# TURNING CONCEPTS INTO REALITY
## *Bridging Requirement Engineering and Model-Driven Generation of Web Applications*

Xufeng (Danny) Liang, Christian Kop
*School of Computing and Mathematics, University of Western Sydney, Sydney, Australia*
*Alpen-Adria-Universitaet Klagenfurt, Klagenfurt, Austria*

Athula Ginige, Heinrich C. Mayr
*School of Computing and Mathematics, University of Western Sydney, Sydney, Australia*
*Alpen-Adria-Universitaet Klagenfurt, Klagenfurt, Austria*

Keywords: Requirements engineering, web engineering, conceptual modeling, rapid development, model driven software development.

Abstract: Today web application development is under the pressure of evolving business needs, compressed timelines, and limited resources. These dynamics demand a streamlined set of tools that turns concepts into reality and minimises the gap between the original business requirements and the final physical implementation of the web application. This paper will demonstrate how this gap can be reduced by the integration of two techniques, KCPM (Klagenfurt Predesign Conceptual Model) and SBO (Smart Business Object), allowing fully functional web applications to be auto-generated from a set of glossaries.

## 1 INTRODUCTION

One of the major challenges in developing web applications is to minimise the requirement and time gap between the development domain and the actual problem domain. End users tend to express their requirements using natural language. They are neither capable nor willing to understand conceptual models that reflect the vocabulary of concepts of the computing domain within which the solution needs to be provided. On the other hand, system designers can easily understand the requirements as well as to identify any missing requirements by looking at conceptual models. In the absence of a coherent conceptual model, i.e. a way for system designers to express the concepts that end users could also understand, the first opportunity end users get to verify whether their requirements have been properly understood by the designers and the developers is when they see the implemented system.

Several methodologies have been developed so far which focus on different concepts and techniques to improve the communication and to automate the web application development. Some of these are goal modeling (Mylopoulos et al., 1999) , scenario based approaches (Sutcliffe et al., 1998; Rolland et al., 1998; Diaz et al., 2005), story boarding (Schewe et al., 2004), tasks (Valderas et al., 2005), template based approaches, navigation approaches (Escalona et al., 2004), specific use cases for web development (Koch et al., 2006; Koch, 2006), as well as user services approaches (de Castro et al., 2004). Furthermore the need for a controlled natural language is proposed in (Hoppenbrouwers et al., 2005) as a meeting point between natural and formal languages.

Between the original concepts and the final physical implementation, often there is a requirement gap and a time gap. A requirement gap can arise due to different levels of communication loss, for example, between end users and designers, or between designers and developers. It also can arise due to end users not being able to identify all the requirements at the beginning. There is also a time gap, which is the time required to turn the original concept into the final physical implementation. Minimising the time gap can effectively reduce the requirement gap, since end users can quickly experience the system, identify any gaps (missing information) in relation to their requirements, and request necessary changes. Therefore, in this paper, we propose an approach that minimises these gaps by integrating Klagenfurt Predesign Conceptual Model (KCPM) (Mayr and Kop, 2002) and



(Danny) Liang X., Kop C., Ginige A. and C. Mayr H. (2007).
TURNING CONCEPTS INTO REALITY - Bridging Requirement Engineering and Model-Driven Generation of Web Applications.
In *Proceedings of the Second International Conference on Software and Data Technologies - Volume ISDM/WsEHST/DC*, pages 109-116
DOI: 10.5220/0001333601090116

Smart Business Objects (SBO) (Liang and Ginige, 2006).

KCPM is aimed to minimise the requirement gap. It provides a glossary representation of the natural language requirements that can be directly used to verify with end users and to identify any missing information immediately. In last year's ICSOFT conference, we have proposed the SBO concept to effortlessly generate web applications by modeling business objects using high level modeling language called SBOML (Smart Business Object Modelling Language) (Liang and Ginige, 2006). By encapsulating common web behaviors and characteristics, SBO understands high-level concepts, such as email, photo, URL, document, video etc. Consequently, SBO is capable of directly generating a rich set of ready-to-use web views (user interfaces) for the modeled business objects, allowing web applications to be developed within an extremely condensed timeframe. In this paper, we further reduce the requirement gap and the time gap by introducing a rule-based engine to automate the process of converting the KCPM glossaries into a conceptual model of the business objects in SBOML.

