Journal of Web Engineering, Vol. 15, No.5&6 (2016) 412-432

WEB APPLICATION EDITOR: A USER-EXPERIENCE DESIGN FRAMEWORK FOR KNOWLEDGE-INTENSIVE ORGANIZATIONS

ROBERTO PAIANO, ADRIANA CAIONE, ANNA LISA GUIDO, ANGELO MARTELLA, ANDREA PANDURINO

Department of Engineering for Innovation

University of Salento, Lecce, Italy

{roberto.paiano, adriana.caione, annalisa.guido, angelo.martella, andrea.pandurino}@unisalento.it

Received November 30, 2015
Revised March 23, 2016

The User Experience is an important factor for the success of an information system, because it mostly represents the point of contact between user and application. Besides being superior from a graphical point of view, a successfully software product must also be based on a User Experience model characterized by an adequate content schema. The definition of such a schema must aim at the creation of a single point for integrating and accessing the information to present to the user. The idea of supporting the User Experience engineering process, by providing specific interfaces with knowledge bases and Business Process Management solutions, is particularly interesting.

The paper introduces a Model-Driven framework for the User Experience engineering process, based on the Interactive Dialogue Model methodology, along with a Rich Internet Application prototype generator. During the User Experience engineering process this framework also provides specific interfaces with both the domain knowledge base and the Business Process Management solution.

Key words: User Experience, Knowledge Base, Rich-IDM, Model-Driven Engineering, Application Prototyping, Business Process Management
Communicated by: D. Schwabe & S. Murugesan

1 Introduction

Kraft [1] defines the User Experience (UX) as ‘the feelings that the user gets when using a product’, underlining the importance of the same UX for the success of a product. The UX definition partially takes up the Kansei Concept introduced in 1995 by Nagamachi [2] and revised further by Schütte in [3]. Essentially, Kansei can be defined as the ‘user’s subjective impression’ in using a product. From this definition is possible to introduce Kansei engineering, which represents ‘a method aiming at aligning design details in relation to a user’s Kansei in order to determine and evaluate new design solutions’ [4]. The UX significance is still valid in the computer science context, where it also represents one of most important factors for the success of an information system. For this reason, it is

almost essential to include a careful and specific UX modelling phase within the information system design. Abowd et al. [5] affirm: 'Whenever humans interact with computers, directly or indirectly, information, semantics, data, or knowledge is exchanged'. Indeed, the UX of an application represents the real point of contact between the user and the software system. The importance of a good UX design is also emphasized in [6], where the author says that the interface represents an important factor for a successful navigation design. In particular, the interface constitutes the interpreter that guides the user to navigate the contents. So, for a cohesive UX it is necessary to bind the architecture and interface of the logical structure and visual meaning.

Addressing the knowledge acquisition and representing problems using traditional approaches in the context of user interface developing, Peschl and Stary [7] affirm that: 'in order to achieve the goal of developing human-oriented (in contrast to technology-oriented) human-computer interfaces developers have to develop sound knowledge of the structure and the representational dynamics of the cognitive system which is interacting with the computer'.

Indeed, the information context must support the user in an appropriate way, during the course of their work activities. So, such a context not only includes the domain information for the application, but must also be characterized further, using contents that are complementary to the same domain information. Laws and regulations are part of this category of content, which is complementary to the context information. Indeed, laws and regulations represent the legal framework that the company must comply with. In this respect in [8] authors report:

> The ever-increasing obligations of regulatory compliance are presenting a new breed of challenges for organizations across several industry sectors. Aligning control objectives that stem from regulations and legislation with business objectives devised for improved business performance is a foremost challenge. Compliance is defined as ensuring that business processes, operations, and practice are in accordance with a prescribed and/or agreed set of norms.

Using a software product, the user expects to be guided and supported by the system during the course of their work. In particular, the user of a software system must be able to:

- Contribute to the business processes advancement and completion through their activities;
- Access the support contents in an easy and integrated way, in order to consciously perform their activities. In this way, the user can consult all the rules and laws provided by the relevant regulations, before proceeding with the task completion.

The idea of supporting the UX engineering process, by providing interfaces with company systems such as knowledge bases and Business Process Management (BPM) solutions is particularly interesting.

