International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:4, 2007

Multi-Dimensional Concerns Mining for Web Applications via Concept-Analysis

Carlo Bellettini, Alessandro Marchetto, and Andrea Trentini

***Abstract*—**Web applications have become very complex and crucial, especially when combined with areas such as CRM (Customer Relationship Management) and BPR (Business Process Reengineering), the scientific community has focused attention to Web applications design, development, analysis, and testing, by studying and proposing methodologies and tools. This paper proposes an approach to automatic multi-dimensional concern mining for Web Applications, based on concepts analysis, impact analysis, and token-based concern identification. This approach lets the user to analyse and traverse Web software relevant to a particular concern (concept, goal, purpose, etc.) via multi-dimensional separation of concerns, to document, understand and test Web applications. This technique was developed in the context of WAAT (Web Applications Analysis and Testing) project. A semi-automatic tool to support this technique is currently under development.

***Keywords*—**Concepts Analysis, Concerns Mining, Multi-Dimensional Separation of Concerns, Impact Analysis.

I. Introduction

WEB applications quality, reliability and functionality are important factors because software glitches could block entire businesses and determine strong embarrassments. These factors have increased the need for methodologies, tools and models to improve Web applications (design, analysis, testing, and so on).

This paper focuses on legacy Web applications where business logic is embedded into Web pages. Analyzed applications are composed by Web documents (static, active or dynamic) and Web objects [5]. This paper describes an approach to help application developers to document, understand and test Web software. Our goal is to describe a Web application Object-Oriented model, and then define a set of application/design slices ("points of view") to analyze and test the application itself, e.g., to generate a set of test cases specific for these points of view. Several Object-Oriented Web modeling methodologies are presented in literature (see Section II). Web OO diagrams (such as Conallen UML [12]) used to describe applications may be very complex, large, and rich of information. Models (above all generated ones) may be difficult to read and comprehend, so that they may not be much usable as core information to document, analyze and test applications. Our approach may be useful to slice or traverse models for software analysis. For example, it may be very interesting to test or reuse single components or tasks or properties, but it may be very complex to spot the relevant details within the whole design documentation. Software concerns are pieces of software that are responsible for a particular task, concept, goal, etc; while "separation of concerns" refers to the ability to identify, encapsulate and manipulate those software parts relevant to a particular concern.

This paper describes a semi-automatic approach to help the user to document, understand and test Web software by slicing applications diagrams. Application model slicing is based on concerns identification and grouping. Our approach describes a set of guidelines to analyze application evolution under different "points of view" (i.e., slices). In particular we would like to define a concern-mining process to help the user to generate application test cases and/or to verify their coverage measure. Our approach is useful to identify multi-dimensional concerns (MDSOC, [11],[18]]) in design applications, it uses the MDSOC "dimensions of Hyperspace" concept to describe application slices in Web software. "Hyperspace" is the concept underlying MDSOC, it provides a powerful composition mechanism that facilitates non-invasive software integration and adaptation. In Hyperspaces, concerns are space dimensions. Our concerns mining approach is based on: concepts analysis[1] [4] (as unit-base to identify concerns); impact analysis [22] (to limit software analysis); and token-based concerns identification (to search identified information relationship). This technique is part of the WAAT (Web Application Analysis and Testing) project [5],[6].

This paper is organized as follows. Section II describes related works. Section III describes applications modeling. Section IV describes our concerns mining approach. Section V presents a sample. Section VI presents conclusions.

Manuscript received January 27, 2005.

Carlo Bellettini, Alessandro Marchetto, and Andrea Trentini are with Information and Comunication Department, University of Milan. Via Comelico 39, 20135 Milan, Italy.

{Carlo.Bellettini, Alessandro.Marchetto, Andrea.Trentini}@unimi.it

[1] Concept analysis is "traditionally" used to show all possible software modularizations in a concise *lattice* structure

II. Related Works

Several Web applications modeling approach are presented in literature [5], the ones related to our concerns mining technique are the OO derived, such as Conallen UML WA-extensions[12], WARE[8], ReWeb[7], Rational Rose Web Modeler[20], WebUml[5], and so on.

More details about Aspect Oriented programming are in [9], while [3] presents the AspectJ famous software. [11],[18] describe the MDSOC and HyperJ tool, while [17] studies the relations between quality factors and MDSOC, while [13] the relations between MDSOC and testing. [2] describes SOC used to reduce the complexity of Web applications. [16] presents an approach to separate Web navigation concerns and application structure. [14] evaluates AOSD code quality influence and presents an approach for reverse engineering aspects, based on concern verification and aspect construction. [15] evaluates the suitability of clone detection as a technique for the identification of crosscutting concerns via manual concern identification. [1] introduces aspect mining and identification in OO. [21],[19] show an approach to aspect mining based on dynamic analysis technique via program traces investigation, to search recurring execution relations. [10] applies three different separation of concerns (SOC) mechanisms (HyperJ, AspectJ, and a lightweight lexically based approach) to separate features in the two software packages. This paper studies effects that various mechanisms have on code-base structure and on restructuring process required while performing separations.