Web applications can be developed in several ways, such as by navigation, tasks (Liang and Ginige, 2006), or user services (Koch, 2006; de Castro et al., 2004). In order to fully understand the requirements of the required application, a business analyst needs to understand the terms and concepts in the target domain and the relationships among them. A glossary is commonly used to maintain a list of the terms as well as their meanings. Thus, in practice, business analysts do not always follow the rule *tasks first* in requirement analysis, apart from the initial tasks to obtain a broad, high-level understanding what the intended application should do. Since it is natural for end users to use jargon to express concepts in their domain (e.g. *customer*, *product*, *icd10*), business analysts will soon confront the underlying data model of the intended application during a detailed analysis. From this point on, in practice, domain analysis, or the analysis of the underlying end users *ontology*, must go hand in hand with task analysis.

Existing approaches in deriving the data model (i.e. the conceptual modeling of data) from user requirements, such as the *template* mechanism used in (Koch et al., 2006; Escalona et al., 2004), requires business analysts to make decisions for the structure of the required data model (such as which concepts should become classes or attributes) at a very early stage of the development process. On the contrary, since KCPM is capable of using a rule-based engine (explained in section 5) to systematically derive the required data model based on the terms and relationships stored in its glossaries (explained in section 2), the laborious work of manually constructing the data model is alleviated from business analysts or system designers. Additionally, KCPM has an extensible structure to support the need for collecting additional information about the domain specific concepts.

The integration of KCPM and SBO leads to an approach where functional web applications can be generated purely by collecting information about data. This is a data-driven approach to develop web applications. Existing web modeling techniques, such as WebML (Ceri et al., 2000) and OOHDM (Rossi et al., 2000), use a similar data-driven approach. However, the starting point of existing data-driven approaches is the conceptual modeling of data (using ER diagrams or UML class diagrams), from which web applications can be generated or developed. The integration of KCPM and SBO allows us take a step further. We take user requirements (domain specific terms and concepts) in the form of KCPM glossaries as our starting point, where the conceptual model can be derived automatically and web applications can be then generated.

The paper is structured as follows: We will introduce the concept of KCPM and SBO in sections 2 and 3 respectively. In section 4, we will explain our data-driven approach to generate web applications from requirements. Section 5 explains how the KCPM glossaries are mapped to SBOML expressions. We will then demonstrate the tools we have developed to facilitate the generation process in section 6. Section 7 describes the future directions.

## 2 KCPM

An important aim for the development of KCPM was to provide both developers and end users with an interface for their mutual understanding. The most important modeling notions for modeling the structural (data) aspects of a software system are: thing-type, connection-type and perspective. In this section, we will explain, via examples, these notions and their associated meta-attributes in the glossaries.

A thing-type is a generalisation of conceptual notion classes (entity type or entity set, respectively) and attributes. Thus, typical things (instances of thing-types) are:

- Natural or juridical persons (e.g. author),
- Material or immaterial objects (e.g. book, product, idea),
- Abstract notions (e.g. contract), as well as,

- Descriptive characteristics of the abovementioned examples (e.g. a customer name, a product number, a product description)

It is insufficient to collect only the names of the thing-types, such as author, ISBN, book, etc. KCPM also use a set of meta-attributes to define thing-types:

- Description - a semantic description of the thing-type
- Examples- a list of examples for the specific thing-type.
- Synonyms - a list of alternative names to describe the same thing-type
- Value Constraint - a description about the constraints of a value (e.g. the value should be between 4 and 10)
- Format Constraint - a description for the format of the value (e.g. YYYY-MM-DD).
- Quantity Description - a description for the possible amount of instances of a particular thing-type (e.g. 1000 customers, 3 colors)
- Requirement Source - the references to the sources (documents) from which the particular thing-type was derived.

