Applying Interaction Patterns: Towards a Model-Driven Approach for Rich Internet Applications Development

Francisco Valverde, Oscar Pastor
Department of Information Systems and Computation
Universidad Politécnica de Valencia, Spain
{fvalverde, opastor}@dsic.upv.es

Abstract

Recently, a wide array of Web Applications has evolved to Rich Internet Applications (RIAs). This new application paradigm emphasizes the use of client-side technologies in order to provide more responsive and interactive Web Interfaces. The main contribution of this work is an Interaction Model to specify the new semantics to deal with the Model-Driven RIA development. This model is made up of Interaction Patterns that describe, from a conceptual point of view, a generic solution for a common user-system interaction. Following the HCI principles, this model has two views: 1) an Abstract view made up of Abstract Interaction Patterns that describe the interaction without taking into account technological details and, 2) a Concrete view made up of RIA Interaction Patterns which specify the new interaction and interface requirements. Applying this Interaction Model within a Model-Driven software process, a functional RIA can be obtained. Finally, to illustrate this approach, two RIA Interaction Patterns are described.

1. Introduction

Latest Web Applications are providing rich interfaces with a more intuitive interaction experience for the end-user. The traditional Web Application paradigm implies that for each user interaction (for instance, a button click) the client browser must request a new Web Page. In order to improve the interaction, a reasonable approach is to request only the new content required and to avoid the full page refreshing. As a result, from the end-user perspective, the interaction is automatically performed. The Web Applications that use this new paradigm are called Rich Internet Applications (RIAs) [2]. Around the RIAs development field have arisen several technologies (Adobe Flex, Microsoft SilverLight, OpenLaszlo etc.), which use AJAX, REST Services or JavaScript frameworks. Using these new development technologies, Web Applications with richer client UIs have been developed. For instance, the Google Maps (maps.google.com) application is a well-known example that uses the RIA paradigm to request on demand map pictures.

In recent years, Web Engineering Methods [8] have improved Web Application development applying the Model-Driven Engineering principles. These methods have provided promising results to enhance the development of traditional Web Applications. However, they do not properly support the interaction and presentation requirements [1] and the behavior expressiveness [3] required by RIAs. Firstly, the interaction between the users and the system, which is mainly implemented in the client-side, is not described with the same detail than the system functionality and navigation. And secondly, there is not a clear distinction between the interaction, which describes the set of actions that the user can perform together with the Web System, and the interface, which constitutes the graphic elements that support that interaction (buttons, grids, multimedia components, etc.).

Regarding the HCI community, several approaches have been proposed to specify the interaction between the user and the system. A common agreement [12] is to define two abstraction views in order to model the UI: an Abstract view to describe interaction without taking into account technological issues and a Concrete view to deal with the specific technological requirements. This approach is more flexible than the definition of a single Presentation Model that has traditionally been proposed by several Web Engineering Methods.

The main objective of this work is to define a Model-Driven approach to support the RIA development. Since the main improvements in RIAs are defined in the Presentation Layer, the models to introduce must address the RIA interaction specification. Taking into account the HCI principles, an Interaction Model is proposed that defines the RIA Presentation layer from the user-system interaction point of view. In order to define this Interaction Model, the Interaction Pattern (IP) concept has been introduced. An Interaction Pattern describes a solution for

a common user-system interaction in terms of the Problem Space (Conceptual Models)

This paper is focused on the Concrete view in which the semantics to produce RIA interfaces are introduced. Therefore, a RIA Interaction Pattern describes precisely an interaction which is used in the RIA domain. Using these patterns as guidelines, a set of transformation rules can be defined to produce the RIA interface code. To illustrate this approach, two common RIA Patterns (Auto Complete and Information Summary) are defined using the Interaction Model proposed.

