Web 2.0 Patterns: A Model-driven Engineering Approach

Ana Rosa Guzman, Victoria López, Francisco Valverde
PROS Research Center
Universitat Politècnica de València
Camino de Vera S/N, 46022, Valencia, Spain
{aguzman,vlopez,fvalverde}@pros.upv.es

Jose Ignacio Panach
Escola Tècnica Superior d'Enginyeria,
Departament d'Informàtica, Universitat de València
Av. de la Universidad, s/n 46100 Burjassot, València, Spain
joigpana@uv.es

***Abstract*—Web 2.0 is a concept that is not only discussed by the Software development community but also in the research community. However, Web Engineering methods, specifically Model-driven Engineering (MDE) ones, are not yet ready to cope with advanced user involvement features that Web 2.0 is demanding. In this work the concept of Web 2.0 pattern is introduced to support the user involvement concern. With the goal of providing a formal description, Web 2.0 patterns are defined precisely using conceptual models that represent both interaction and functionality, since in Web 2.0 applications both features are intertwined. This work also presents how the conceptual models that describe these Web 2.0 patterns can be integrated into a model-driven Web Engineering method. As proof of concept of the contribution, the integration of the Quick Comment pattern with the OOWS 2.0 method is discussed.**

I. INTRODUCTION

Nowadays Web 2.0 is a topic with a remarkable impact on the development of Web applications. However, the Web 2.0 definition [14] has been traditionally ambiguous because it defines a set of best practices and technologies that, at first glance, are not clearly related. Nevertheless, everybody agree that the Web 2.0 summarizes the evolution of the Web in recent years [13]. This evolution has introduced a set of development technologies that have improved: 1) the usability perceived by the end-users [28], and 2) the information integration among several Web 2.0 applications [29]. Simultaneously, Social Webs such as Facebook or Twitter, in which user involvement is a key concern, have gained relevance as the best examples of Web 2.0 applications. Two main distinctive characteristics describe these Social Webs: 1) the creation of virtual communities to link users that share the same interests, and 2) the definition of the Web content by users. Therefore, a successful user involvement has been also a decisive factor in the current success of the Web 2.0.

Industry has perceived the great potential of this new tendency and it has started to demand Web 2.0 features inside their Information Systems. The main detected benefits are the ability to reach new audiences and the increase of customers' involvement and loyalty. Additionally other interesting advantages have been detected. Firstly, the information available in such Web 2.0 applications can be extremely useful to target a product for a specific users group. Since people introduce their personal profile in these Social Webs, customized adverts can be easily addressed. And secondly, the feedback among customers, employees and third parties is improved. Therefore, though Web 2.0 was introduced as a marketing word, nowadays it is a philosophy with a considerable influence on the development of industrial Information Systems.

Web Engineering has provided interesting MDE methods [17] to face the development of Web Applications. These methods have defined conceptual models to support different concerns of the Web Application development. Whereas several works [3][12] have addressed how to introduce new Web 2.0 technologies in a model-driven method, few works have proposed how the user involvement can be supported as well. As the demand of Web 2.0 applications is increasing, the inclusion of this facet within a MDE development process is an interesting research topic. The main advantage is that the analyst could easily model the social facet of the Web application at the problem space avoiding implementation decisions. From that modeling step, the final code can be generated from these conceptual models. As a result, the Web 2.0 Application development should improve in terms of time and maintenance, according to the MDE benefits that the Software Engineering community states [25][26].

Currently, there are sound works in the MDE paradigm to model interface and functionality separately. For example, UML Class Diagrams to represent data persistency and UMLi [27] or CTTs [16] to represent the interaction. However, these works have not been adapted to the Web 2.0 development, in which interaction and functionality has to be considered as a whole. The main contribution of this work is to provide a holistic model-driven approach in order to support the Web 2.0 development in the current MDE methods.

Many Web 2.0 Applications are developed using design patterns that are useful to emphasize the user involvement. Some examples of these patterns are the user mechanisms for creating and evaluating the content in Web 2.0 applications. This work proposes introducing those solutions described as

patterns at the conceptual level. To achieve this goal, the Web 2.0 pattern concept is defined as a reusable solution based on conceptual models for improving the end-user involvement in the creation of Web content. In current approaches, a Web 2.0 pattern is specified using a pattern template made up of the common sections for describing design patterns [9]. However, this textual description is sometimes ambiguous and cannot be included in a MDE paradigm, where model-to-model (M2M) and model-to-code (M2C) transformations require unambiguous rules.

With the aim of providing a more precise description, we have proposed a Web 2.0 pattern template that also includes the definition of two conceptual models: 1) a Functionality Model represented using an UML Class Diagram to describe the data structures and operations abstracted by the pattern and, 2) an Interaction Model represented using the Concur-Task Tree notation [19] to describe the user interaction when the pattern is applied.

Additionally, this work presents a strategy to integrate these Web 2.0 patterns into a model-driven Web Engineering method using model-to-model transformations (M2M). To illustrate that strategy the OOWS 2.0 Web Engineering method, which also follows the MDE paradigm, is extended with the inclusion of the Quick Comment pattern.

The rest of the work is organized as follows: Section 2 describes the related work. Next, section 3 introduces the Web 2.0 pattern concept and presents an example: The Quick Comment Pattern. Section 4 introduces the strategy for integrating the Web 2.0 patterns at conceptual level. This strategy is applied into the OOWS 2.0 method in order to integrate the Quick Comment pattern previously described. And finally, Section 5 presents the concluding remarks and future works.