All these information can be derived by analysing the textual requirements or by asking the end users (Figure 1). The advantage of this approach is that system designers can immediately identify any inconsistent or missing information, for example, if end users have specified two different format constraints, "YY-MM-DD" and "DD-MM-YYYY", for the same thing type.

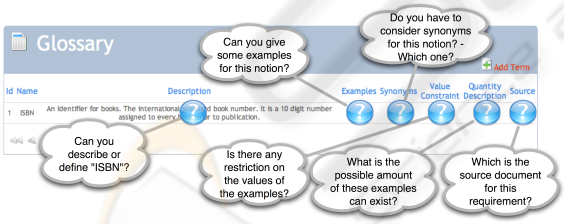


Figure 1: Completing the Glossary.

Things in the real world are related. To capture the relationships between things, KCPM also introduces the notion of connection-type. Two or more thing-types can be connected via a connection-type. To completely define a connection-type, we must specify the viewpoint (perspective) of all of the involved thing-types. This corresponds to the NIAM Object/Role Model (Nijssen and Halpin, 1989). It is possible to specify the cardinalities of a connection-type within a perspective. Natural language sentences can lead to connections (and perspectives), for example:

- "An Author writes many books." - from the perspective of an author
- "A Book is written by many authors." - from the perspective of a book

KCPM also allows special connection-types with specific semantic meanings to be defined. These special connection-types can be mapped to conceptual models. Typical semantic connection-types are:

- Identification - specifies that the instances of a thing-type identifies the instances of another thing-type (e.g.: An ISBN number identifies a book)
- Generalisation ("is-a") - a set inclusion on the instance level
- Has Property ("has") - a generic connection-type can lead to different semantic relationships. In this way, systems designer can express that a thing-type has some "properties" without specifying whether the "property" is an attribute or a class (in OO terms). For example, an address is a typical property of a person. However, the address may not necessarily become an attribute of person. The address itself may consist of other relationships with other information, such as city, zip code, and street, which enforce the address to be a class. Thus, further information about the thing-type address is required to determine whether it should become an attribute or not. This strategy is based on (Storey, 1993), where the word "has" have different semantic meanings.

There are three methods for end users to specify connection-types in KCPM:

- Adding new connection-types directly in the glossaries
- Generating connection types along with their associated thing-types via a graphical editor
- Using on sentence patterns of a controlled languages (Hoppenbrouwers et al., 2005) to define connection-types, as well as their cardinalities. For example, "A Book is written by many author", "An ISBN number identifies a book", etc.

If the third method is adopted and assuming the sentences in the requirements conform to the sentence patterns of a control language, then we can use an interpreter (a tool) to translate those sentences to KCPM connection-types.

## 3 SMART BUSINESS OBJECT

Smart Business Object (SBO) is built on early work done by Reenskaug in MVC (Model-View-

Controller): a modeling approach to bridge the gap between users' mind and computer data (Reenskaug, 1979b; Reenskaug, 1979a). The aim of SBO is to raise the level of abstraction in modeling business objects, as well as auto-generating "web-ready" views of the modelled business objects for web-based business applications.

In building web-based business applications, we need to develop web user interfaces (UIs). In order to accelerate the development of web-based business applications, higher-level, semantic-rich Abstract Data Types (ADTs) that embrace web characteristics and behaviours is desirable. For example, an *email* data type should be by default rendered as a mailto hypertext link for view operations, while both server side and client-side validation are enforced on update operations. Similarly, an *photo* data type should be by default rendered as an image on the web browser, while uploading facilities are automatically provide on update options. In other words, these Abstract Data Types also need to be also context-aware, in terms of the web user interfaces they are to generate for different operations (such as CRUD). We call these *Web-Oriented Abstract Data Types*.