The rest of the paper is structured as follows: Section 2 describes the background of this work and the related work regarding RIA development in the Web Engineering community. Section 3 presents an overview of the Model-Driven approach proposed. Next, section 4 describes briefly the Abstract view of the Interaction Model. Section 5 is focus on the description of the RIA Interaction Model. Finally, the concluding remarks are stated.

2. Background and Related Work

The use of patterns at modeling level is an approach that has been previously applied in the context of our research group. For instance, the Presentation Model for the OO-Method [9] software production method was defined using the JUST-UI [11] approach. JUST-UI proposes a language made up of Conceptual Modeling patterns for the specification of UIs and code generation techniques. The work presented in this paper, reuses the concepts defined in the JUST-UI approach and previous experiences in the OOWS Web Engineering method [10], to define the Abstract View of the Interaction Model. Therefore, the main contribution of this work is to extend those previous approaches, with a set of Interaction Patterns to support RIA development.

Several works in the Web Engineering field have proposed methodological extensions to face the RIA development. Firstly, Bozzon et al. [5] extend the WEBML method to support the RIAs specification. The proposed extensions allow the definition of which data, operations, hypertext links and page content must be processed by the client-side. In the context of the OOHDM method, Urbieta et al. [6] propose and approach to designing RIA interfaces, dealing with the separation of the interface structure and behavior. This work uses an aspect-oriented perspective, to combine different concerns related to the interface composition.

In contrast, several works have addressed the RIA development using HCI principles. Firstly, a metamodel for defining RIA UIs from an Abstract Interface Model is proposed in [4]. And secondly, the RUX-model [7] proposes how to define at Concrete level the time-related behaviors (Temporal Presentation) and the event-response actions (Interaction Presentation) to define RIA interface components.

The approaches mentioned above, propose mainly solutions for supporting the data synchronization and the specification of the client-side behavioral aspects. Though these approaches point out a key issue, our approach must also address the richer RIA UIs, which are defined using new graphical widgets, and the complex interactions between the server and the client rich interface. These new improvements have widely been described by the RIA development community using UI Design Patterns [15]. Therefore, the final goal of this work, is to define how the knowledge from those patterns, can be introduced in a Model-Driven development process.

3. A Model-driven approach to produce RIAs

When an Information System is specified, the interaction is usually described in different models. For example, several Web Engineering methods have described navigational and data retrieval interactions inside the Navigational Model. However, some aspects related with the interface of those interactions, are described in a Presentation Model in where aesthetics requirements (colors, fonts, etc.) are also included. This approach to define a RIA Presentation Layer is not flexible enough, because implies to introduce changes in different models related to different concerns. Our approach to model the RIA Presentation Layer introduces a single model, the Interaction Model, to define all the interaction requirements.

In the context of this work, the interaction is defined as the actions that take place between a human user and the software system in order to perform a task. Therefore, the interaction comprises both the UI specification (the interface components) and the integration with the system functionality (the UI-system communication interface). To model these two requirements, an Interaction Model is introduced as Fig.1 illustrates. The aesthetic requirements of the UI are not considered in this model.

The Abstract view of the Interaction Model is made up of two main elements (Fig. 1 up): tasks and Abstract Interaction Patterns (AIPs). The different users have visibility over a set of tasks that describes the interactions available. The basic units to specify these tasks are the AIPs. An AIP models two kinds of user-system interactions: population retrieval or service execution. It is important to mention that the specific technological details (UI components, events) are not defined with these patterns.

The AIPs are specialized in Concrete Interaction Patterns that define the Concrete view of the Interaction Model. These Concrete patterns address the technological details related to the interaction: the specific interface components, the available events, the communication mechanisms etc. Different sets of Concrete IPs can be

defined to specify the requirements of several technological platforms. In this work, the RIA Interaction Patterns, which extend the AIPs with semantics from the RIA domain, have been introduced. However, the same approach can be used to define HTML IPs (as Fig. 1 illustrates) or Mobile device IPs. This approach is very flexible because the analyst can firstly describe abstract interactions and later, those interactions can be specialized in terms of several technological platforms.