II. Related Work

The concept of pattern, which was proposed by Alexander [1] in urban architecture, has been widely used in the Software Engineering community. In software development, design patterns [9] have been applied as implementation guides to improve the code quality. Furthermore, in the User Interface (UI) development discipline, several pattern libraries have been defined by authors such as Tidwell [19] or Welie [30]. These solutions have been shown to benefit end-users; they have already been implemented, evaluated, and proved. However, design patterns are focused on functionality whereas UI patterns are focused on interaction issues. Web 2.0 application requires a solution that includes both concerns.

Since Web 2.0 applications are built around clearly defined best practices, it is an interesting approach to gather that knowledge as patterns. In this reasoning line, one of the most relevant works is the pattern library developed by Yahoo [31]. This library includes both technological and social patterns that address several best practices of the Web 2.0. The main interest of this library is that the patterns are illustrated using examples from Yahoo Web applications. Additionally, an extension of this library is presented by Crumlish and Malone [6]. Another interesting pattern library can be found in the UI patterns Website [20]. Though this library is focused on the UI design, it also includes five social or community-driven patterns. These patterns are described using a standard template as well.

Since the inclusion of patterns in model-driven environments is a relevant topic, Levendovszky et. al [11] formalize the different constructs needed for supporting domain-specific model patterns in metamodels. This proposal provides a set of theoretical foundations that can also be applied to patterns from the Web 2.0 domain. In a similar reasoning line, Koch et. al [10] propose a pattern approach for the model-based engineering of Rich Internet Applications (RIAs). This approach is based on UML state diagrams that model the behavior of the RIA pattern. However the patterns proposed (Autocomplete and Dynamic refresh) deal only with technological issues related to Web 2.0 Applications development.

A model-driven approach for supporting the social facet of Web 2.0 development has been proposed by Fraternali et. al [8]. In that work, several community-driven patterns extracted from popular Web 2.0 applications are described. Authors also discuss how the patterns can be defined using models from the WebML method [5]. However, how the same pattern can be introduced into another Web Engineering method is not addressed.

From all this study we can conclude that, in general, existing pattern libraries are based on textual descriptions and code examples. These descriptions are ambiguous and, at the end, the developer is who decides the specific implementation of the pattern. In order to mellow this problem, we propose a more precise solution using conceptual models. Apart from defining the patterns unambiguously, another advantage of our approach is that the proposed solution is neither linked to a specific technology nor implementation decisions. Moreover, the analyst does not have to recurrently implement the pattern; the code is automatically generated by M2C transformation rules that implement the targeted transformation.

Existing approaches that deal with Web 2.0 patterns in a MDE context, such as the works of Fraternali et al. and Noch et al, are based on a specific Web Engineering method (WebML and UWE respectively). For emphasizing the knowledge reuse, we aim to develop a proposal that can be applied to any MDE method. Additionally, we also aim to define a strategy for integrating the Web 2.0 patterns into the metamodel of the methods, such as the work of Levendovszky et. al justifies.

III. A Model-based Approach for Web 2.0 Patterns

Patterns are usually defined in the literature using a template for structuring their specification. These templates only provide a textual description of the proposed solution. However, a more precise description is required to successfully integrate them into a MDE development process. In this work, the following template has been used for describing the Web 2.0 patterns:

- Problem: a brief description about the problem that the pattern solves.

- Contexts of use: scenarios where the pattern is recommended to be applied. Specific constraints that advise against the use can be documented as well.
- Solution: a textual description that introduces in detail how the pattern solves the stated problem. Technological details and variations over the generic behavior of the pattern can also be introduced.
- Rationale: Why the solution provided by the pattern is useful to improve the user involvement. The advantages of the Web 2.0 pattern with regard to other solutions can also be stated in this section.
- Real examples: several examples of Web 2.0 applications where the pattern has been applied for a specific purpose. This section must optionally include a screenshot of an application that uses the social pattern or sample code.

These five sections are common to previous analyzed works [6][8] and traditional literature about patterns [9]. However, in our pattern template two more sections, which are not defined in previous works, have been introduced. The goal of these sections is to document the Web 2.0 pattern precisely by means of conceptual models. Hence, the semantic gap between the solution representation at the conceptual level and the solution to be implemented is reduced. In order to provide a generic approach, the selected notations to define these models are not linked to a specific method. The two proposed conceptual models are:

- Functionality Model: this model is defined using a UML Class Diagram that describes the different classes, attributes, relationships and operations that implement the Web 2.0 pattern. Hence, this model provides a static view that defines the objects to be created when the pattern is applied. The UML Class Diagram has been selected because it is the most common notation for representing functionality and data.
- Interaction Model: this model provides a description of the interactions performed between the user and the system when the pattern is applied. In the context of this work, the interaction is defined as the communication flows between the user and the system by means of an abstract interface. Concur-Task Trees [16] (CTT) have been selected to create this model, since it is a widely accepted notation to define the interaction in the HCI community.

In order to illustrate the template in detail, a Web 2.0 pattern is described using our proposal: the Quick Comment pattern. Though in this work, CTT and class diagrams have been used to document the patterns, it is worth mentioning that other suitable standard notations can be used for the same purpose.

A. Quick Comment Pattern

Problem: the user wants to share their point of view regarding some specific content, using a very simple process.

Contexts of use:

- To provide textual feedback about some specific content.
- To broadcast textual information about personal thoughts (tweets).
- To provide a personal evaluation or rating about the content

Solution: the pattern is implemented as a text component for introducing the comment below the related content, and one submit button that publishes the introduced text. When the user is registered, the comment is published along with some user information, like the user alias.

Rationale: this pattern is often used in Web 2.0 applications to gather users' feedback. The process to introduce the comment is performed in the same page as the related content and its simplicity avoids the user to give up. Another advantage of the pattern is that the comment publication takes place immediately. All the mentioned advantages provide a very simple interaction, which significantly encourages the user participation.