In order to represent the attributes of common business objects, SBO is designed to support *Web-Oriented Abstract Data Types*, such as email, document, photo, video, etc. Thus SBO greatly reduce the laborious and repetitive work in building web user interfaces for business objects. SBO not only consider the database schema level semantics (i.e. data types in the database) for determining the most suitable web user interfaces, but also maintains a global schema for representing *Web-Oriented Abstract Data Types*. This schema maintains the additional aspects, such as presentation logic, validation logic, and content handling mechanism, for customising or creating new *Web-Oriented Abstract Data Types*. SBO maintains a global schema for defining *Web-Oriented Abstract Data Types*.

The direct support for *Web-Oriented Abstract Data Types* in SBO enables the construction of a formal modeling language for modeling business object at a very high level of abstraction. The proposed modeling language is called Smart Business Object Modeling Language (SBOML). The syntax keywords of SBOML are based on the English lexicon, allowing users to model their domain specific business object using near-natural language syntax. A detailed explanation of the SBOML syntax can be found in (Liang and Ginige, 2006).

An example would be the use of the "has" and the "many" keyword. According to Storey (Storey, 1993), the word "has" can be used to describe several types of semantic relationships, including "one thing connected to another" and "a property or characteristic of something". In order to make the syntax more generic, SBOML use the "has" keyword for defining both "object-object" relationships (composition) and "object-attribute" relationships. For example, the expression "employee has position" implies an "object-attribute" relationship if "position" is not a known business object. Otherwise, the same expression will be considered as an establishment of a composition relationship between the "employee" and the "position" business object. To specify the cardinality in relationships between business objects, SBOML uses the "many" keyword in combination with the "has" keyword. For example, assuming that "car" is a known business object, then the expression "employee has many car" will imply a cardinality of [1..*]. SBO is also capable of making logical inferences about the attribute types of business objects based on a set of rules. This eliminated the need for explicitly specifying the *Web-Oriented Abstract Data Types* for attributes when defining business objects, allowing the SBOML syntax to be closer to the natural language.

Following is an example using SBOML to model business objects:

```
in organisation, employee has first name,
last name, gender, date of birth, photo,
email address, home phone, hobby (which
could be reading and skiing and coding),
position (has title, description)
```

The above expression literally defined an "employee" business object and a "position" business object in the "organisation" namespace, where "employee" has a "position" (establishing a composition relationship). This expression is sufficient for SBO to automatically generate the necessary physical implementation to represent the two business objects to the web. In other words, we can directly generate the necessary web user interfaces (views) for the "employee" business object, such as web forms for adding new employees or listing all the existing employees in web tables.

SBO embraces a rich set of rendering APIs for generating commonly used web user interfaces in business web applications, such as tables, forms, charts, and menus. Thus, by modeling business objects using SBOML and using the SBO toolkit (illustrated in section 6), we can effortlessly generate fully functional web applications without any coding.

Figure 2 is an example of rendering the both the "employee" and the "position" business objects as web tables within a tab menu. According to the definitions of the *Web-Oriented Abstract Data Types* stored in the SBO global schema, in the generated web user

interface automatically:

- Displays *Photos* as images,
- Formats *Email* addresses as email hypertext links,
- Utilises Google maps to show the location of the *Addresses*,
- Presents *Date of Births* in a format based on the users' locale.

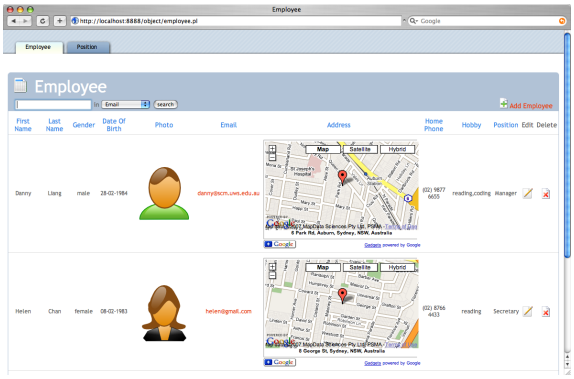


Figure 2: Example of an Auto-Generated Web User Interface.