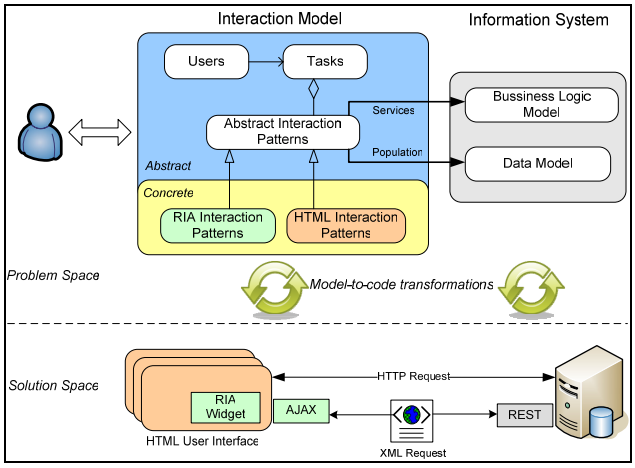

Figure 1. Model-driven approach overview.

From both the Abstract and the Concrete view of the Interaction Model, a set of model-to-code transformations generate the final client UI. This code can be divided into two parts: 1) the final RIA Interface implemented using a JavaScript UI framework (EXT-JS, Google Web Toolkit) or a RIA technology (Adobe Flex, Open Laszlo, etc.) and 2) a business façade which communicates with the server- side by means of XML Messages using AJAX and REST Services. Nevertheless, some interoperability code should be added to the business façade.

In order to integrate this approach with the Information System two requirements must be satisfied:

1. The Information System must have been described using a Conceptual Model that represents the underlying functionality.
2. The Business Logic must be accessible by means of an interoperable XML API.

In the OO-Method development process these two requirements have been fulfilled. Firstly, the OO-Method is made up of a set of Conceptual Models from which the system functionality can be generated. And secondly, because an approach to produce XML Web Services from those models [17] have been proposed. Therefore, using this Interaction Model inside the OO-Method development process, a final functional RIA can be obtained.

4. The Abstract Interaction Model

Each type of user has a set of tasks that describe the available interactions. For instance, in an e-commerce application, the tasks available may be 'Log in to the system' or 'Checkout the order' among others. These tasks are defined in the Interaction Model using an Interaction Map associated to each user. This map is a directed graph, whose nodes represent the tasks available, and whose arcs represent the transitions between tasks. The two main objectives of the Interaction Map are:

- To define which interactions are available for each user, or in other words, to restrict the access to specific interactions.
- To provide a global view, very close to a Navigational Map, about how the user can access to the system.

From the analysis of several UIs in different platforms and devices, we have detected a set of generic interactions which can be abstracted as technologically-independent patterns. The different tasks can be described as an aggregation of one or more of these Abstract Interaction Patterns (AIPs). Therefore, the main objective of AIPs is to provide a detailed description of each task defined in the Interaction Map. An AIP describes the interaction in terms of two main communication flows between the users and the Information System: a) a population retrieval flow, that provides information from the system to the user, and b) a service execution flow started by the user, which performs a change of the state of the Information System. Considering both flows, two main AIPs can be defined:

- *Population AIP*: it represents an information retrieval from the system. The Population AIP is defined as a view over the model that represents the system data entities, for example an UML Class Diagram.
- *Service AIP*: it abstracts a service (for example a Class method or a Web Service operation) that modifies the state of the system. The *Service AIP* is defined as a view over the arguments from one service. For each argument from the service signature, an *Input Argument AIP* is created to insert the corresponding value.