Real example: this pattern is applied in the Washington Post (http://www.washingtonpost.com) website, where there is a text box below each article to introduce comments. This example also supports the creation of recommendations or replying a previous comment.

Conceptual models: Figure 1. (up) shows the functionality model for this pattern represented with a class diagram. The classes that made up the model represent the information about the comment. If the comment is not anonymous, there is a relationship with the user who creates it; optionally, the comment can also be related to a specific content or to another comment. The model not only supports the creation of new comments (postNewComment operation), but also supports comments modification (edit operation), comments replying (replyTo operation) and subscription for automatically receiving responses (subscribe operation).

Figure 1. (bottom) shows the CTT that represents the corresponding interaction model. The first mandatory operation is to create a new comment that establishes a trigger for the subscriptions ("Create Comment" task). Any subsequent operation is represented by the abstract task "Comment Operations" and it can be repeated indefinitely. The tasks "Reply comment" and "Edition" can be used to modify a comment and save the changes using the task "Store comment modifications". When the user is subscribed to a comment because of the execution of the task "Subscription", two additional system tasks are executed: one task to store the subscription and the other one to send the information related with the subscription.

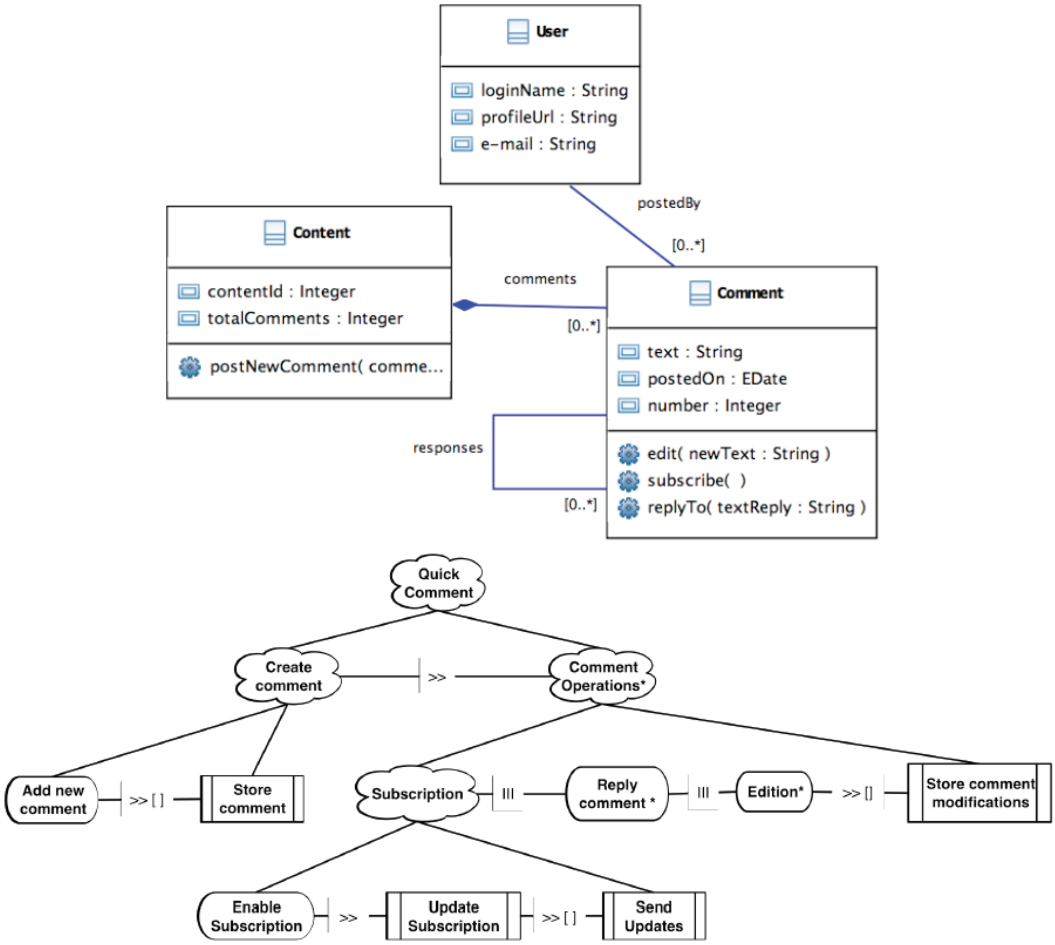

Figure 1. Functionality and Interaction Models for the Quick Comment Pattern

IV. INTEGRATING WEB 2.0 PATTERNS INTO AN MDE METHOD

In the previous section a Web 2.0 patterns has been documented using the proposed pattern template. The main contribution of this work is to document and model these patterns and to provide a strategy to include them in an MDE method.

The modeling phase of Web Engineering methods is defined using a set of modeling elements that made up the method metamodel. When the analyst creates instances of these modeling elements, a model that represents the application is specified. The current Web Engineering methods models are expressive enough to represent the functionality of the most common Web 2.0 patterns. However, because these methods do not provide any reuse mechanism, analysts must define the same models continually for specifying common solutions. As Web 2.0 patterns are fairly frequent, a solution for improving MDE methods is to include them in the modeling phase.

A common approach for extending a MDE method expressivity is the introduction of new modeling elements. In the context of Web 2.0 patterns, this approach has a main drawback. With the purpose of representing each pattern, at least, one new modeling element is introduced. Because there is not an intermediate model between that pattern modeling element and the code generation process, the pattern transformation to code has to be specified and implemented into the MDE method. Furthermore, this model transformation is far from being trivial. The main reason is because Web 2.0 patterns abstract a complex solution, thus there is a wide semantic gap between the modeling entity and the code to be generated.