A knowledge base represents a network of concepts and relationships that is modelled by using a standard semantic language. In this case, a concept represents an abstraction of something that exists in the company reality, such as a worker, an event and a regulation. As well as representing a valid solution for the BPM and for the organizational issues, knowledge bases and ontologies can be also used to support the user in a transparent way, during both the design and the run time of a system.

Moreover, such information has to be complex enough to model the system in a reasonable level of detail [9,10].

The possibility to refer to the knowledge base support during the UX modelling of a software system has already been assessed in literature. Buriano et al. [11] report 'the adoption of ontologies for modelling the domain, the context and the adaptation process can contribute to tailor the right information/service to users and thus facilitate the user-system interaction and the system communication with other agents'. For the Knowledge Management System (KMS) Razmerita et al. [12] underline the importance of the knowledge bases stating that: 'Due to their powerful knowledge representation formalism and associated inference mechanisms, ontology-based systems are emerging as a natural choice for the next generation of KMSs operating in organizational, inter-organizational as well as community contexts'.

With regard to the characteristics of the BPM solution user interface, Van Greunen et al. [13] affirm:

> One of the most important components of successful business process management is the group of enterprise software users – those people who interact with the workflows at various points in order to enable a certain task or objective to be completed. Good user interface design offers considerable benefits to these users. When user interface design complements the way users think and learn, accommodates their physical needs, and meets their expectations for comfort and convenience, the interaction of human and machine becomes more productive.

Referring to the knowledge base and the BPM solution, the information system UX can represent a connection point between the business process, modelled using the BPM solution, and the related support concepts provided by the knowledge base.

Such a connection point allows the user to manage business processes and information system content in an integrated and centralized manner. Moreover, the reference regulations for a company are continuously subject to changes that unavoidably produce effects on the corresponding business processes. So, this close relationship between the company reference regulations and business processes also makes it necessary to constantly update the UX software system.

The solution proposed in this paper is called the Web Application Editor (*WebAE*) and aims to solve the following issues:

- Modelling the UX schema of a business information system that is integrated and contextualized for the business domain, with respect to the processes, organization structure and reference regulations.
- Easily adapting the UX schema with respect to the induced reflexes produced by changes in the business processes, organization structure and reference regulations.
- Automatically generating the Rich Internet Application (RIA) prototype that implements the UX schema previously defined.

The *WebAE* system is developed within the HSEPGEST (Management of Health, Safety, and Environmental Protection in enterprise processes) research project, and consists of an integrated

framework for UX engineering, along with the automatic generation of the corresponding RIA prototype. For the purposes of the HSEPGEST project, the *WebAE* system represents the UX modelling solution with particular reference to the Health, Safety, and Environmental Protection (HSEP) scope.

The following list the *WebAE* framework requirements:

- It is based on the project phases provided by the Interactive Dialogue Model [14] (IDM) and Rich-IDM [15] methodologies.
- It uses the Model-Driven approach [16] for the UX model definition of the software system.
- It extends the framework already designed and implemented in [17,18].
- It interacts with the workflow engine solution, in order to retrieve data concerning the business process design and execution. Indeed, the interactions with the BPM solution occur during the UX schema definition, but also when the RIA prototype is running.
- It interacts with the business knowledge base, in order to retrieve the domain concepts related to the organizational structure, processes and reference standards. The interactions with the knowledge base occur during the UX schema definition, but also when the RIA prototype is running [19,20].
- It is able to automatically generate the RIA prototype application, starting from the corresponding UX model. For this purpose, the *WebAE* framework applies a special Model-to-Text transformation set to the UX model.

The present paper is composed as follows: Section 2 introduces the related works using knowledge bases during UX design and implementation. Section 3 introduces the activity sequence related to the UX engineering process. Section 4 shows both the design and the architectural aspects related to the *WebAE* framework. This section is intended to introduce the macromodules that make up the *WebAE* framework: *IDM Editor* and *RIA Prototype Generator*. Section 5 shows both the design and the architectural aspects related to each of the components provided by the *WebAE* framework. Section 6 represents an evaluation study that aims to compare the needed effort in developing an application using a traditional approach and the framework we propose. Finally, Section 7 includes some conclusions and future work.

