Recovering Conceptual Models from Web Applications

Giuseppe Antonio Di Lucca
Research Centre On Software Technology, University of Sannio
Via Traiano, Palazzo ex Poste
82100 Benevento, Italy
dilucca@unisannio.it

Damiano Distante
Research Centre On Software Technology, University of Sannio
Via Traiano, Palazzo ex Poste
82100 Benevento, Italy
distante@unisannio.it

Mario Luca Bernardi
Research Centre On Software Technology, University of Sannio
Via Traiano, Palazzo ex Poste
82100 Benevento, Italy
mlbernar@unisannio.it

ABSTRACT

This paper proposes a reverse engineering approach for abstracting conceptual user-centered models from existing Web applications to re-document them at a high level of abstraction and from a user perspective.

The recovered models are specified by referring to the Ubiquitous Web Application (UWA) design methodology. UWA models are able to describe the structure of the application contents, the semantic relations among contents, the different views on contents the application offers to users, and the navigation paths and the navigation nodes used to present contents to users.

The approach exploits existing reverse engineering methods and tools to extract fine grained structural information from the analyzed applications and abstracts UWA models from them.

The architecture of a tool to support the reverse engineering approach is described and the results from some preliminary experiments are discussed.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – Documentation; Restructuring, reverse engineering, and reengineering.

General Terms

Documentation, Design.

Keywords

Reverse Engineering, Web Applications, Documentation, Conceptual modeling, Design, UWA.

1. INTRODUCTION

In the last years we have seen a wide diffusion of Web Applications (WAs) that are requested to provide and support functionalities and services more and more complex. WAs have therefore evolved from the earlier "simple" applications allowing some e-commerce functionality to complex software systems providing their users with the access to/management of complex services. Requirements for these applications take also into account the growing number of increasingly exacting users asking for high quality applications.

To face the challenges of building such types of WAs, several methodologies have been proposed in the literature. These methodologies support the disciplined design, development, maintenance and evolution of WAs [1]-[5][10][11].

Unfortunately, mainly due to the pressure of short time-to-market and extremely high competition, such development methodologies, as well as good software engineering principles, usually are not applied in the practice. As a consequence, analysis and design documentation in existing WAs is very poor, if not completely absent.

WAs are also characterized by continuous maintenance and evolution operations to meet new functional and not functional requirements of the evolving context in which WAs are used. For example, new requirements may derive from the need to meet some new business rules, the need to adopt new technology, as well as the need to implement some ad-hoc functionality.

Very often such a maintenance and evolution interventions are just implemented without being documented. Most of the times, the reasons behind this way to proceed are found again in time-to-market constraints but also in the relative quickness in which WAs code can be modified and new versions of the application deployed. When for a given WA some design documentation is available, this erroneous way to proceed causes its degradation and misalignment compared to the application implementation state.

The lack of adequate and up-to-date documentation makes the maintenance and evolution of a given system becoming a difficult and risky task, potentially compromising the effectiveness and correctness of the whole system. In this common situation the usage of tools and approaches for the semi-automatic recovery of models and documentation from the system to maintain has been proved to be very useful, reducing the effort and risks of the task to be executed.

Several approaches and tools for the reverse engineering of WAs have been proposed in the literature. Some of them aim at obtaining an architectural view of the WA that depicts WA components (i.e., pages, or inner page components) and their relationships at different levels of detail [12][13][14]. Some others [15] allow abstracting a description of the functional requirements implemented by the WA which is cast into UML use

SIGDOC'06, October 18–20, 2006, Myrtle Beach, South Carolina, USA.

case diagrams [16]. Some others else [17][18] recover UML class diagrams [16] of the application business objects and the logical relationships between them or models of the business processes implemented by the WA [7][8][9].

When executing a maintenance or evolution task it might be highly beneficial to have user-centered conceptual models of the WA. Such models, indeed, enable the maintainer to move from the user perspective on the WA when deciding on some functionality or some characteristics to be added, removed or improved. The conceptual models of a WA are also useful when migrating it towards different technologies. Indeed, as these models are independent from implementation and technological aspects, they are suitable for being implemented in any possible technology.

WA conceptual models may include the design of: (*i*) the contents provided to the user and the relations among them; (*ii*) the navigational paths the user can follow and the information nodes the user is presented; (*iii*) the operations the user can execute and the workflows implemented to assist the user in executing a task in a certain business process.

Most of WA reverse engineering methods and tools proposed in the literature enable the recovery of models at a low level of abstraction (often in terms of pages and page components), and aren't able to describe the application from the user-perspective. Also the results of these tools offer partial views on the application (navigation structure, business objects, business process, etc.) and use different representations and meanings.

In this paper we propose an approach to reverse engineering conceptual models from existing WA by referring to the Ubiquitous Web Application (UWA) design methodology [2][3]. The approach builds on existing WA reverse engineering methods and exploits existing tools, using the information they recover to abstract UWA models. Since most of the analysis and abstraction process involve the HTML client pages that the application presents to the user, the method is applicable to most of the existing WA.