Since all the interaction specification with a system cannot be defined using only the Population and the Service AIPs, auxiliary AIPs are introduce to constraint the behavior and/or to refine more accurately this two main interactions. Two examples are:

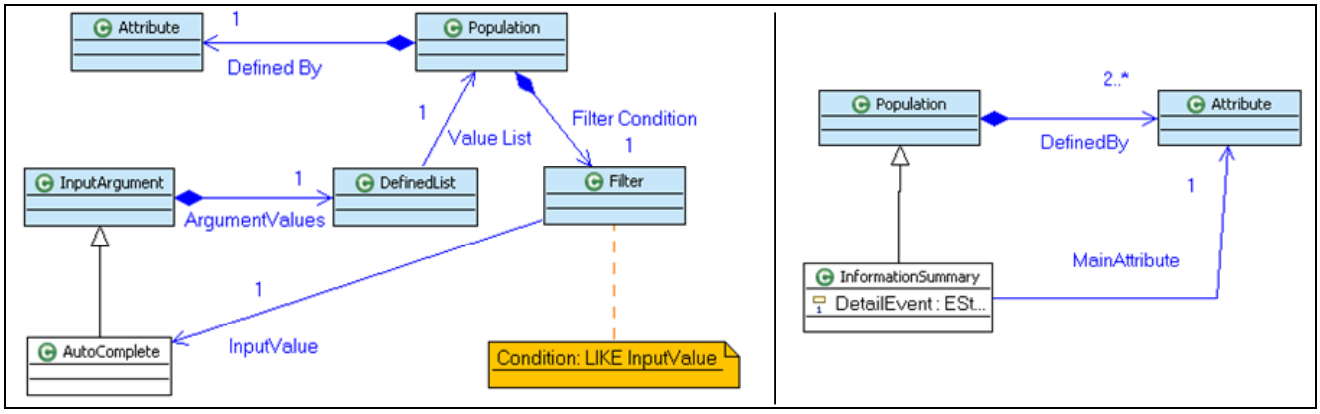

Figure 2. Metamodels for the RIA Interaction Patterns.

- *Filter AIP:* By default, a Population AIP retrieves all the information that is defined by the interaction. This pattern defines a well-formed formula that restricts the information to be retrieved.
- *Defined Selection AIP*: this pattern defines a set of values associated to an *Input Argument AIP*. Hence, the user can only choose, from the provided list, one value to fill the input.

Currently, ten AIPs which can be consulted in [16] have been detected. These patterns define an abstract pattern language to model the interaction. From this Abstract Interaction Model a preliminary and functional UI can be generated. For example, a direct transformation from a platform-independent model (PIM) [10] can be applied to generate a Web Application interface. Nevertheless, to take advantage of the rich features provided by the RIA development technologies, a RIA Interaction Model is recommended.

5. The RIA Interaction Model

The Web Development Community [15] and the UI design literature [13] have defined several patterns to improve the RIA development. These patterns are validated solutions because they have been applied in real-world examples. The proposal of this work is to apply the knowledge described in these patterns in order to define a Concrete view for the Interaction Model: the RIA Interaction Model. As the Abstract view, this model is made up of Interaction Patterns but fitted into the RIA domain.

RIA Interaction Patterns are defined as a specialization from an AIP, with the aim of extending the interaction semantics required for RIA development. In these patterns, technological concepts related to the RIAs are introduced; for example the client-request and server-response flow or which widget implements the interaction. In addition, an AIP can be specialized to different RIA Interaction Patterns that extends the same abstract interaction: for instance, a Population AIP can be represented as a data table, as several dynamic text items or as an editable grid. For that reason, the analyst must select for each AIP, the RIA Interaction Pattern more suitable for the interaction to be modelled.

According to previous works [14], [11], the Interaction Patterns have been defined using a pattern template with these sections: 1) *Problem*: the problem that the pattern is intended to solve, 2) *Context*: in which scenario is recommended to use the pattern, 3) *Solution*: a brief description of the proposed solution and, 4) *Example:* a real scenario to show how the pattern can be applied.