III. Web Applications Modeling

In the WAAT project Web applications are modelled via UML diagrams. The UML model used is based on class and state diagrams. We have defined a UML meta-model [5], a Web application model is an instance of this meta-model. Class diagrams are used to describe application structure and components (i.e., forms, frames, Java applets, HTML input fields, session elements, cookies, scripts, and embedded objects). State diagrams are used to represent behaviour and navigational structures composed by client-server pages, navigation links, frames sets, form inputs, scripting code flow control, and so on. The OO application model let us define a mapping between traditional Web application concepts (such as static-dynamic pages, forms, Web objects, and so on) and the MDSOC concepts. This map let us apply separation of concerns methodologies in the Web context, for example to analyse or test specific assets of existing software. Our approach may be used to "slice" application models by "points of view".

IV. Concerns Definition Algorithm

MDSOC technique is used to build application slices, where every concern (or concerns composition) may be used to define a software code/design slice. MDSOC is realized through Hyperspaces: concerns space organized in multi-dimensional structure. In this structure every dimension is a set of disjoint concerns (i.e., they have no software units in common). We define a semi-automatic concerns mining approach, so concerns identification must be limited to information extracted from applications models or source code. For example, software functionality identification is a semi-automatic task, because the user helps to identify software components. We may lower user interactions by only applying MDSOC to concerns that are automatically extracted. When functionality information cannot be automatically identified, than we use: variables, functions, class, Web documents/objects, links, input-variables, and so on.

Our approach is composed by: **Application Modeling** (**AM**, model definition), **Concerns Elaboration** (**CE**, Hyperspaces definition through model and source code analysis, and Hyperspaces use to reduce models and code taken into account), **Testing** (**T**, the extracted and reduced information may be used to define/refine test cases).

Application Modeling (**AM**) consists in application model definition. We use reverse engineering techniques to define UML diagrams for existing applications. Moreover, diagrams may also be manually refined by the user.

Concerns Elaboration (**CE**) to identify, define, and extract concerns based on application model or source code analysis, subdivide in:

- *Artifacts extraction*: from application model we extract some interesting artifacts such as class, association, variables, methods, links, Web pages (e.g., static, dynamic, dynamically generated), objects (e.g., database, files, reused code), and so on. We use this knowledge to identify concerns (it may be a limitation, i.e., concerns about functionality cannot be completely defined without user know-how).
- *Objects-attributes selection*: from the selected artifacts we define "object-attribute"[2][4] couples to use in concept analysis. We may limit the number of couples by asking user help. Generally speaking, example of couples may be: variables-classes, functions-attributes, and so on.
- *Impact matrix definition*: from the application model we define a matrix I = [class x class]. $\forall\ i_{k,m} \in I = 1$ if $\exists$ class relationship (i.e. association in class diagram between $class_k$ and $class_m$), 0 otherwise. The matrix is then used to decrease analysis computational cost.
- *Context matrix definition*: for every couple defined we build an objects-attributes matrix C = [object x attribute]. $\forall\ c_{k,m} \in C = 1$ if there is a def-use relationship between $object_k$ and $attribute_m$, 0 otherwise.
- *Concept definition/visualization*: we define concepts through the C contexts matrix. We analyze this matrix grouping the maximum number of objects that have common attributes (by concept definition in concept-analysis). To visualize the defined concepts we may use the concept-lattice

[2] where "objects-attributes" is defined in concept-analysis theory

[4][23] structure.

- *Concerns definition*: we identify concerns by iteratively grouping previously defined concepts. To define concerns we may choose one of the following grouping strategies: "one concern=one concept", "one concern=one set of concepts", "one concern=one concept partition [4]", and so on. We currently use the "one concern=one set of concepts" one. We define a new "attributes-concepts" matrix[3] A = [attribute x concept]. $\forall$ $a_{k,m} \in A = 1$ if *attribute* is contained in *concept*. By recursively applying the "attributes-concepts" matrix, at each step we build supersets of concepts (grouping concepts that share attributes) that are used as concepts as well in the next step.

Testing (**T**): to define test cases on reduced information; to use reduced information to compute application coverage level for a set of already available test-cases. For example, we may define test cases from a UML model (e.g., from a statechart, see Section III) via traditional OO techniques and then use the reduced diagram to verify test-cases coverage (e.g., uniformly coverage or specialized one). Otherwise we may define test-cases directly from the reduced diagrams, because they represent sets of application features (software fragment with potentially independent behavior).