The paper is structured as follows. Section 2 synthetically describes the UWA design methodology and its main models and design concepts with some examples. Section 3 presents the approach for recovering the considered UWA models from the code of an existing WA. Section 4 describes the overall architecture of a tool that we are developing to support the reverse engineering process. Finally Section 5 briefly reports on some preliminary experiments and introduces future work.

2. THE UWA DESIGN METHODOLOGY: MODELS AND DESIGN CONCEPTS

The UWA design framework includes a set of methodologies, meta-models, and tools for the user-centered design of data and operation intensive ubiquitous (i.e., multi-channel, multi-user and generally context-aware) WAs. UWA models provide separated but related views on the design of a WA. Each UWA model is focused on a different aspect characterizing a WA and uses a UML-based notation.

UWA models are conceptual user-centered models, i.e., they specify from a user perspective design aspects meaningful for the users of the WA. All the different types of users of the WA are considered during the design process and, if necessary, different model views are realized for each of them.

The typical UWA design process is structured into four main activities: 1) Requirement Elicitation, 2) Hypermedia and Operation Design (accomplished by using the W2000 [1] design methodology), 3) Transaction Design, and 4) Customization Design.

In this paper the focus is on the UWA Hypermedia Model which results from the UWA Hypermedia Design activity [19] and includes the Information model, the Navigation model and the Publishing model of the WA.

Figure 1 reports an excerpt of the UWA Hypermedia meta-model that highlights the Information meta-model and the Navigation meta-model.

The meta-model is represented by means of the UML MOF [20].

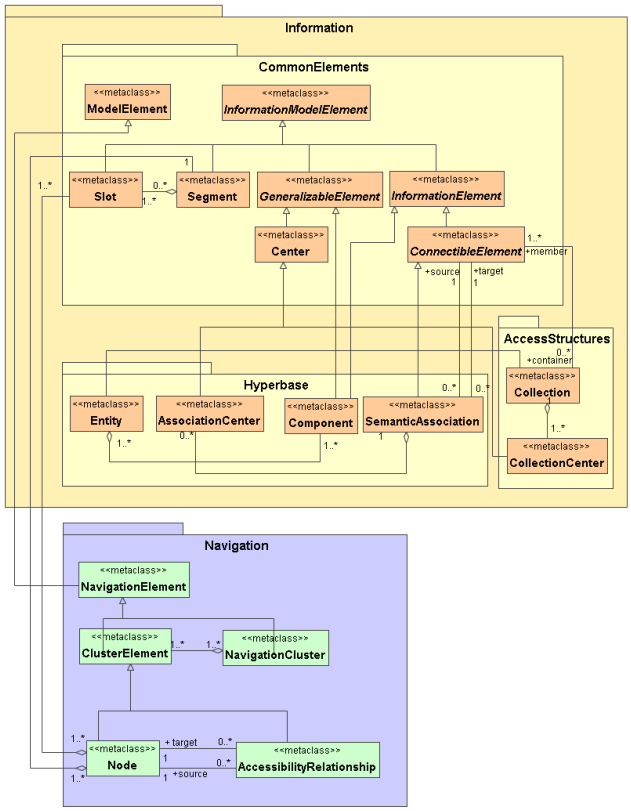

Figure 1 An excerpt of the UWA Hypermedia meta-model.

The UWA Information Model is composed of the Hyperbase and the Access Structures of the WA. The Hyperbase describes the structure of the WA contents and the relations among them in terms of Entities and Semantic Associations. The Access Structure defines the set of Collections that represent views of interest for the user on the Hyperbase.

An Entity is related to an object or a concept of the "real world" making sense for the user. Entities are the fundamental information elements defined by UWA. An Entity is organized into Components (in the same sense as books are organized into chapters). A Component holds all the content that corresponds to a "chapter" of the Entity. Components can be further decomposed in Sub-Components and Sub-Components in Base Components. A

Base Component is finally defined in terms of Slots, which are the elementary information element of the UWA Information Model.

Figure 2 shows the portion of a page from the GAP[1] e-commerce clothing Web site presenting a product in the catalog. The information characterizing each product, as seen by the user, includes the product name, the product code, the product textual description, a small and a large picture of it, the different variation (versions) in which it is available, and the price.

Figure 2 A portion of a page showing a product at GAP.com

For each version of the product (i.e.; regular, tall), the set of available sizes and colors are shown. Additional information, such as suggestions on how to wash and treat the product (fabric & care), might also be available. The UWA Entity diagram modeling the product information is shown in Figure 3. In addition to describing the set of elementary granules of information characterizing each product (referred as Slots in UWA), it represents the organization of these Slots into Components (Overview, Fabric & Care, and Version), the cardinality of each Slot and Component, as well as the minimum, maximum and typical number of instances of the Entity Product in the catalog.