2 Related Works

The *WebAE* framework is developed within the HSEPGEST project, which considers the problems related to the HSEP themes within the company workplace, in a process-oriented manner. The *WebAE* framework can also be used to manage the content to include in the information system UX that is not limited to the HSEP context.

The studies we have conducted on the related works have also involved the main modelling methodologies of RIAs, including [21-27]. However, they do not provide the possibility of referring to the domain knowledge base and the BPM solution during the modelling phases.

In [28], there is an interesting survey in which the emphasis is on the improvement that the use of knowledge bases can make for the purpose of the user interface development. The authors affirm that an ontology-enhanced user interface represents, 'a user interface whose visualization capabilities, interaction possibilities, or development process are enabled or (at least) improved by the employment of one or more ontologies'.

In [10], an ontology-based approach for the user interface development is preferred over a model-based one. The main reasons for this are related to the possibility to:

- reuse the ontologies and their fragments; and
- use a specific library, in order to implement some system functions.

As a result, user-interface generation and maintenance can be achieved at lower cost.

Companies use the domain knowledge bases to store general information about the organization and the business processes, including all the aspects related to regulations and laws in the domains of health, safety and environmental protection. Such a prerogative for the knowledge bases represents both semantic and information support for the methodological and technological *WebAE* framework, that constitutes one of the main outputs provided by the HSEPGEST project.

An alternative to the *WebAE* framework is represented by the Web Information System auto-construction Environment (WISE) project [29]. This platform makes it possible to model the user interface of a Web Information System (WIS) and to automatically generate the corresponding source code, through the interaction with the WIS ontology. The WISE architecture adopts Ontology Driven Software Development (ODSD) and consists of the following components:

- *WISE Builder*: represents a graphical editor used for defining domain and behaviour ontologies;
- *WISE Mapper*: implements the system transformation between ontologies and relation database management systems;
- *Code Generator*: generates the WIS source code based on the designed ontologies.

In particular, WISE aims to generate automatically, starting with formal domain models, in order to achieve a good level of extendibility and maintenance for the solution.

In [30] a candidate methodology is reported, along with a mix of conceptual models for addressing the design and development of business process-based Web applications supported by rich interfaces. In particular, this methodology aims to model a business process using the BPMN notation, transform it into a WebML [31] specification of data and navigation models, and apply the RUX-Method [21] presentation model for obtaining a RIA. So, the approach presented in [30] is able to provide for integration mechanisms with the BPM solution during the RIA design, but starting from different assumptions with respect to the methodology we propose. Such an approach aims primarily to create a BPMN model of a business process and then use it in order to implement the UX artefacts required to perform the same process.

The solution we propose changes the perspective of the design with respect to the approaches mentioned above. The design phase starts from the UX schema definition, also referring to the knowledge base and the BPM solution, in order to proceed with the automatic generation of the

correspondent prototype. For the purpose of the project, the reference BPM workflow engine is represented by jBPM (*www.jbpm.org*).

3 Web Application Editor: RIA Prototype UX Engineering Process

The prototype UX engineering process defines the sequence of activities the designer performs to model and generate the prototype UX, using the functionality provided by the *WebAE*. Such an engineering process provides the activity sequence already introduced in [18], with the exception of the BPM solution integration. In the following such an activity sequence is reported as a list of steps, trying to highlight the integration points provided by the *WebAE* with the knowledge base and the BPM solution, respectively.

Step 1. The designer uses the C-IDM editor to model the corresponding diagram, in order to represent all the concepts, along with the corresponding relationships, related to the specific business domain. Considering the low level of detail that characterizes the C-IDM diagram, at this stage there are no integration points either with the knowledge base or with the BPM solution.

Step 2. The L-IDM editor uses the previously obtained C-IDM model as input for the automatic generation of the corresponding diagram. At the build completion, the designer may decide to further detail the previously defined macro-arguments, also referring to the concepts present in the knowledge base. In this way, the designer defines the first integration points with the knowledge base, within the prototype UX schema. The designer also uses the L-IDM editor to perform the L-IDM-Rich-IDM mapping, which represents the configuration phase of the Rich-IDM diagram generation. In particular, the designer performs the mapping in order to define how many and which Context Views to create within the Rich-IDM diagram, but especially how to distribute the topics provided by the L-IDM diagram among these. At the completion of the L-IDM-Rich-IDM mapping, the designer may require the L-IDM editor to generate the Rich-IDM diagram. At the same time, the designer specifies the coordinates to access the business-processes repository, along with the home directory of the application server of the BPM solution.