V. Sample

"*MiniLogin*" is a simple Web application composed by some PHP/HTML files, and its main functionality is to control reserved login-password Web area.

Application Modeling (**AM**): we reverse engineer the application UML model, composed by class and state diagrams. Figure 1: MiniLogin UML Class diagram shows the generated application class diagram (meta-model instance).

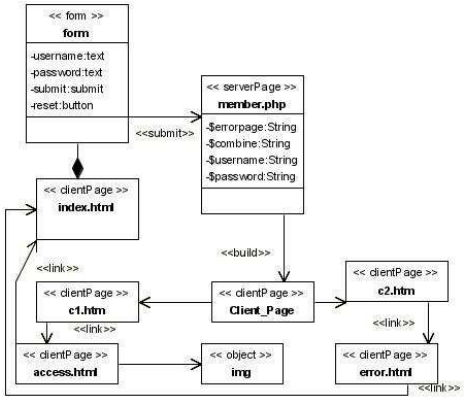

Figure 1: MiniLogin UML Class diagram

Concerns Elaboration (**CE**): defines MiniLogin concerns.

Artifacts extraction: we extract MiniLogin artifacts, lists of: classes, variables, functions, links, and so on.

- *Objects-attributes selection:* we manually select couples of objects attributes to use in concept analysis. E.g., variables-class (named "case-A"), variables-functions, and so on.
- *Impact matrix definition* (due to lack of space we exemplify only a couple of entries): "form" is related to "member.php", while "form" is not related to "C2.html".
- *Context matrix definition* (due to lack of space we exemplify only a couple of entries): for "case-A" "$errorpage" is related to "member.php", "username" is related to "form", and "username" is related to "member.php", while "username" is not related to "C2.htm"
- *Concept definition/visualization:* we use the context matrix to define concepts as defined in formal concept-analysis [4]. We may use existing tools (such as ToscanaJ [23], to define and visualize concepts through concept-lattice). Table I shows "case-A" concepts.

TABLE I
CASE-A, CONCEPTS

Concept	Object	Attribute
Top	*...all...*	-
C3	{username, password, $errorpage, $combine, $username, $password}	{member.php, Client_Page, c1.htm}
C2	{username, password, $combine, $username, $password }	{member.php, Client_Page, c2.htm, c1.htm }
C1	{username, password, submit, reset}	{form }
C0	{username, password}	{form, member.php, Client_Page, c1.htm, c2.htm }
Bottom	-	*...all...*

- *Concerns definition:* we identify concerns via concepts grouping. We build the attributes-concepts matrix, with attributes used (rows) and concepts (columns). A cell is = 1 if the attribute is related to the concept (see Table II). Then we group concepts by looking for attributes sharing (in our "case-A", variables). E.g., for "case-A" we group concepts into Z0-to-Z4 groups. Where Z0={C0,C1}; Z1={C0,C2}; and Z2/3/4={C0,C2,C3}.

TABLE II
CASE-A, "ATTRIBUTES-CONCEPTS" MATRIX

Attribute	C0	C1	C2	C3
index.html				
form	1	1		
member.php	1		1	1
Client_Page	1		1	
c1.htm	1		1	1
c2.htm	1		1	1
img				
access.html				
error.html				

Now we repeat the attributes-concepts matrix definition, using the same attributes list, but with the newly-grouped

[3] where *attribute* is from the C matrix, and *concept* was defined in the previous "Concept definition" step

concepts (Z0-to-Z4) and then we group these concepts attributes-based defining other new concepts (called ZZ0-to-ZZ4). Then we stop because these concepts are completely overlapped. Finally, we may define the set of concerns, where every Cx, Zx and ZZx is a good candidate (usable for our testing task). To reduce the number of candidates we delete overlapped concerns (see Table III). Every defined concern represents a clearly defined software behavior. We use these concerns to describe the Hyperspace slicing our application, and define the reduced diagrams.

TABLE III
CASE-A, CONCERNS (EVERY ROW)

Concept	Object	Attribute
C3	{username, password, $errorpage, $combine, $username, $password}	{member.php, Client_Page, c1.htm}
C2	{username, password, $combine, $username, $password }	{member.php, Client_Page, c2.htm, c1.htm }
C1	{username, password, submit, reset}	{form }
C0	{username, password}	{form, member.php, Client_Page, c1.htm, c2.htm }
Z2	{username, password, $errorpage, $combine, $username, $password}	{form, member.php, Client_Page, c1.htm, c2.htm}
Z0	{username, password, submit, reset}	{form, member.php, Client_Page, c1.htm, c2.htm }
ZZ0	{username, password, $errorpage, $combine, $username, $password, submit, reset}	{form, member.php, Client_Page, c1.htm, c2.htm }

Testing (**T**): from the reduced diagrams we may automatically define test cases or we may use these diagrams to verify coverage measures of already available test cases (such as in the user metrics driven test cases definition process [6]).