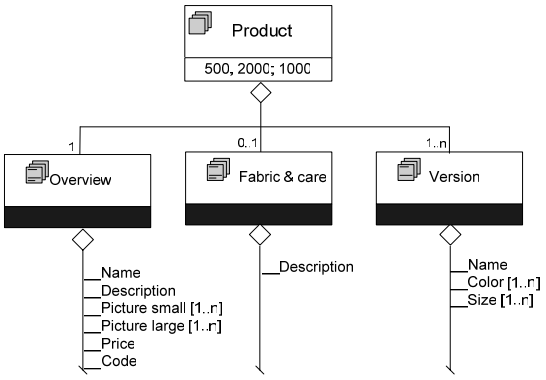

Figure 3 The UWA Entity diagram corresponding to the Product information object at GAP.com

Entities may be related to each-other by means of Semantic Associations. A Semantic Association connects two different Entities and has a semantic associated to it; moreover it creates the "infrastructure" for a possible navigation path.

To refer to the GAP Web site, the hyperlink named "you'll also like" appearing on the side of the product shown in Figure 2 brings the user to the page shown in Figure 4.

Besides presenting an excerpt of the information associated to the original product, this page also reports a list of products previews. These are the previews of the products suggested as good match with the original product. The user will be able to navigate from each of the product previews to a page similar to the one shown in Figure 2 providing detailed information on the chosen product.

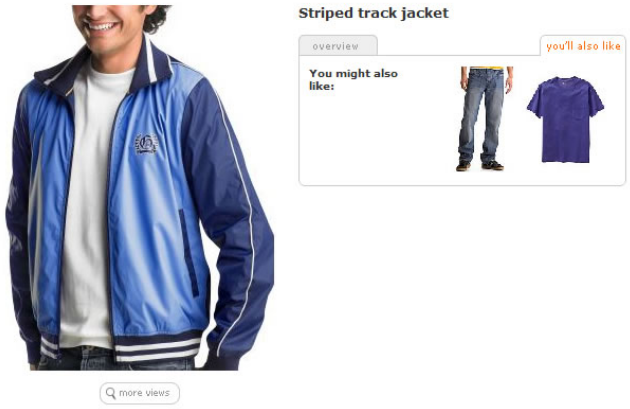

Figure 4 A portion of the page showing a product and the list of products suggested as a good match with it (associations "you'll also like") at GAP.com

Figure 5 represents the UWA Semantic Association diagram modeling the association between a product of the GAP catalog and the products that match well to it. The Center of the Association identifies the preview information provided for each of the matching products.

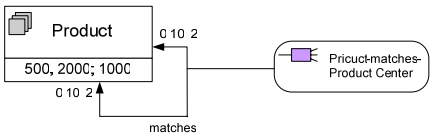

Figure 5 The UWA Semantic Association diagram corresponding to the association "you'll also like" between products at GAP.com

Entities and Semantic Associations may take part in Collections. A Collection is an organized set of objects, called members, created in order to assemble together all the objects that, under certain circumstances, can be interesting for the user. Objects may be Entity instances and/or their Semantic Associations.

Figure 6 shows one of the collections of products available on the GAP Web site. In particular the ones shown are the products of the category "Jacket" in the department "Outerwear". Each product of the collection is presented with some preview information (including the small picture, the name and price) and a link connecting to the page providing full description of the product.

[1] www.gap.com – accessed on May 2006.

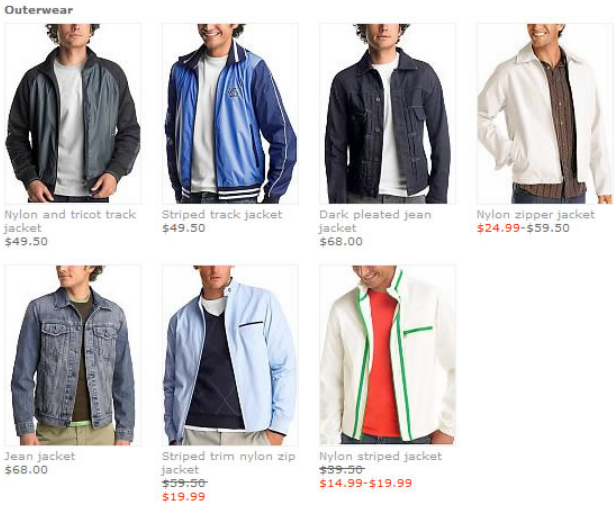

Figure 6 A portion of the page showing a collection of products belonging to the Outwear category at GAP.com

The UWA Collection diagram associated to the collection of products in Figure 6 is shown in Figure 7. Besides showing the Entity Types taking part in the collection and the information making part of the Collection Center, the diagram also specifies the cardinality of the members of the collection and the cardinality (min, max, expected) of the Collection.