The auto-generated user interfaces implies an implicit navigation model for performing various operations to the business object. For instance, in Figure 2, once the user clicks on the "Add Employee" link, the web table will hide and a form is then shown. There are different methods to customise or enhance the default navigation behaviours:

- By customising the parameters passed to the corresponding rendering API to make fine adjustments, such as whether to hide or show a web form on certain events,
- By specifying custom web templates to the corresponding rendering API to incorporate additional navigation paths, or
- By defining custom *controllers* (refers to the *controllers* in MVC architecture) in the current user interface to trigger the rendering of another user interface, and thus, different user interfaces are sequenced.

Due to the scope of this paper, we will not focus on the details of customising the implicit navigation model of the rendering APIs.

To generate web applications using SBO, however, requires a middle person, usually system designers, to manually interpret the end user requirements and manually construct the SBOML expressions (i.e. conceptual modeling). As previously mentioned, end users are neither capable nor willing to understand conceptual models. Despite the fact that SBOML has a near-natural syntax, it still assumes the understanding of prerequisite Object Oriented concepts, for example, the differences between objects-attribute relationships and object-object relationships, as well as their implications to the final implementation. These OO concepts are beyond the capability of normal end users. Since KCPM is capable of automatically deriving objects-attribute relationships and object-object relationships between thing-types, integrating KCPM and SBO is a plausible approach to even eliminate the manual process of conceptual modeling using SBOML expression.

## 4 A DATA-DRIVEN APPROACH

A complete web application is composed of many aspects, such as data (business objects), view, navigation, presentation, access control, personalisation and so on. In business web applications, most of the other aspects are either depended on the data or influenced by the data. Understand data in the target domain is indispensable for business analysts to have a better understanding of the associated business processes or tasks. Thus, data modeling is inevitable even if business analysts prefer to start with task modeling in developing web applications. In developing business web applications, we believe it is plausible to begin with data modeling in parallel with task modeling.

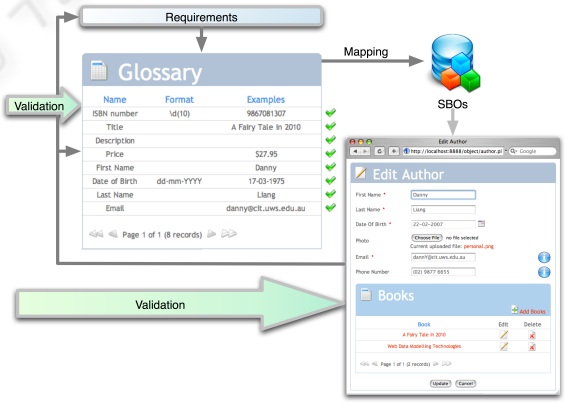


Figure 3: From User Requirements to Implementations.

Figure 3 illustrates the process used in the proposed approach. Natural language specifications are first filtered and classified manually. This step is to populate the KCPM glossaries, which contain important domain specific terms (concepts) and the relationships among them. Then system designers can use the KCPM glossaries to verify the completeness of the requirements with end users by iterating through each collected term, for example, whether a particular

term requires a format description (e.g. "YYYY-MM-DD") or a quantity description. Once this process is completed, we can convert the terms and relationships stored in the KCPM glossaries (using the conversion mechanism is explained in section 5) to SBOML specifications via tools, from which web applications can be quickly generated. End users can then experience fully functional applications, allowing them to pinpoint incorrect or missing requirements and refine the KCPM glossaries. We can effortlessly regenerate the web application if necessary using the refined KCPM glossaries until end users are satisfied. Once end users are satisfied with the generated version of web application, system developers can then take on and further refine the application, allowing the application to evolve into the final production-ready system.

The advantage for integrating KCPM and SBO, is that it allows end users, who are the real experts of the problem domain, to have two checkpoints to validate the solution in the computer domain:

- By validating the KCPM glossary, and
- By validating the generated application from SBO.

As a result, both the requirement gap and the time gap between the concept and the final physical implementation are minimised.