Step 3. Using the parameters previously specified by the designer, the Rich-IDM editor performs the automatic generation of the corresponding diagram. During this process, the editor also retrieves the following information:

1. The business process references list retrieved directly from the repository workflow engine;
2. The lists of roles and users authorized to access the web console of the BPM solution.

In this way, the editor acquires everything that is necessary for the designer to perform the following operations:

1. The creation of the access points to the BPM solution, within the prototype UX;
2. The definition of an access management policy to the contents provided by the Context Views and the User Experience Cores of the UX schema.

Step 4. At the completion of the automatic generation, the editor displays the Rich-IDM diagram to the designer for the necessary changes. Similar to the L-IDM diagram, the designer can change the

structure of the diagram, in order to better detail the content to present to the user. In particular, within the UX model the designer can also create some integration points with the knowledge base, and some access points to the BPM solution. Such access points are implemented by the designer through the definition of an association between a reference to a Business Process and a Context View, or a User Experience Core of the UX schema. The Design Time phase ends when the designer completes the editing of the Rich-IDM model.

Step 5. At the completion of the Rich-IDM diagram editing, the designer can require the *WebAE* to generate the corresponding RIA prototype. Indeed, such an operation is delegated to a *WebAE* module called a *RIA Prototype Generator* (*RIA-PG)*, using the Rich-IDM model as input. The prototype generation is performed during the Prototyping Time phase, which consists of the following 4 substeps: *Infrastructure*, *View Component Artefacts*, *Model Component Artefacts* and *Integration*. Below is a description about the purposes provided by each of these subphases.

Infrastructure: represents the operations set necessary for the RIA prototype infrastructure generation. Indeed, such an infrastructure is not obtained from scratch, but the generator refers to an application template that already also implements the logic required for the integration with both the knowledge base and the BPM solution. With regard to the knowledge base, the template has the ability to collect and integrate the concepts the designer has provided at Design Time into the prototype UX schema. With regard to the BPM solution, the template provides the business logic needed to interact with the web console of the workflow engine. This interaction is based on the definition of a specific work session with the web console. The prototype uses this work session both to start a new process instance and to retrieve the list of tasks related to the user, and/or the role logged on the prototype.

View Component Artefacts: represents the artefacts generation phase related to the prototype View component. The artefact generation is based on the processing of the Rich-IDM model elements. The View component artefacts are added directly to the prototype infrastructure that the generator has obtained previously. During such a process, the access points to the BPM solution are also created within the prototype UX, in the form of hyperlinks.

Model Component Artefacts: represents the Java artefacts generation phase related to the prototype Model component. These artefacts are implemented directly and compiled through the processing of the Rich-IDM model elements, called 'Slots'. Although not part of the Rich-IDM notation, these elements are used by the designer to better detail the contents provided by UX.

Integration: represents the recovery phase of the contents provided by the UX prototype that reside in the knowledge base. The result of the Integration phase represents the working RIA prototype, which may be deployed by the generator on the application server specified previously by the designer.

Step 6. On completion of the deployment, the designer can access the prototype in order to evaluate the performance of the UX model, in the form of a web application. Following this evaluation, the designer decides whether action is needed, and what kind of interventions to perform. If the designer only needs to intervene on the graphical aspects of the prototype, they can access the View component artefacts directly, using a WYSIWYG HTML editor. In contrast, if the designer

evaluates that the UX model performance does not meet their design ideas, they may decide to return, even several times, to the Design Time phase, to generate the prototype again. The RIA prototype evaluation represents a very important task for the designer, since it allows the examination of the UX schema from the user point of view. In particular, the RIA prototype evaluation is especially important with respect to both the integration points and the access points of the prototype. Indeed, the prototype is able to interact with both the knowledge base and the BPM solution. With regard to the knowledge base, the prototype implements a specific service that retrieves the concepts to present to the user periodically, in order to keep them synchronized with the same knowledge base. With regard to integration with the BPM solution, the prototype user triggers this directly, using the corresponding access points present in the UX prototype. Such an operation is then completed by the business logic already provided by the application template used to generate the prototype. This logic is able to start a new instance of the process selected by the user, but also to retrieve any control values from the prototype UX, to use as boot parameters for the process itself.

