - APWeb 2005. Springer-Verlag, ISBN 3-540-25207-X, LNCS 3399. Shangai, China (2005), pp 453-463.

[5] Dayal U.: "Active Database Management Systems", In Proc. 3rd Int. Conf on Data and Knowledge Bases, pp 150–169, 1988

[6] Facca F. M., Ceri S., Armani J. and Demaldé V., Building Reactive Web Applications. Poster at WWW2005, Chiba, Japan (2005).

[7] Garrigós, I. ,Casteleyn, S., Gómez, J.: "A Structured Approach to Personalize Websites using the OO-H Personalization Framework". In Web Technologies Research and Development - APWeb 2005. Springer-Verlag, ISBN 3-540-25207-X, LNCS 3399. Shangai, China (2005), pp 695-705.

[8] Gómez, J., Cachero, C., and Pastor, O.: Conceptual Modelling of Device-Independent Web Applications, IEEE Multimedia Special Issue on Web Engineering (2001) pp 26-39.

[9] Houben, G.J., Frasincar, F., Barna, P, and Vdovjak, R.: Modeling User Input and Hypermedia Dynamics in Hera (ed.): International Conference on Web Engineering (ICWE 2004), Lecture Notes in Computer Science, Vol. 3140, Springer-Verlag, Munich(2004) pp 60-73.

[10] ILog JRules,http://www.ilog.fr/products/jrules/features.cfm

[11] Kappel G., Retschitzegger W.,Kimmerstorfer E., Proll B, Schwinger W. & Hofer Th. "Towards a Generic Customization Model for Ubiquitous Web Applications". Proceedings of the 2nd International Workshop on Web Oriented Software Technology (IWWOST), in conjunction

with the 16th European conference on Object-Oriented Programming (ECOOP), Malaga, Spain, June 2002

[12] Koch, N.: “Software Engineering for Adaptive Hypermedia Systems, Reference Model, Modeling Techniques and Development Process”, PhD Thesis, 2001

[13] OpenRDF, The SeRQL query language, rev. 1.1, url: http://www.openrdf.org/doc/users/ch06.html

[14] Rik Gerrits, "Business Rules, Can They Be Re-used?" Business Rules Journal, Vol. 5, No. 9, URL: http://www.BRCommunity.com/a2004/b203.html, 20 04

[15] Schwabe, D. and Rossi, G. A Conference Review System with OOHDM. In First Internacional Workshop on Web-Oriented Software Technology, 05 2001.

[16] Simple Rule Markup Language, http://xml.coverpages.org/srml.html

[17] UML 2.0 OCL specification www.omg.org/docs/ptc/03-10-14.pdf

[18] Vdovjak R., Frasincar F., Houben G.J., and Barna P.: “Engineering Semantic Web Information Systems in Hera” Journal of Web Engineering (JWE), Volume 2, Number 1-2, pages 3-26, Rinton Press, 2003