However to apply these patterns in a Model-Driven development process, a more precise description is required. For that reason, three more sections are introduced to the pattern template:

- *Metamodel:* it defines using a metamodelling language the concepts that the pattern abstracts (See Fig 2.). This representation must be used to create specific instances of the pattern.
- *Metamodel Specification:* it describes the different entities, relationships and properties defined in the metamodel. Specific constraints about the metamodel must be explained here.
- *Interaction semantics:* it specifies precisely the interaction expected when the pattern is applied. Therefore, it describes the interface components, the interface events and the communication with the business logic. Additional models, as UML Interaction diagrams, can be used to explain the semantics.

Using this template, the RIA Interaction Patterns can be applied as guidelines to define a Model-Driven code generation process. On the one hand, the metamodel describes the conceptual primitives from which the transformation rules must be defined. On the other hand, the

interaction semantics provide the translation from the metamodel to the expected code. To better illustrate the concepts presented here, two RIA Interaction Patterns are described next.

5.1. Auto Complete Pattern

Problem: the user must select or introduce a text value from a huge defined list.

Context: there is a huge list of possible text values to be chosen by the user and it has to introduce one without mistakes.

Solution: as the user writes the text, a dropdown list shows the values from the list that match with the introduced characters. Then, the user selects the desired value or, considering the reported information, he introduces the correct text value.

Example: in a form for uploading an academic paper, the user must input the author's full name that is stored in the system. The user knows the first name but he does not remember the last name. Applying the Auto Complete pattern, all the system authors that match with the first name introduced by the user will be shown. The figure below shows the expected UI.

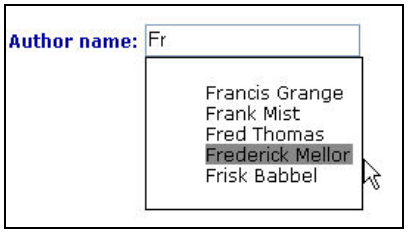

Figure 3. UI for the Auto Complete Pattern.

Metamodel Specification: this pattern is specialized from an *Argument Input AIP* (see Fig. 2 left). To define this pattern a *Defined List AIP* linked to a *Population AIP* is needed to constraint all the possible values that the user can introduce. These values are represented by the only attribute associated to the *Population AIP*. When the user types a set of characters, a *Filter AIP* establishes automatically a string-like condition over the defined *Population AIP*. The filtering result defines the available values for the *Defined List AIP*.

Interaction semantics: the widget that is used for inserting the text value (an input textbox), must raise an event to detect that the user has introduced a set of characters. When this event is detected by the client interface, an interaction requesting the population values is sent to the server. Next, the client receives the data and renders the received values as a drop-down list widget. If the user introduces more characters, the values are filtered in the client interface and no new request is needed. However, if the user deletes the first set of characters, the population request interaction must be performed again.

5.2. Information Summary Pattern

Problem: the user receives high amount of information but he only wants to focus on a single piece of information.

Context: the data shown to the user is composed by several pieces of information with a high density (for instance, large texts). However, the user wants to see only one piece of information at the same time.

Solution: provide a brief summary, i.e. a single text line, for each piece of information. When the user selects this summary, the rest of the information is displayed. If the user selects another summary, the previous information is hidden again.

Example: the user wants to find the latest news about computer technology. Since there are a huge amount of news, to simplify this task only the headlines are shown. When the user selects a particular headline the rest of the information is displayed. The Fig. 4 illustrates this pattern.

Figure 4. UI for the Information Summary Pattern.

Metamodel Specification: this pattern extends a *Population AIP* (see Fig. 2. right) that must have two attributes defined at least: one for describing the summary and another for recovering the detailed information. To specify the summary information, a new relationship (*DefinedBy*) is required between the pattern and an attribute from the *Population AIP*. A *DetailEvent* property is added to define which UI event will trigger the display of the detailed information.