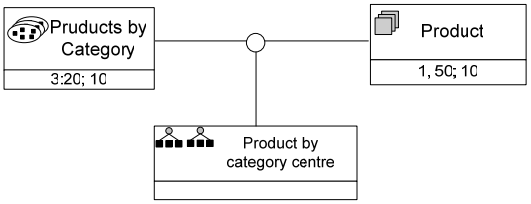

Figure 7 The UWA Collection diagram corresponding to the collection of Products of a given category at GAP.com

The UWA Navigation Model is made up of Nodes and Clusters. A Node defines a unit of information delivered as a whole by the application to the user from/to which the users can navigate. A Node may be derived from either: (*i*) an Entity, (*ii*) a Semantic Association, (*iii*) a Collection Center. A Cluster is a container holding together a number of Nodes and defining the navigation pattern among the interconnected Nodes (i.e.; guided tour, index, etc.). A default Node is defined for each Cluster, specifying where the navigation should start when the Cluster is accessed. UWA defines three different categories of Clusters: Structural Clusters (containing Nodes that are pieces of the same "object"), Association Clusters (containing nodes related together via a Semantic Association), and Collection Clusters (containing Nodes that are parts of a Collection).

Figure 8 shows the UWA Cluster diagram that models the structural navigation between the Components of the Entity product in the GAP Web site. The Node "Product Overview" is the default Node of the Cluster. From this Node it is possible to navigate towards the Nodes showing the Slots included in the other two Components of the Entity "Product".

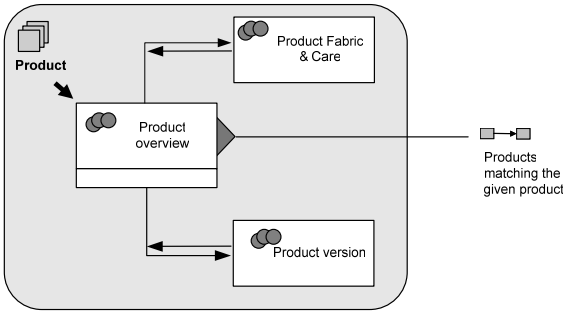

Figure 8 The UWA Cluster diagram describing the navigations between the Components of the Entity 'product' at GAP.com

Each of the UWA design concepts can appear in an application in two ways: as an instance of "Type" or as one-of-a-kind, i.e., "Single". A "Type" defines a "categories of objects" sharing some common properties that are defined by the type. A "Single" defines an object that does not share its properties with any other object, thus having cardinality 1 in its instances. The Entity "Product" and the other UWA concepts found for the GAP Web site and described above are all instances of "Types" or also "Typed" concepts. On the opposite, the Entity "About GAP Inc." which provides information on the GAP Company is an example of Single Entity. In the following of the paper, unless otherwise specified, we'll refer to UWA concept types omitting the "Type" suffix.

3. RECOVERING UWA MODELS FROM EXISTING WEB APPLICATIONS

3.1 The Reverse Engineering Process

A reverse engineering process to recover the concepts included in the UWA Information and Navigation Model from the code of undocumented existing WAs has been defined. The process is based on the static analysis of the WA pages and is structured as follows:

1. Identification of Entity Types
2. Identification of Semantic Association Types
3. Identification of Collection Types
4. Identification of Node Types and Cluster Types

The methods to fulfill each step of the process are described in the following sub-sections.

3.2 Identifying UWA Entities and Semantic Associations

Entities and Associations represent, respectively, the relevant domain business object processed by the WA and the relationships among them. A UWA Entity is a group of related attributes (Slots) making up a more abstract concept. Slots may be grouped into Components in turn. Techniques searching the source code for groups of logically related data representing the attributes of Classes can be used to identify Entities. These techniques are usually based on those language mechanisms that allow groups of related data implementing a relevant concept, either from the domain of the application or from domain of the solution, to be defined in the code. These mechanisms include

those for the definition and use of data structures such as records, user data types, and table schemas in databases, or the state of objects.

Although most web technologies and languages (such as HTML, ASP, PHP, VBS, JSP, etc.) provide syntactic constructs for declaring data groups (such as RecordSets, Collections, and Classes), they are not used frequently. Moreover, since a WA is usually implemented as a multi-tier system, a simple WA code analysis may not allow the identification of the persistent data stores and the data store schemas because they can be deployed on a different tier of the application, and may be inaccessible.

The method proposed in [17] and [18] to identify the attributes of a WA business objects has been exploited to identify UWA Entities. The method is used to identify the attributes (Slots) making up an Entity. In the following we summarize the way the attributes are identified.

We consider as candidate to form the attributes of an Entity the groups of data items that are involved in: (*i*) the same user input/output operation (such as data displayed in input/output HTML forms, or HTML tables); (*ii*) in the same read/write operation on a data store (such as an ASP Recordset, or an array of heterogeneous data in PhP language), (*iii*) the data set involved in a database query operation. In addition, data groups that are passed throw distinct pages or instances of Classes used in the pages are taken into account.