4 Web Application Editor: Design and Architecture

The Web Application Editor represents the Design Time component used by the process designer to model the UX schema and to generate the corresponding RIA prototype. Section 2 reported a brief description of two RIA development methodologies that provide an integration mechanism with a knowledge base or BPM solution, even if not at the same time [29,30]. In particular, with respect to the methodology introduced in [30], the solution we propose changes the perspective of the design.

During the design process of the UX RIA prototype, the designer can refer to the concepts of the knowledge base, but also to the business processes defined in the BPM solution.

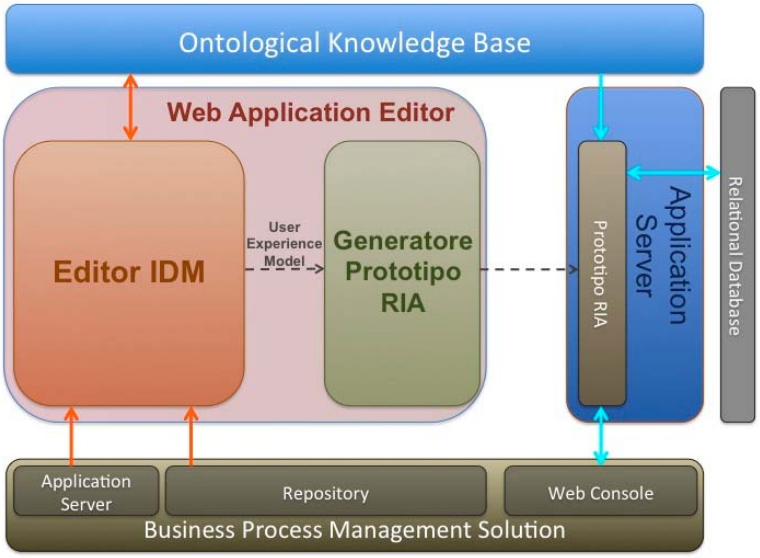

Figure 1 Web Application Editor: Architecture

The designer aims to generate a UX model, using RIA features that can help the prototype user to complete their business tasks, supporting them with useful context information.

The RIA prototype that implements such a UX model consists of the following components:

- Integration point: represents a point where the prototype user can consult all the information related to the processes they are involved. Such information is inherent to both the regulations and the laws, inside and outside an organization.
- Access point: represents a link that the prototype user employs to start a new instance on the BPM solution of the business process to which the access point refers.

4.1. Web Application Editor: Design

The *WebAE* allows the designer to define the prototype UX schema by also referring to the content provided by both the Process Repository and the knowledge base. In particular, the *WebAE* uses the *KB Middleware* module in order to define the integration points with the knowledge base. Figure 1 shows the *WebAE* architecture, which consists of the following software modules:

- *IDM Editor*: represents the module used by the designer to model the prototype UX schema, according to the design phases provided by the IDM methodology. During the UX modelling process, the designer can refer to both the concepts present on the knowledge base and the business processes defined in the repository of the BPM solution.
- *RIA Prototype Generator*: represents the module that is delegated to the RIA prototype generation, starting from the UX schema modelled by the designer using the *IDM Editor*. The generator also performs the deployment of the prototype on the Application Server instance specified by the designer at the prototyping process start-up.

As an explicit requirement, the *WebAE* architecture represents an extended version of the framework already introduced, analysed and implemented in [18].

Bearing in mind the structure and the functionality provided by the *WebAE*, it is possible to identify two macrophases, called Design and Prototyping Time, related to the use of the corresponding component by the designer.

During the Design Time, the designer also uses the *IDM Editor* to engineer the prototype UX schema, even referring to the content retrieved from the knowledge base. Furthermore, the content of the UX schema can also be associated with references to the business process present in the repository of the BPM solution. It is also possible to identify a further phase, denoted as Run Time. Such a phase begins when the prototype is working fully, and the designer can access it. The Run Time represents a particularly relevant phase for the designer, because they can evaluate the validity of the UX model they designed, in terms of RIA application. Evaluating the validity of the prototype UX, the designer is able to decide whether to resume, in whole or in part, the UX modelling in order to change both the graphical and the informational aspects.