VI. CONCLUSIONS

We proposed a semi-automatic multi dimensional concerns mining approach based on: concept analysis combined with a grouping technique. This approach may help the user in slicing applications via model analysis, and it may be used to semi-automatically define application test cases, or test coverage measures or also to understand software evolution. We are currently investigating efficient pruning techniques to reduce the number of concerns generated by our approach. We are also working on a tool to integrate our approach in the WAAT project.

REFERENCES

[1] A. Deursen, M. Marin, and L. Moonen, "Aspect Mining and Refactoring". *First International Workshop on REFactoring: Achievements, Challenges, Effects (REFACE03)*, Canada. November 2003.
[2] A. Reina, J. Torres, and M. Toro, "Aspect-Oriented Web Development vs. Non Aspect-Oriented Web Development". *Workshop of nalysis of Aspect-Oriented Software (AAOS 2003)*, University of Darmstadt, Germany. July 2003.
[3] Aspectj. http://eclipse.org/aspectj
[4] B. Ganter and R.Wille, "Formal Concept Analysis". *Springer-Verlag*, Berlin, Heidelberg, New York, 1996.
[5] C. Bellettini, A. Marchetto, and A. Trentini, "WebUml: Reverse Engineering of Web Applications". *19th ACM Symposium on Applied Computing (SAC 2004)*, Nicosia, Cyprus. March 2004.
[6] C. Bellettini, A. Marchetto, and A. Trentini, "TestUml: User-Metrics Driven Web Applications Testing" 20th ACM Symposium on Applied Computing. USA 2005
[7] F. Ricca and P. Tonella, "Building a Tool for the Analysis and Testing of Web Applications: Problems and Solutions". *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'200)*, Genova, Italy. April 2001.
[8] G. A. Di Lucca, A. Fasolino, F. Faralli, and U. De Carlini, "Testing web applications". *International Conference on Software aintenance (ICSM'02)*, Montreal, Canada. October 2002.
[9] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin, "Aspect-Oriented Programming". *11th Europeen Conf. Object-Oriented Programming*, Springer Verlag. 1997.
[10] G. Murphy, A. Lai, R. Walker, and M. Robillard,. "Separating Features in Source Code: An Exploratory Study". *23rd International Conference on Software Engineering*, Toronto, Canada. May, 2001.
[11] Hyperj. http://www.research.ibm.com/hyperspace
[12] J. Conallen. *Building Web Applications with UML*. Addison-Wesley, 2000.
[13] J. Stanley and M. Sutton "Multiple Dimensions of Concern in Software Testing". *First Workshop on Multi-Dimensional Separation of Concerns in Object-oriented Systems (OOPSLA'99)*, November 1999.
[14] M. Bruntink, A. van Deursen, and T. Tourwè "An Initial Experiment in Reverse Engineering Aspects from Existing Applications". *11th IEEE Working Conference on Reverse Engineering (WCRE 04)*, Netherlands. November 2004.
[15] M. Bruntink, A. van Deursen, R. van Engelen, and T. Tourwè, "An Evaluation of Clone Detection Techniques for Identifying Cross-Cutting Concerns". *IEEE International Conference on Software Maintenance (ICSM 04)*, 2004.
[16] M. Han and C. Hofmeister, "Separating and Representing Navigation Concerns in Web Applications". *Lehigh University, Technical Reports*, 2004
[17] N. Noda and T. Kishi, "On Aspect-Oriented Design Applying Multi-Dimensional Separation of Concerns on Designing Quality Attributes". *First Workshop on Multi-Dimensional Separation of Concerns in Object-oriented Systems (OOPSLA'99)*, November 1999.
[18] P. Tarr, H. Ossher, W. Harrison, J. Stanley, and M. Sutton, "N-degrees of separation: Multi-Dimensional Separation of Concerns". *21st International Conference on SoftwareEngineering*, IEEE Computer Society Press, 1999.
[19] P. Tonella and M. Ceccato, "Aspect Mining through the Formal Concept Analysis of Execution Traces". *11th IEEE Working Conference on Reverse Engineering (WCRE 04)*, Netherlands. November 2004.
[20] Rational Rose Web Modeler, http://www.rational.com
[21] S. Breu and J. Krinke. "Aspect Mining Using Event Traces". *19th. Conference on Automated Software Engineering 2004 (ASE 04)*, Linz, Austria. September 2004.
[22] T. Apiwattanapong, A. Orso and M.J. Harrold, "Efficient and Precise Dynamic Impact Analysis Using Execute-After Sequences" *27th IEEE and ACM SIGSOFT International Conference on Software Engineering (ICSE 2005)*. USA. 2005
[23] ToscanaJ, http://toscanaj.sourceforge.net/