The rationale behind this choice is that the set of data items that a user enters in an input form, or that are shown to a user by an output report, usually represents a concept of interest for the user in the domain of the application. Analogously, data items that are read from, or written to a persistent data store may be representative of meaningful concepts of the business domain.

Groups of related data items are found also by exploiting the method proposed in [21] and [22] to identify groups of cloned web client pages. Each group of cloned pages is characterized by the same *control component* (i.e., the set of items - such as the HTML code and scripts - determining the page layout, business rule processing, and event management), but a different *data component* (i.e., the set of items - such as text, images, multimedia objects - determining the information to be read/displayed from/to a user). Groups of pages having the same control component will show the same rendering and functional behavior. Thus they can be considered as equivalent pages (i.e. clones), just differing for the data component they contain. The data component of each group of cloned pages is then analyzed to identify groups of common data items contained in each page of that group. These items usually correspond to labels of fields, or table headings, showing the values of some attributes of domain Entities. Thus the groups of common data items identified for each group of cloned pages can be considered as attributes of candidates Entities.

After having solved the problem of *synonyms* (*i.e.*, attributes identifiers with different names but same meaning) and *homonyms* (*i.e.*, attributes identifiers with the same name but different meanings), the recovered data groups are automatically analyzed in order to identify the ones that are more likely to be associated to Entities. The analysis is based on two heuristic rules: (*i*) the more the references of a same data group in the code, the greater the likelihood that it represents a meaningful concept; (*ii*) groups with a small size may represent more simple and atomic concepts than larger groups, and larger groups may represent more complex concepts made up of joined smaller groups.

An automatic procedure organizes the groups of data into a list and ranks each group according to some criteria (such as the number of occurrences of the groups) as well as re-organizes the groups themselves (such as when a group is included into a larger one) producing an ordered list including the set of data groups that have been selected as candidate Entity. Each group in this list will have to be assigned with a meaningful name describing the concept it represents, i.e. the Entity. Each data item will correspond to a Slot of an Entity and each sub-group of data items will be candidate to make up a Component. If a group including one or more other groups presents more references than the included groups, just the container group will be considered as a candidate Entity, while the smaller ones will not (this can be considered as a Component of the Entity, and its attributes as Base Components).

A Semantic Association will be identified for each set of candidate Entities having common attributes. Each common attribute (Slot) will be assigned just to one Entity of the set, and all these Entities will be linked by a UWA Association. The software engineer intervention may be required to establish the correct assignment of the attributes to the Entity.

Moreover, Semantic Associations are identified by analyzing the content of forms, tables, and others reports displayed to users: if attributes from different Entities are required by a form or displayed in a report, an Association will be considered to exist between those Entities. For example, for an e-commerce application, this would be the case when a Customer has to fill in a form with some of his/her personal data, and data of the Products he/she wants to buy. In this case, a Semantic Association named ‘Purchase’ between the ‘Customer’ and ‘Product’ Entities can be identified.

A validation of the candidate Entities and Associations found in this step has to be carried out before proceeding to the next step. Of course knowledge of the application domain would help the needed validation activity.

A cross reference list is generated to trace the pages where each identified Entity/Association (i.e., their attributes) was found. The cross reference list shows: the names of the identified Entities and Associations, their attributes, the name of the pages where each Entity/Association is referred, the name of the Slots referred in each page. In the case of static client page (i.e. web pages whose content is fixed and stored in a file on the web server), the name of the web page is the name of file storing the page content; For dynamic client pages (i.e. client pages generated at run time by server pages), the page name corresponds to the name of the server page generating that client page.

3.3 Identifying Collections

UWA Collections are organized sets of Entity or Association instances (Collection members) representing views of interest for a user. A Collection lists a sub-set of attributes for each member, and usually provides an index (e.g., a list of hyperlinks) to access

more detailed member information. A UWA Collection Centre is associated to the index of the Collection. Examples of Collections are: the list of products in a category in an e-commerce Web site, a list of documents to browse in an electronic library.

The identification of Collections is based on the ways usually used to implement them. These include: (*i*) the usage of forms reporting a list of fields with data related to the attributes of an Entity or the Entities involved in an Association; (*ii*) the usage of a table where each column reports the values of an attribute of an Entity or an Association (each row of the table represents an instance of the Entity/Association); (*iii*) a list of hyperlinks to pages reporting more detailed information about an Entity or an Association; (*iv*) full text reporting a list of fields with the values of an Entity or an Association. The HTML code of web client pages is analyzed to identify Collection and Collection Centre. In the case of dynamic client pages (i.e., client pages generated at run time by server pages) these are 'captured' and stored to analyze the generated code making up the pages.