In our view, a more suitable solution to this issue is the reuse of previous modeling elements for representing the pattern transformation. Hence, the current modeling entities are used as an intermediate model and the complexity is managed at the modeling level. Furthermore, applying this approach, the previous code transformation rules can be reused.

In the context of this work, a Web 2.0 pattern is represented as a model defined using the modeling elements of a MDE method. This method-specific model must be compliant with the interaction and functionality model that represent the

pattern. Our approach is based on the following mechanism: when the analyst uses the modeling element that refers to a Web 2.0 pattern, the underlying model, which is abstracted by the pattern, is used as a substitute. The main advantage is that the complexity of that underlying model is hidden to the analyst. A similar mechanism has been put in practice in the Rational Architect tool [18] in order to introduce model-based patterns into development processes. The ideas of that work are applied in our proposal for integrating Web 2.0 patterns within MDE methods.

Because each MDE method is based on their domain-specific models, obviously an all-encompassing solution cannot be defined. For that reason, each pattern has a specific model for each MDE method. To deal with this issue, we propose a generic strategy that simplifies the introduction of Web 2.0 patterns in a MDE method. This strategy is made up of four steps:

1. Extension of the MDE method semantics: The first step is to check whether or not the current modeling elements of the method support the expressivity required by the Web 2.0 pattern. This check is mandatory because Web 2.0 patterns are built using these modeling elements. The analysis must consider every modeling element that is useful to support both the functionality and the interaction concerns. Therefore, the functionality and interaction model proposed in the pattern template are used as guidelines to perform this check. In case the analysis concludes that current modeling elements do not support the expressivity, the MDE method models must be previously updated.

2. Redefining the method metamodel: Next, the MDE method metamodel must be extended to support the definition of Web 2.0 patterns. To achieve this goal, the metamodel must contain a modeling element that represents each Web 2.0 pattern to be introduced. Additionally, the pattern must be related by means of a unary relationship to at least another modeling element of the metamodel. This relationship defines the methodological extension point from which the pattern is applied (See Figure 2. up). As the figure shows, the pattern element is related to the modeling element "C" from the metamodel. Therefore, when analysts create a new model, the pattern can be applied over the instances of the modeling element "C". Optionally, the new pattern element can include metamodel attributes and relationships with the rest of the metamodel elements. Both attributes and relationships are used as additional information, which are used in the modeling phase, to configure the final implementation.

3. Generation of the pattern model: Once the method supports the required expressivity, Web 2.0 patterns must be defined as a model made up of the modeling elements of the method. This step defines the underlying model that is actually used when the pattern is applied. The underlying model is precisely defined using a M2M transformation. The purpose of the transformation is to substitute the modeling element that represents the pattern by the corresponding modeling elements of the method. The transformation proposed is also known as refining, because both target and source metamodels are the same. An advantage of using a M2M transformation is the possibility of defining different model configurations according to attributes of the pattern. For instance, in the case of the Quick Comment pattern, a Boolean attribute can be defined in the pattern for enabling the edition of comments.

Figure 2. (bottom) shows this process graphically. The analyst creates an M model in which the modeling element of the pattern is used (for instance, a class that represents the "Share content" pattern). Then, the model M is transformed into another model M' by means of a refining M2M transformation. The resultant model M' is then used by the current set of M2C transformations to generate the pattern functionality. Applying this approach, the M2C transformation rules do not change because the new pattern modeling entities are not used in this last transformation.

4. Tool support: Finally, the tool that supports the MDE method must be extended in order to include the Web 2.0 patterns. This step also takes into account the definition of a textual or graphical notation for specifying the patterns. Whether new semantics have been included in the first step, the M2C transformation rules must be modified as well.

As proof of concept, this strategy has been applied in the OOWS 2.0 Web Engineering method. In the next subsections, the OOWS 2.0 method is briefly described, and this strategy is illustrated with the integration of the Quick Comment pattern.

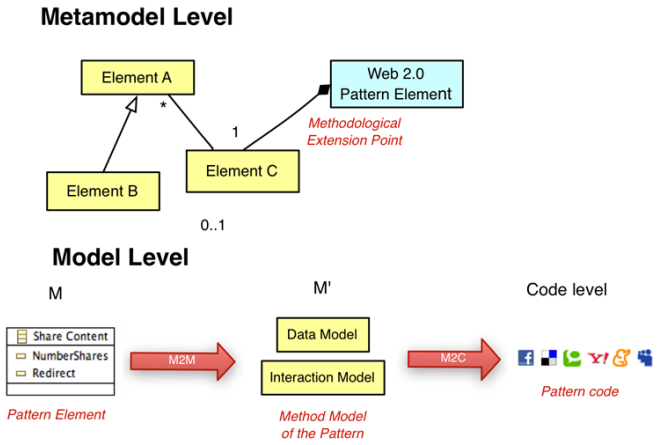

Figure 2. Overview of the proposed strategy

A. *The OOWS 2.0 Web Engineering method*

OOWS (Object-Oriented Web Solutions) [7] is a Web Engineering method that provides methodological support for Web application development. This method has been developed as an extension of OO-Method [15], an automatic code generation method. Both OOWS and OO-Method are examples of MDE methods because conceptual models are considered key elements in all development steps. Software products are generated from a system conceptual specification by means of four complementary OO-Method models and two OOWS models, which capture the data, the business logic and the interface requirements of a given Web Information System.