Interaction semantics: The client side must request all the information defined by the *Population AIP*. This information is stored and, for each information item retrieved, only the summary attribute is shown. When the user triggers the event attached to the summary text, the rest of the information will be shown without any additional request. When the user selects another summary, the previous information item must be hidden again.

6. Concluding Remarks

In this work, a Model-Driven approach to improve the RIA development has been presented. This approach introduces an Interaction Model, with uses the concept of RIA Interaction Pattern to capture the new semantics required. The use of Interaction Patterns provides two advantages:

- Since they have been defined using real world examples, the analyst is dealing at modeling level with concepts close to the final UI.
- They provide a mechanism to interaction knowledge reuse. Hence, an abstracted interaction can be imported to another Model-Driven approaches.

With the goal of using industrially the described RIA Interaction Patterns, a Model-based tool to define the Interaction Model and to generate the RIA related code is needed. Further work, will define the tool support for building the Interaction Model presented as well as the model transformation rules. Once the proposed environment has been developed, it should be validated by means of several case studies. These case studies will provide an interesting feedback about the most suitable patterns to implement RIAs.

The final step is to include the proposed Interaction Model inside the OO-Method software generation process. As a result a complete RIA, which satisfies the expected interaction requirements, can be generated.

Acknowledgements

This work has been developed with the support of MEC under the project SESAMO TIN2007-62894.

References

[1] J. C. Preciado, M. Linaje, F. Sánchez, and S. Comai, "Necessity of methodologies to model Rich Internet Applications," in 7th IEEE International Symposium on Web Site Evolution Budapest, Hungary, 2005.

[2] J. Duhl, " White Paper: Rich Internet Applicattions - IDC Report," 2003.

[3] S. Comai and G. T. Carughi, "A Behavioral Model for Rich Internet Applications," in 7th International Conference in Web Engineering Como, Italy, 2007.

[4] F. J. M. Ruiz, "A Development Method for User Interfaces of Rich Internet Applications" DEA Thesis. Université catholique de Louvain, 2007.

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

[6] M. Urbieta, G. Rossi, J. Ginzburg, and D. Schwabe "Designing the Interface of Rich Internet Applications," in Fifth Latin American Web Congress (LA-WEB) Santiago de Chile, 2007.

[7] M. Linaje, J. C. Preciado, and F. Sánchez-Figueroa, "Engineering Rich Internet Application User Interfaces over Legacy Web Models," IEEE Internet Computing, pp. 53-59, 2007.

[8] G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, Web Engineering: Modelling and Implementing Web Applications: Springer, 2008.

[9] O. Pastor and J. C. Molina, Model-Driven Architecture in Practice. A Software Production Environment Based on Conceptual Modelling: Springer, 2007.

[10] F. Valverde, P. Valderas, J. Fons, and O. Pastor. A MDA-Based Environment for Web Applications Development: From Conceptual Models to Code. in 6th International Workshop on Web-Oriented Software Technologies. 2007. Como (Italy).

[11] P. J. Molina, S. Meliá, and Ó. Pastor, "JUST-UI: A User Interface Specification Model" in Proceedings of Computer Aided Design of User Interfaces, Valenciennes, Francia., 2002

[12] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt, A Unifying Reference Framework for multi-target user interfaces. Interacting with Computers, 2003.

[13] J. Tidwell, Designing Interfaces: O'Reilly Media, 2005

[14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software: Addison Wesley, 1995.

[15] Yahoo Design Pattern Library. http://developer.yahoo.com/ypatterns/

[16] F. Valverde, J. I. Panach, and Ó. Pastor, "An Abstract Interaction Model for a MDA Software Production Method," in 26th International Conference on Conceptual Modeling (Posters). Auckland, New Zeland, 2007.

[17] M. Ruíz, F. Valverde, and V. Pelechano. Desarrollo de servicios Web en un método de generación de código dirigido por modelos. in II Jornadas Científico-Técnicas en Servicios Web. 2006. Santiago de Compostela, Spain.