First the pages referring Entity/Association attributes are selected. The code of these pages is analyzed to verify the presence of those structures (forms, table, list of hyperlinks, etc.) that could implement a Collection, or a Collection Center. In particular, the code is searched for identifying: (*i*) forms with repeated fields having the same labels corresponding to Entity/Association attributes; (*ii*) tables with a header made up of Entity/Association attributes; (*iii*) tables whose cells contain words/labels corresponding to Entity/Association attributes; (*iv*) sequences of hyperlinks whose labels are Entity/Association attributes. Clone analysis is useful in looking for Collections too. In this case the aim is to find cloned pieces of HTML code repeated in the page to show the same structure with different values of an Entity/Association, e.g. the row of a table, or the line of a bullet list containing Entity/Association attributes repeated more times in the page.

The identification of Web Interaction Design Patterns (WIDP) in web pages can be useful to identify Collections. WIDPs provide a solution to classical interaction problems in WAs. Specific WIDPs have been proposed in the literature [23]. The WIDPs belonging to the 'Managing Collections' class (e.g. View, Collector, List Builder, List Sorter, Table Filter, Table Sorter, Parts Selectors) are possible ways to implement Collections. In [24] an approach to support the automatic identification of WIDPs in a web page is proposed. This method can be used to verify in a page the presence of a pattern implementing a Collection.

All the above described techniques have been exploited to get a more precise identification of Collections and to reduce the number of false positives. Problems arise when a Collection is implemented by a plain full text: in this case the above techniques may not be fully adequate.

A Collection Center is identified by searching for a list of hyperlinks in a page, where all the hyperlinks point to: (*i*) the same server page (but passing each time some different parameter values) that dynamically builds a client page whose contents report the required information; (*ii*) to different static client pages forming a set of cloned pages.

This step will result in a list of candidate Collections, and Collection Centers, that have to be validated by the analyst. Each Collection/Collection Center will be assigned a meaningful name to univocally identify it.

Also in this step a cross reference list reporting the identified Collections and the client pages referring them is produced.

3.4 Identifying Nodes and Clusters

A Node is a unit of information that an application will provide or ask to a user as a whole. The information contained in a Node may refer to an Entity/Association, or a portion of them if just some of their attributes are reported in the Node, as well as to a Collection.

Nodes are identified by associating them to the elements of client page displaying/requiring information from/to the user.

Also in this case, first the client pages related to Entities, Associations, and Collections are selected. Each page is then analyzed to: (*i*) identify which attributes of each Entities, Associations, or Collections are referred in the page; (*ii*) associate a Node to each group of attributes (if the same group of attributes is present in different pages, they will be associated to a unique Node modeling them); (*iii*) identify hyperlinks connecting Nodes in the same page or in different pages. If a page contains more than one Node, a Cluster of Nodes will be defined. A Cluster is also defined when a hyperlink exists between two Nodes (in the same page or in different pages).

A list of candidates Nodes and Clusters is the result of this step. Each Node and each Cluster will be assigned a meaningful name.

4. A TOOL TO SUPPORT THE REVERSE ENGINEERING PROCESS

In order to support the activities of the proposed approach, a RE tool, called RE-UWA, is under development. It aims to provide software engineers with a useful and extensible environment supporting WAs re-documentation, comprehension, maintenance and testing tasks.

Figure 9 shows the overall layered architecture of the RE-UWA tool.

At the lower layer we find the Web Application Reverse Engineering (WARE) tool [13] and Clone Detector [22] tool. WARE statically analyzes the source code of the WA and extracts notable information about: (*i*) the pages making up the WA; (*ii*) the inner component of each page (e.g., forms, scripts module, frame, applet, etc.); (*iii*) the different types of hyperlinks connecting the pages (e.g., link, build, submit, redirect). The extracted information are stored into the WARE Repository and used to abstract design and analysis documents represented by UML diagrams according to Conallen's UML extensions [11]. The WARE tool includes modules to support the abstraction of UML class diagrams at the conceptual business level [17][18] as well as modules to identify Web Interaction Patterns [24]. Also the abstracted information is stored in the WARE Repository.

The Clone Detector tool statically analyzes the web client pages (Web client pages dynamically built from server pages are 'captured' on the fly and stored to be analyzed successively) to identify cloned pages according to the approach proposed in [21] and [22].

The RE-UWA Abstractor Layer includes the components to abstract the UWA Information and Navigation Model concepts according to the process described in the Section 3. The WARE Importer component extracts the needed information from the WARE Repository and arranges them in a format suitable for the Abstractor modules. The Information Model Concepts Abstractor is responsible for recovering candidate UWA elements of the Hyperbase and the Access Structure Models (e.g., candidate Entities, Associations and Collections). The Navigation Model Concepts Abstractor is responsible for recovering the elements of the Navigation Model (i.e., candidates Nodes and Clusters). The Concepts Validator module mainly implements a user interface allowing the browsing of the recovered candidate elements. By using this module the analyst will be able to decide on if to accept, make some modification before accepting, or reject the recovered candidate UWA elements. The validated recovered UWA concepts are stored into the Metadata Repository.

The top layer of the RE-UWA architecture is the UWA tool set developed as Rational Rose add-in to support the UWA methodology. This tool is able to create and manage UWA models using stereotyped UML diagrams within the Rose environment [25]. The tool includes an Import/Export component which is responsible for importing/exporting UWA models in an XMI format.