OOWS 2.0 has been recently developed in order to support the specific features of Web 2.0 applications. This evolution of the method basically extends OOWS at the modeling phase by providing the following set of models:

- Abstract Interaction model: this model describes the interaction between users and the system.
- Web 2.0 patterns model: this model introduces a set of Web 2.0 patterns following the guidelines explained in this work.
- RIA Interface model: this is an optional model to specify a user interface generated using a RIA technology.

OOWS 2.0 modeling phase is made up of four activities. Figure 3. shows the relationship between the different method activities for the OOWS 2.0 modeling phase (left) and the associated models (right) required in each one. The first activity (OO-Method Modeling) is the definition of the models that describe the system data and functionality. This activity is also shared with the previous version of the method.

The next activity of the method is the Interaction Modeling. The key element of this activity is the Abstract Interaction Model. This model represents the interaction between the users and an Information System without taking into account technological details. This model is made up of several modeling elements. First, real users that interact with the system are represented as a User Diagram. Next, the abstract tasks that the users can carry out with system, for instance add a new customer to the system, are represented as Interaction Contexts. The relationship between Users and Interaction Contexts is described using an Interaction Map. This map describes which Interaction Contexts are accessible to a specific User and the interaction path or tasks to be completed in order to execute them.

With the goal of specifying the interaction, the model introduces the concept of Abstract Interaction Units or AIUs. Each Interaction Context is composed by several AIUs that describe the represented interaction. An AIU is defined as an interaction view over the state of the Information System objects. There are two types of AIUs: (1) Population AIU, which retrieves a chunk of data to interact with, and (2) Service AIU, which represents an interface to input the arguments for a service execution. Therefore AIUs represent two main interactions: data retrieval and functionality execution.

Since all the interaction specification with a system cannot be defined using only the Population and the Service AIUs, Auxiliary Interaction Patterns (AIPs) are introduced to constraint the behavior and/or to refine more accurately these two main interactions. Examples of such patterns are “Filter”, which is used to constrain the data to be retrieved, or “Validation Rule”, which defines a Boolean rule that must be accomplished by the input data. Currently, the Abstract Interaction Model includes thirteen of these patterns providing a wide array of interaction expressivity. Further details about the Abstract Interaction Model are discussed in [21].

The next activity is the Web 2.0 Patterns modeling. This activity is directly related with this work. Finally, the details about the RIA Interface Modeling activity, which addresses the User Interface concern, can be consulted in [23].

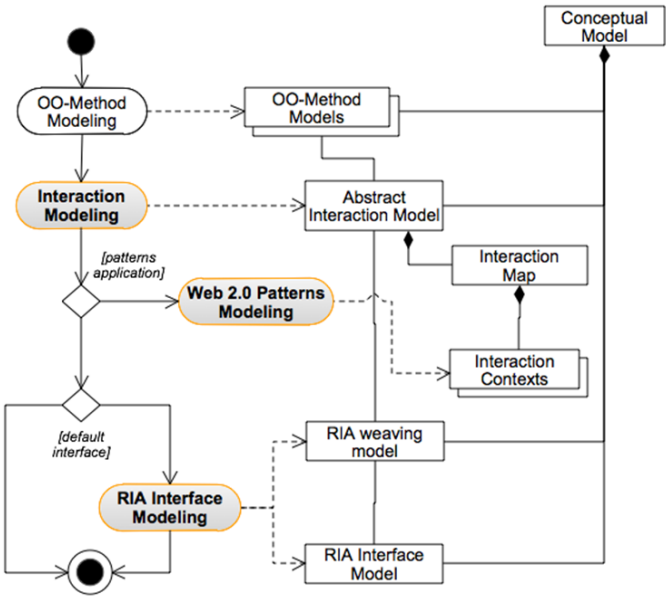

Figure 3. OOWS 2.0 Modeling Phase

B. *Introducing the Quick Comment pattern into OOWS 2.0*

In this section we explain how the Quick Comment pattern is integrated into the OOWS 2.0 method, according to the strategy presented above. With the purpose of illustrating the strategy, a scenario has been selected from the 23andMe website (http://www.23andme.com): a personal genetics Web 2.0 portal that provides knowledge about the genetic information of registered users. In this website, a registered user can participate in a regular forum called “23andMe Community” to discuss about its personal genetics. This scenario provides the interaction for discussing about several genetic topics in a collaborative way. Each thread is created by one user and is related to one topic discussion and to one tag. Registered users can also reply to one thread any number of times and can add threads to a favorites list. Therefore, this scenario uses clearly the semantics of the Quick Comment pattern.

The **first step** for applying the integration strategy is to check, whether or not, the scenario can be modeled using the current semantics of OOWS 2.0. With this goal in mind, the illustrative scenario has been modeled using the Abstract Interaction Model. Semantics are quite straightforward in this case; hence there is no need to add additional expressivity.

The scenario is represented in the Interaction Context “Discussion Threads” shown in Figure 4. Discussions to be shown to the user are represented as a Population AIU. This AIU is made up of population classes that represent a view over the corresponding OO-Method Object Model, a variation of a UML Class Diagram which specifies the system functionality. This view is represented graphically as stereotyped classes with the «manager» or the «complementary» keyword with the set of attributes to be shown.