# 5 CONVERTING KCPM GLOSSARIES TO SBOML

We utilise a conversion algorithm that is based on the KCPM to UML mapping developed in (Mayr and Kop, 2002) to convert the terms and relationships in the KCPM glossaries into SBOML expressions. The goal of the conversion algorithm is automate the conversion process, i.e. the process of creating a conceptual model in SBOML from user requirements in the form of KCPM glossaries, as much as possible. To ensure all terms and relationships are considered, those that cannot be converted automatically using the algorithm are forwarded to system designers for their decisions.

The conversion algorithm uses two types of rules, laws and proposals, to derive classes and attributes from the terms and relationships stored in the KCPM glossaries. The rules are orthogonal in the sense that each rule analyses another piece of information about a concept. When rules are applied to a term, not only possible conversions are derived, but also contradicting results are detected. While a proposal may suggest the suitable concept that a thing-type should be converted (mapped) to, a law will always force a thing-type to be converted to a particular concept for the target conceptual model. Since our target conceptual model is SBOML, classes and attributes are the two possible concepts that thing-types can be converted to. Laws are here to ensure the syntactical correctness of the final conceptual model. Therefore, by definition, laws override proposals.

Moreover, rules are also further classified as:

- Direct rules - Direct rules are rules that rely only on existing information (terms and relationships) in the KCPM glossaries. An example of a direct rule would be:*if two thing-types are connected by an "is-a" relationship, then both thing-types should be mapped (converted) to classes*. This direct rule is based on the fact that "is-a" relationships can only exist between classes, according to Object Oriented design principles. A similar rule would be:*if there exists a relationship between two thing-types, X and Y, such that X "has" Y, then X must be a class*. However, using this rule alone, we cannot make any conclusions for Y yet, until further information is available.
- Indirect rules - Indirect rules are rules that require information about thing-type(s), which have been mapped (converted) to some concepts by applying direct rules. For example, assuming that a thing-type A has been mapped to an attribute and a thing-type E has been mapped to a class, both due to some direct rules. Assuming that the target conceptual model does not support nested attributes, then an indirect rule can be applied to conclude that the thing-type D, by having relationships (connection-types) to both E and A, must be a class.

The conversion algorithm is a two-step process. The first step is to apply all the direct rules to each thing-type. The second step is to apply all the indirect rules to the result (output) of the first step. After the execution of the mapping procedure, all the connection-types are associated with at least two or more classes, which are derived from the corresponding thing-types. Due to space limit, we are unable to include the actual logic of the mapping procedure in this paper.

# 6 GENERATING WEB APPLICATIONS FROM REQUIREMENTS

In this section, we will demonstrate the tools we have developed to streamline the web application genera-

tion process from KCPM glossaries. Based on the mechanism described in section 5, we have developed a prototype tool called the KCPM2SBO Converter, which can convert the terms and relationships in the KCPM glossaries into SBOML expressions. Given the following user requirements (in a controlled language) to populate the KCPM glossaries:

> "A book is written by many authors. An author writes many books. A book has an ISBN number, title, description, and price. An ISBN number identifies a book. An author has a first name, last name, email address, photo, and a phone number."

```
<sbo>
  <business_objects>
    <business_object>
      in publisher, author has first name,
      last name, photo, email, phone number
    </business_object>
    <business_object>
      in publisher, book has ISBN number, title,
      description, price
    </business_object>
    <business_object>
      in publisher, book author has author(id),
      book (id)
    </business_object>
  </business_objects>
  <relationships>
    <relationship>
      publisher book has publisher author
      via publisher book author
    </relationship>
  </relationships>
</sbo>
```

We can then use the XML output to generate web applications using the SBO toolkit that we have previously developed (Liang and Ginige, 2006). The SBO toolkit is designed to streamline the creation (modeling) and consumption (execution) of SBOs for developing web applications on the CBEADS© (Ginige et al., 2005) web framework. CBEADS© consists of a user management module and an application development module that enable developers to easily create and manage web applications within the framework itself. The SBO Builder allows users to model and create SBOs and relationships among them using the SBOML. The SBO User Interface Generator allows users to easily create applications on the CBEADS© framework by rendering SBOs using the SBO rendering APIs.