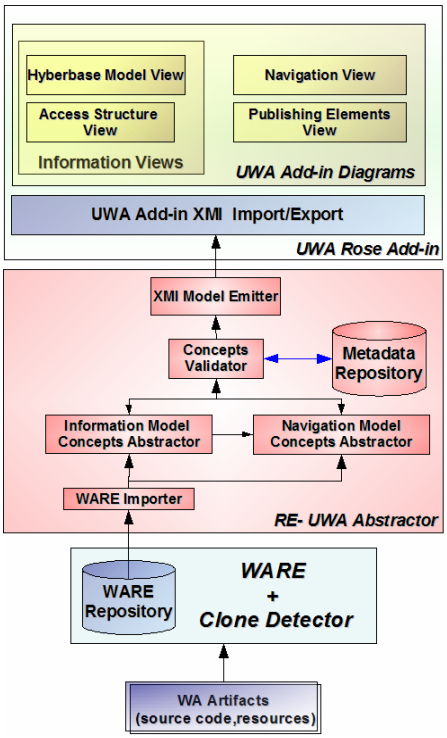

Figure 9 The overall architecture of the RE-UWA tool.

All the UWA add-in modules share the following common structure:

- Custom menus that send messages to UWA COM objects.
- Extensible properties. The UWA model elements can be extended with additional properties (UML tagged-values).
- Stereotypes (along with visual customization).
- Events (COM Server). Each add-in registers for a set of events where a method on a UWA COM object is called.
- Automatic documentation report generation.
- XMI-based persistence to store and load UWA model instances.

The RE-UWA Model Emitter is in charge of generating the UWA Information and Navigation Models of the analyzed WA according to the UWA Rose Add-in specified XMI format.

The RE-UWA Abstractor layer has been designed to provide a generic abstractor API, supporting interoperability to a great extent. Hence more intermediate representations targeting many others RE environments can be supported in the future.

At current time a prototype of the WARE Importer and the Information and Navigation Abstractor modules have been developed.

5. RESULTS AND CONCLUSIONS

Some preliminary experiments were carried out to asses the feasibility and the effectiveness of the proposed approach.

In a first experiment, some simply web pages were developed: each page implemented just one of the UWA concepts. The pages were developed by undergraduate students and different techniques were used to implement a same concept (in particular to implement Collections). The approach showed to be successful in identifying all the UWA concepts. It is worthwhile to note that the technique based on the identification of cloned pages showed to be better to identify concepts (in particular Entities and Collection) when just plain text were used in client pages to implement them.

A second experiment considered a complete, small size, WA developed by graduate students. The application allows users to make predictions about football matches. Also in this case the approach proved its effectiveness by identifying correctly the several Entities (e.g., User, FootBallTeam, Bet), Associations (e.g., FootBallMatch, UserBet), Collections (e.g., MatchesOfTheDay, TeamRanking), Nodes and Clusters. In this case the analysis of the statements accessing the data base in the server pages allowed a more precise identification of the Entities and Associations.

Finally a third experiment considered a large number of client pages downloaded from some web sites available on the net. The pages of each web site were analyzed and for each site the UWA concepts recovered. The approach showed a lower precision in identifying Entities and Associations in those Web sites where both pages with forms and pages with a structure reporting explicit labels were lacking. In this case the cloned pages based method was found to be more effective in identifying concepts implemented by just plain text.

5.1 Conclusions

The results of the conducted experiments were satisfactory and encouraging: they showed the feasibility of the approach and a good effectiveness.

The definition of alternative criteria to identify UWA Entities as well as the refinement of the used ones is expected to improve both the effectiveness and precision of the approach. Further experiments will help in this direction.

Future work will be devoted to complete the development of the RE-UWA tool and to extend the RE approach itself to recover the UWA Publishing Model (which describes how Nodes and Clusters are arranged into WA pages) and the UWA Transaction Model (which models the business processes implemented by the WA).

6. REFERENCES

[1] L. Baresi, F. Garzotto, P. Paolini, "Extending UML for Modeling Web Applications." In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences* (HICSS-34: Honolulu, HI, 2001). Los Alamitos, CA: IEEE CS Press.

[2] UWA Project Consortium, "Ubiquitous Web Applications". In *Proceedings of the eBusiness and eWork Conference 2002*, (e2002: 16-18 October 2002; Prague, Czech Republic).

[3] UWA Project Consortium, "Deliverable D2: General Definition of the UWA Framework." 2001.

[4] D. Schwabe, and G. Rossi, "An Object-Oriented Approach to Web-Based Application Design." *Theory and Practice of Object Systems* (TAPOS), Vol 4 (1998) 207-225.