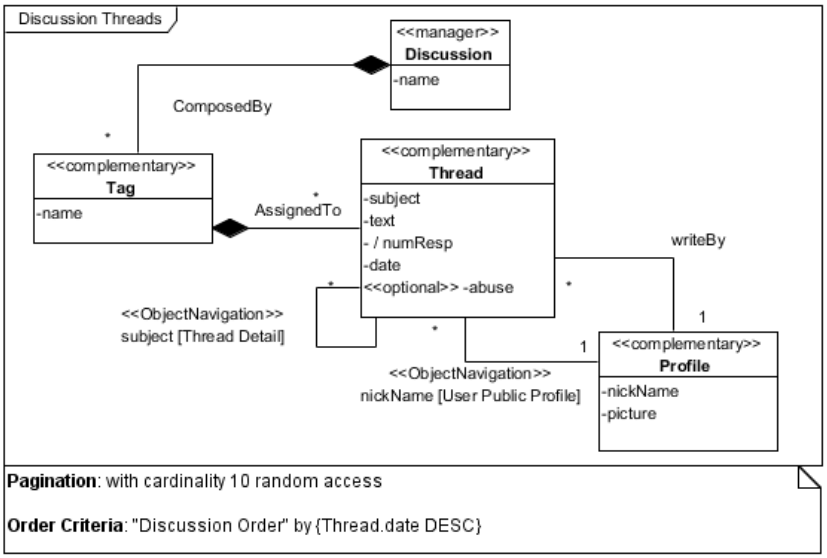

Figure 4. Interaction Context for describing discussion threads

The Population AIU represents the next retrieval of data: one discussion is composed by one or more tags; a thread is assigned to one tag and written by a user which is represented by the profile class; a thread can optionally have any number of replies, each one is written by on user; one user can have any number of threads in her/his favourite list.

The Interaction Context also includes four AIPs: 1) a Pagination AIP to show the discussion in groups of ten threads, 2) an Order Criteria AIP to sort the discussions according to the thread date, 3) and Object Navigation AIP to navigate to the detail of a thread, and 4) an Object Navigation AIP to navigate to the user profile information.

The **second step** is to specify the methodological extension point from which each pattern is included. The Quick Comment pattern must be related to a class from which the users will add new comments. This fact implies that the methodological extension point for this pattern is the "Population Class" modeling element. Figure 5. shows the extension point in the OOWS 2.0 metamodel. Two new relationships are defined: 1) the "contentId" relationship that is related to an attribute which is used as comment identifier, and 2) the "content" relationship that relates the pattern with the class which represent the content to be commented.

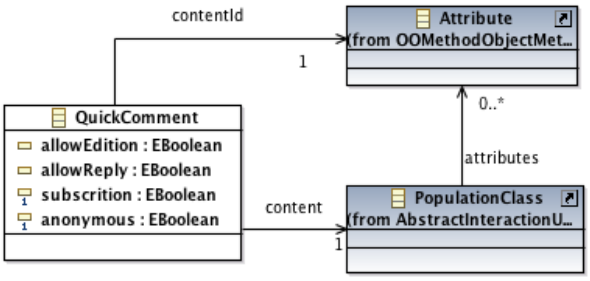

Figure 5. Quick Comment pattern extension point

Additionally, the pattern specification includes four attributes for configuring the specific behavior. The first two "allowEdition" and "allowReply", activate the mechanisms to edit and to reply to comments. The "subscription" attribute, enables the reception by users of new replies to their comments. Finally, the "anonymous" attribute, enables the creation of comments by non-registered users.

Figure 6. (up) shows an example of an Interaction Context in order to reply to the threads created in the proposed scenario. This Interaction Context is reached throughout an Object Navigation AIP from the "Discussion Threads" context. The selected notation to represent the pattern is a stereotyped UML class with the word pattern, linked to the class that represents the content to be commented. In order to create the replies to a thread, the pattern has been applied over the class "Thread". The result is the ability for associating new comment instances to this class that represent the replies.

The **third step** of the strategy is the generation of the pattern model. When the analyst instantiates a Web 2.0 pattern, what really happens is that a more complex model is generated at the background. That generated model (M' model in Figure 2.) must be based on the functionality and interaction models proposed in the Web 2.0 pattern template. Therefore in this third step, functionality and interaction models must be translated to the corresponding modeling elements of the OOWS 2.0 method. As the OOWS 2.0 counterpart of the functionality model resembles the UML Class Diagram, this translation is straightforward.

To perform the mappings between the interaction model and the OOWS 2.0 models, the semantics of the CTT tasks must be analyzed in detail (see an example of CTT in Figure 1. bottom). On the one hand, interactive tasks (ellipse boxes) represent the user-system interaction, for that reason they are described by means of AIUs and the corresponding AIPs. On the other hand, computer tasks (rectangular boxes) represent a processing information/functionality of the system; as a consequence they are mapped to a specific operation of the

OO-Method Object Model. Finally, abstract tasks (cloud icon) are a composition of the previous tasks, thus a specific mapping is not required. Next, the transformation process for the Quick Pattern is discussed in more detail.

Figure 6. (bottom) represents the resulting OOWS 2.0 model (M') when the Quick Comment pattern is applied The first level of the Quick Comment CTT (See Figure 1.) is made up of two optional (|| operator) abstract tasks. Each task will be represented as a Service AIU, in which the tasks of the sub-trees are performed. Figure 6. (bottom) shows the generated OOWS 2.0 model entities (depicted with a wider line) for the "Create Comment" and "Comment Operations" abstract tasks. The "Add new comment" task is represented by the Service AIP "postNewComment" defined over the "Thread Class". The "Reply comment" task is represented by the new class "Comment" and the "replyTo" Service AIP that stores the new comment associated to a thread. Besides the pattern application relates each new comment to a user and to its profile URL.

The pattern instantiation provides the definition of three Service AIPs, three additional population classes and an additional Object Navigation AIP using a single modeling element. As a consequence the complexity perceived by the analyst is reduced. It is also worth mentioning that in this specific pattern instantiation, the edition and subscription attributes are set to false. Therefore, a more complex model can be translated from the pattern application.