Once the generated XML file is uploaded to the SBO Builder (Figure 4), all the business objects (classes) as well as the relationships among them are generated. Based on the created business objects using the SBO Builder, we can define new UI definitions using the SBO UI Generator tool shown in Figure 5. Each UI definition keep track of the business

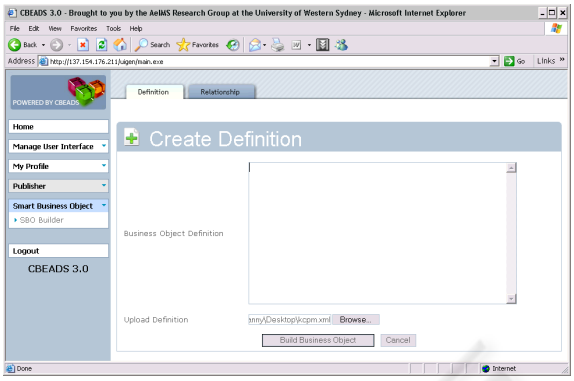


Figure 4: Uploading SBOML Definition using SBO Builder.

object used, the UI type (e.g. form, table, navigation menu, etc.), and various options based on the selected UI type (e.g. the order of the form fields). Then, we can assign the UI definition to a CBEADS© *function* (i.e an application code file in CBEADS©). The SBO UI Generator will generate the necessary codes for the appointed CBEADS© function to render the business object (Figure 6).

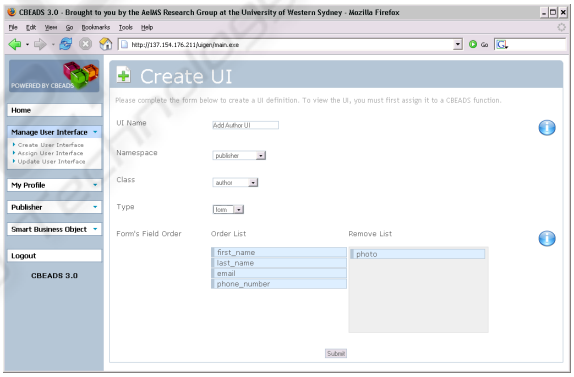


Figure 5: Creating UI Definitions.

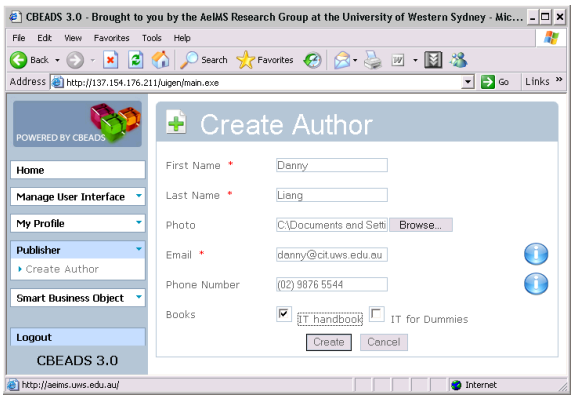


Figure 6: The Generated Application.

# 7 CONCLUSION AND FUTURE WORK

In this paper, we have demonstrated how web applications can be rapidly generated from the initial concepts to the final system: first capture user requirements using the KCPM glossaries, then use the KCPM2SBO converter to generate a conceptual model in SBOML specifications, from which web applications can be generated on CBEADS© framework via the SBO toolkit. We are currently working on the validation of our proposed approach in several industrial web application development projects.

Data is one of the many core aspects of complex business web application we have today. KCPM is capable of deriving conceptual models beyond data while SBO is architected to support complex navigation, fine-grained access control mechanism, and modern workflow engine. This integration of KCPM and SBO is only a starting point, which primarily focuses on generating essential web user interfaces purely based on the conceptual modeling of data. The automated generation of other aspects, such as complex behavior and navigations, dynamic access control, and user preferences, while are not considered in this paper, are to be explored. Our ultimate goal is to support as much as possible the generation of complete web applications from natural language specifications. This requires further work on extracting and transforming other aspects of the web application from the KCPM glossaries into conceptual models from which web applications can be generated.