With regard to the interaction with the other infrastructure modules, the *WebAE* implements direct and indirect interfaces.

Direct interfaces are concentrated mainly at the Design Time of the process implemented by the *WebAE*. Indeed, the designer uses these interfaces to create direct references to the concepts provided by the support systems, within the prototype UX schema. The references the designer has defined previously become information-integration points between the prototype and the corresponding support systems in the Run Time phase. So, these integration points arise from the indirect interfaces the *WebAE* implements with the infrastructure support systems.

The *WebAE* framework provides direct interfaces with the following target infrastructure components:

- Ontological Knowledge Base: this direct interface allows the designer to access the knowledge base concepts in order to properly characterize the content provided by the prototype UX.
- BPM Solution: the *WebAE* provides for direct interactions with the BPM solution, involving both the application server and the business process repository. In this case, the goal is to make available within the *WebAE* all the business-processes references present in the BPM solution, along with the lists of users and roles authorized to access the corresponding web console. So, the designer uses, respectively:
    - the business process references to associate them with particular UX schema elements;
    - the lists of users and roles to manage the prototype contents access.

The *WebAE* framework provides for indirect interfaces with the following components:

- Ontological Knowledge Base: allows the prototype to retrieve all the concepts from the knowledge base, whose references are provided at Design Time by the designer within the UX schema.
- BPM Solution: the prototype uses this interface to interact with the BPM web console in order to:
    - Allow the prototype user to start a new instance of the business process, whose references are provided at Design Time by the designer within the UX schema;
    - Retrieve the list of the tasks assigned to the prototype user;
- Relational Database: the prototype also provides an interface to an R-DBMS solution because, while the designer models the UX schema they can also refer to the contents residing in a relational database. In this case, the designer adds a Slot to an element of the UX model that references the database table and field that contain the value to retrieve.

4.2. Web Application Editor: Architecture

Figure 2 shows the detailed architecture of the extended *WebAE* component, and also demonstrates the *WebAE* components that operate during the Design, Prototyping and Run -Time phases, along with the correspondent design details. The Design and Run Time phases are characterized by a strong interaction with both the knowledge base and the business processes, along the entire UX engineering process.

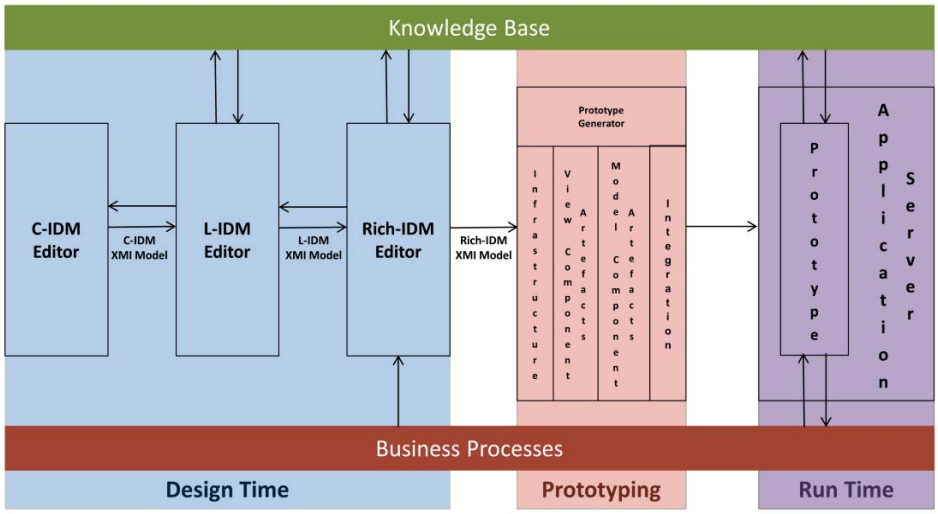

Figure 2 Web Application Editor: UX Engineering Process

At Design Time the *WebAE* architecture provides three visual environments, able to support the corresponding IDM methodology design steps. According to the Model-Driven approach, the implementation of such an environment requires the definition of the corresponding meta-models for each editor. These meta-models aim to define the set of elements, along with their attributes, that the editor makes available to the designer during the UX engineering phases. In particular, each editor produces a model in XMI (XML Metadata Interchange) format as output, which represents the input used by the component that follows in the UX modelling process. The model a component receives in input is used to automatically generate a first version of the corresponding diagram. Such a diagram is then presented to the designer for the appropriate changes. The final result of the Design Time phase is the Rich-IDM model of the prototype UX.