Regarding the **four step**, tool support, the underlying M2M transformation has been defined using ATL [2], because this language is compatible with the OOWS 2.0 metamodel defined in the Ecore language. The result of this M2M transformation is the input for the OOWS 2.0 code generation process. The specific details about this transformation and the code generation process are out of scope of this paper.

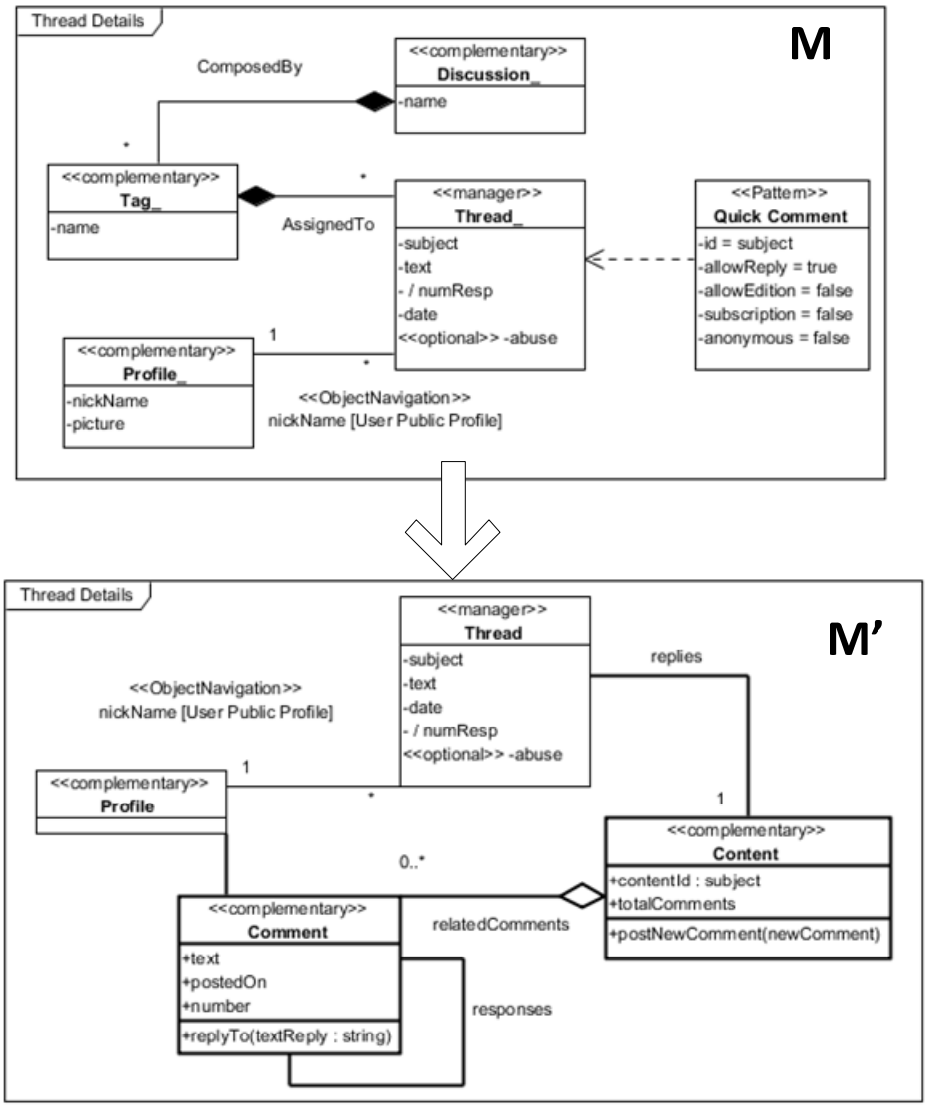

Figure 6. Generation of the Quick Pattern underlying Model

V. CONCLUDING REMARKS

This work has presented an approach to include in MDE methods the best practices of the Web 2.0 development. The two main contributions of this work are: 1) the definition of the user involvement in the Web 2.0 content creation by means of model-based patterns, and 2) a strategy to extend the current MDE methods with this novel Web 2.0 pattern concept. To our knowledge, the research community does not focus yet on these issues from the perspective proposed in this work. However, in the Web 2.0 domain where heterogeneous concerns are involved, the use of Web 2.0 patterns provides interesting benefits.

Several lessons have been gathered from the presented work. Firstly, the use of conceptual models simplifies the translation of Web 2.0 patterns to the specific models of the method. Another interesting lesson is that several Web 2.0 patterns can be modeled with the current Web Engineering methods because they have enough expressiveness. Consequently using patterns, we can benefit from a mechanism to reuse the best practices in both models and applications. Additionally, Web 2.0 patterns must be described using conceptual models based on standard or well-known modeling languages. Therefore the knowledge can be shared among different MDE methods.

The last lesson is with regard the effort to implement the proposal in a specific MDE method. Methods, whose conceptual models have poor expressiveness to represent interaction and functionality, will require the extension with new conceptual primitives. Therefore, it is important to evaluate the expressiveness of the MDE method to use and the extension cost, before applying our proposal.

Though the presented approach provides a noteworthy improvement, there are some issues that must be considered as well. Firstly, a Web 2.0 pattern may involve the abstraction of a very complex model. As a consequence, the M2M transformation to be defined is not a straightforward task. Additionally, for each pattern a new metamodel element must be defined. Therefore, the modeling language provided to analysts may become more difficult to understand. Nevertheless, the main concern is that the method expressivity constrains whether or not the pattern can be introduced. If the models of the method cannot be adapted or extended to support the expressivity required, the inclusion may not be feasible.