[5] A. Bangio, S. Ceri, P. Fraternali, "Web Modeling Language (WebML): a modeling language for designing Web sites". In *Proceedings of the 9th International Conference on the WWW (WWW9)* - 2000. Elsevier: Amsterdam, Holland, 2000: 137-157.

[6] N. Koch, A. Kraus, C. Cachero, and S. Meliá, "Modeling Web Business Processes with OO-H and UWE." In *Proceedings of the 3rd International Workshop on Web Oriented Software Technology* (IWWOST 2003: July 15, 2003, Oviedo, Spain).

[7] D. Distante, T. Parveen, and S. Tilley, "Towards a Technique for Reverse Engineering Web Transactions from a User's Perspective". In *Proceedings of the 12th International Workshop on Program Comprehension* (IWPC 2004: June 24-26, 2004; Bari, Italy). Los Alamitos, CA: IEEE Computer Society Press, 2004, pp. 142 - 150.

[8] D. Distante, S. Tilley, and S. Huang, "Documenting Software Systems with Views IV: Documenting Web Transaction Design with UWAT+." In *Proceedings of the 22nd International Conference on Design of Communication* (SIGDOC 2004: October 10-13, 2004; Memphis, TN). ACM Press: New York, NY, 2004, pp. 33-40.

[9] S. Tilley, D. Distante, and S. Huang, "Design Recovery of Web Application Transactions." In *Advances in Software Evolution with UML and XML* (Editor: Hongji Yang). Hershey, PA: Idea Group Publishing, May 2005.

[10] A. Knight, N. Dai, "Objects and the Web", *IEEE Software*, Mar-Apr 2002, pp. 51- 59.

[11] J. Conallen, "Building Web Applications with UML – 2nd Edition". Addison Wesley Publishing Company: Reading, MA, 2002.

[12] S. Chung, Y.S. Lee, "Reverse software engineering with UML for web site maintenance". In *Proceedings of 1st International Conference on Web Information Systems Engineering*, 2001, IEEE CS Press, Los Alamitos, CA, (2): 157-161

[13] G. A. Di Lucca, A.R. Fasolino, U. De Carlini, F. Pace, P. Tramontana, "WARE: a tool for the Reverse Engineering of web Applications", In *Proceedings of 6th European Conference on Software Maintenance and Reengineering*, Mar. 2002, IEEE CS Press, pp. 241-250.

[14] F. Ricca, P. Tonella, "Understanding and Restructuring Web Sites with ReWeb", *IEEE Multimedia*, 2001, 8(2): 40-51.

[15] G. A. Di Lucca, A.R. Fasolino, U. De Carlini, F. Pace, P. Tramontana, "Comprehending Web Applications by a Clustering Based Approach". In *Proceedings of 10th IEEE Workshop on Program Comprehension*, IWPC 2002, IEEE CS Press, pp. 261-270.

[16] Object Management Group (OMG). Unified Language Modeling Specification (Version 2.0). Online at www.omg.org. 2004.

[17] G. A. Di Lucca, A.R. Fasolino, U. De Carlini, P. Tramontana, "Recovering a Business Object Model from Web Applications". In *Proceedings of 27th IEEE Annual International Computer Software and Applications Conference* (COMPSAC 2003), Dallas, USA, November, 2003, IEEE Comp. Soc. Press, Los Alamitos, California.

[18] G. A. Di Lucca, A.R. Fasolino, U. De Carlini, P. Tramontana, "Abstracting Business Level UML Diagrams from Web Applications". In *Proceedings of the 5th IEEE International Workshop on Web Site Evolution*. Amsterdam, The Netherlands, 22 Sept.. 2003, IEEE Comp. Soc. Press, Los Alamitos, California.

[19] UWA Project Consortium. Deliverable D7: Hypermedia and Operation design: model and tool architecture. 2001.

[20] UML Meta Object Facility (MOF) Core Specification OMG Available Specification Version 2.0 formal/06-01-01. www.uml.org. 2006.

[21] G. A. Di Lucca, M. Di Penta A. R. Fasolino, "An Approach to Identify Duplicated Web Pages". In *Proceedings of 26th IEEE Annual International Computer Software and Applications Conference* (COMPSAC 2002), Oxford, England, August 26-29, 2002, IEEE Comp. Soc. Press, Los Alamitos, California

[22] G. A. Di Lucca, A. R. Fasolino, P. Tramontana, U. De Carlini, "Identifying Reusable Components in Web Applications ", In *Proceedings of the IASTED International Conference on Software Engineering*, Innsbruck, Austria, February 17 - 19, 2004.

[23] Web Design patterns, http://www.welie.com/patterns.

[24] G. A. Di Lucca, A. R. Fasolino, P. Tramontana, "Recovering Interaction Design Patterns in Web Applications". In *Proceedings of the IEEE 9th European Conference on Software Maintenance and Reengineering*, Manchester, United Kingdom, 21-23 March 2005, IEEE Comp. Soc. Press, Los Alamitos, California.

[25] IBM Rational Rose Enterprise. http://www.ibm.com/rational.