# REFERENCES


Ceri, S., Fraternali, P., and Bongio, A. (2000). Web modeling language (webml): a modeling language for designing web sites. *WWW9 Conference*.

de Castro, V., Marcos, E., and Cáceres, P. (2004). A user service oriented method to model web information systems. In *WISE 2004*. International Conference on Web Information Systems Engineering (WISE 2004).

Diaz, I., Moreno, L., Pastor, O., and Matteo, A. (2005). Interaction transformation patterns based on semantic roles. In *NLDB 2005*, pages 239–250. 10th International Conference on Applications of Natural Language to Information Systems (NLDB 2005).

Escalona, M., Reina, A., J.Torres, and M.Mejías (2004). Ndt: a methodology to deal with the navigation aspect at the requirements phase. In *OOPSLA 2004*. OOPSLA Workshop: Aspect-Oriented Requirements Engineering and Architecture Design.

Ginige, J. A., Silva, B. D., and Ginige, A. (2005). Towards end user development of web applications for smes: A component based approach. In *ICWE*, Sydney, Australia. International Conference on Web Engineering (ICWE 2005).

Hoppenbrouwers, S., H.A, P., and van der Weide Th.P. (2005). A fundamental view on the process of conceptual modeling. In *ER 2005*. 24th International Conference on Conceptual Modeling (ER 2005).

Koch, N. (2006). Transformation techniques in the model-driven development process of uwe. In *MDWE 2006*. 2nd Model-Driven Web Engineering Workshop (MDWE 2006).

Koch, N., Zhang, G., and Escalona, M. J. (2006). Model transformation from requirements to web system design. In *ICWE 2006*, pages 281–288. 6th International Conference on Web Engineering (ICWE 2006).

Liang, X. and Ginige, A. (2006). Smart business object - a new approach to model business objects for web applications. In *ICSOFT*, volume 2, pages 30–39. International Conference on Software and Data Technologies (ICSOFT 2006).

Mayr, H. C. and Kop, C. (2002). A user centered approach to requirements modeling. In *Modellierung in der Praxis*, pages 75–86.

Mylopoulos, J., Chung, L., and Yu, E. (1999). From object oriented to goal oriented requirements analysis. *Communications of the ACM*, 24(1).

Nijssen, G. and Halpin, T. (1989). *Conceptual Schema and Relational Database Design - A Fact Oriented Approach*. Prentice Hall.

Reenskaug, T. (1979a). Models - views - controllers.

Reenskaug, T. (1979b). Thing-model-view-editor: an example from a planning system.

Rolland, C., Souveyet, C., and Achour, C. B. (1998). Guiding goal modeling using scenarios. *IEEE Transaction on Software Engineering*, 24(12):1055–1071.

Rossi, G., Garrido, A., and Schwabe, D. (2000). Navigating between objects. lessons from an object-oriented framework. *ACM Computing Surveys (CSUR)*, 32(1).

Schewe, K., Thalheim, B., and Zlatkin, S. (2004). Modelling actors and stories in web information systems. In *ISTA 2004*, pages 13–23. 3rd International Conference on Information Systems Technology and ist Applications (ISTA 2004).

Storey, V. C. (1993). Understanding semantic relationships. *The VLDB Journal - The International Journal on Very Large Data Bases*, 2(4):455–488.

Sutcliffe, A., Maiden, N. A., Minocha, S., and Manuel, D. (1998). Supporting scenario-based requirements. *IEEE Transactions on Software Engineering*, 24(12):1072–1088.

Valderas, P., Fons, J., and Pelechano, V. (2005). Transforming web requirements into navigational models: An mda based approach. In *ER 2005*, pages 320–336. 24th International Conference on Conceptual Modeling (ER 2005), Springer.