Future works will address the definition of an extensive Web 2.0 pattern library. The OOWS 2.0 method will be extended to support this library and to generate code from the models defined as Web 2.0 patterns. Finally, an empirical study must be carried out to validate the improvement of the user experience and the analyst modeling effort.

ACKNOWLEDGMENT

This work has been developed with the support of, MICINN (PROS-Req TIN2010-19130-C02-02), GVA (ORCA PROMETEO/2009/015), and co-financed with ERDF.

REFERENCES

[1] Alexander, C., Ishikawa, S. & Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press (1977)

[2] Atlas INRIA & LINA: ATL User Guide http://wiki.eclipse.org /ATL/User_Guide Last Visit: December 2011 (2011)

[3] Bozzon, A., Comai, S.: Conceptual Modeling and Code Generation for Rich Internet Applications. 6th International Conference on Web Engineering (ICWE), California, United States (2006)

[4] Budinsky, F., Merks, E., Steinberg, D., Ellersick, R., Grose, T.J.: Eclipse Modeling Framework. Addison-Wesley Professional (2003)

[5] Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a modeling language for designing Web sites. WWW9 Conference, Amsterdam, (2000) 137 - 157

[6] Crumlish, C., Malone, E.: Designing Social Interfaces: Principles, Patterns, and Practices for Improving the User Experience. O'Reilly Media (2009)

[7] Fons, J., Pelechano, V., Albert, M., Pastor, O.: Development of Web Applications from Web Enhanced Conceptual Schemas. ER 2003,Vol. 2813. LNCS Springer(2003) 232-245

[8] Fraternali, P., Tisi, M., Silva, M., Frattini, L.: Building Community-based Web Applications with a Model-Driven Approach and Design Patterns. www.webml.org/webml/upload/ent5/1/CommunityPatterns-final.pdf. Last Visit: November 2011

[9] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addyson Wesley (1995)

[10] Koch, N., Pigerl, M., Zhang, G., Morozova, T.: Patterns for the Model-Based Development of RIAs. Web Engineering (2009) 283-291

[11] Levendovszky, T., Lengyel, L., Mészáros, T.: Supporting domain-specific model patterns with metamodeling. Software and Systems Modeling 8 (2009) 501-520

[12] Linaje, M., Preciado, J.C., Sánchez-Figueroa, F.: Engineering Rich Internet Application User Interfaces over Legacy Web Models. IEEE Internet Computing (2007) 53-59

[13] Murugesan, S.: Understanding Web 2.0. IT Professional 9 (2007) 34-41

[14] Oreilly, T.: What is Web 2.0?. Design Patterns and Business Models for the Next Generation of Software. Vol. 2008 (2005)

[15] Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice. A Software Production Environment Based on Conceptual Modeling. Springer-Verlag, Berlin Heildeberg (2007)

[16] Paternò, F.: ConcurTaskTrees: An Engineered Notation for Task Models. In: Diaper, D., Stanton, N., Stanton, N.A. (eds.): The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Associates, London, United Kingdom (2004) 483-501

[17] Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.): Web Engineering: Modeling and Implementing Web Applications. Springer (2008)

[18] Swithinbank, P., Chessell, M., Gardner, T., Griffin, C., Man, J., Wylie, H., Yusuf, L.: Patterns: Model-Driven Development Using IBM Rational Software Architect. IBM Redbooks (2005)

[19] Tidwell, J.: Designing Interfaces. O'Reilly Media (2005)

[20] UI Patterns library: http://ui-patterns.com/ Last visit: December 2011

[21] Valverde, F., Panach, I., Aquino, N., Pastor, O.: Dealing with Abstract Interaction Modeling in an MDE Development Process: a Pattern-based Approach. In: Macías, J.A., Granollers, T., Latorre, P. (eds.): New Trends on Human-Computer Interaction. Springer, London (2009) 119-128

[22] Valverde, F., Pastor, O.: Applying Interaction Patterns: Towards a Model-Driven Approach for Rich Internet Applications Development. In: Gaedke, M., Bieliková, M. (eds.): 7th International Workshop on Web-Oriented Software Technologies. Vydavateľstvo STU, New York, United States (2008) 13-18

[23] Valverde, F., Pastor, O.: Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach. Web Information Systems Engineering - WISE 2009 (2009) 131-144

[24] Valverde, F., Valderas, P., Fons, J., Pastor, O.: A MDA-Based Environment for Web Applications Development: From Conceptual Models to Code. In: Brambilla, M., Mendes, E. (eds.): 6th International Workshop on Web-Oriented Software Technologies (IWWOST). Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, Como, Italy (2007) 164-178.

[25] Sendall, S., Kozaczynski, W.: Model Transformation: The Heart and Soul of Model-Driven Software Development. IEEE Software 20 (2003) 42-45.

[26] Hailpern, B., Tarr, P.: Model-Driven Development: the Good, the Bad, and the Ugly. IBM Syst. J. 45 (2006) 451-461.

[27] Silva, P.P.d., Paton, N.W.: User Interface Modeling in UMLi. IEEE Software 20 (2003) 62-69

[28] Noda, T., Helwig, S.: Rich Internet Applications - Technical Comparison and Case Studies of AJAX, Flash, and Java based RIA. Best Practice Reports University of Wisconsin-Madison, Vol. 2008 (2005)

[29] Schroth, C., Janner, T.: Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services. IT Professional 9 (2007) 36-41

[30] Welie, M.v., Traetteberg, H.: Interaction Patterns in User Interfaces. 7th. Pattern Languages of Programs Conference, Illinois, USA (2000)

[31] Yahoo design pattern library: http://developer.yahoo.com/ypatterns/ Last visit: december 2011