The Rich-IDM model also represents the main input used in the Prototyping phase in order to start the RIA prototype generation. The prototype generation ends with the deployment of the same prototype on the application server. At the same time, the prototype deployment starts the Run Time phase provided by the UX modelling process.

5 Web Application Editor: Components

This section reports and describes all the components that compose the web application editor.

5.1. Knowledge Explorer

In both the web application design and the UX prototype generation phases, the *WebAE* framework queries a domain knowledge base in order to retrieve and collect the concepts referenced by the designer. As already provided in the previous version, the software module the *WebAE* framework used to handle the interaction with the knowledge base is the *Knowledge Explorer*. Such a component is able to communicate with the knowledge base using the *KB Middleware* module. The purpose of such a module is to grant access to the knowledge-base information that is independent from the way

the same information is structured/stored. The communication implemented by the *Knowledge Explorer* is based on *RESTful* web services invocation and is able to support:

- Knowledge-base consultation – the UX expert has the possibility to enter free text, select one or more keywords and select the suggested concepts directly.
- Knowledge-base concept change management – in order to receive and notify the changes that occur to the knowledge-base concepts, *Knowledge Explorer* still uses the same states: *Active*, *Replaced* and *Deleted*. In particular, for the change notification it implements a mechanism of events publish/subscribe.

5.2. IDM Editor

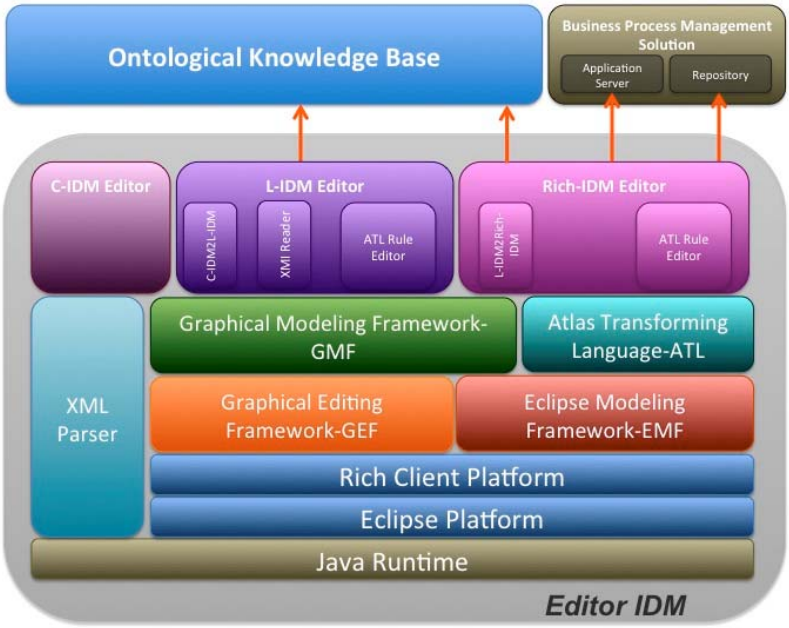

Figure 3. IDM Editor: Technological infrastructure

In the following, some technical aspects about the design and the architecture of the *IDM Editor* component are reported.

IDM Editor: Component Design

The architecture of the UX design framework, also called *IDM Editor*, is based on both the Model-Driven approach and the IDM methodology [14]. The choice to use IDM as the reference methodology for the framework is justified taking into account its main strength; that is, the particular focus on the interaction between user and system. With regard to the Model-Driven compliance, the *IDM Editor* provides an Eclipse-based architecture, with the GMF and the GEF frameworks. These frameworks adopt the Model-Driven approach for the assisted and semi-automatic generation of visual editors, based on the meta-models the developer defines.

The reference architecture for *IDM Editor* is shown in Figure 3. As can be seen, such architecture is essentially obtained by reusing, adapting and strengthening the *IDM Editor* component already introduced and analysed in [18]. The main innovation with respect to the previous version is represented by the additional interface with the BPM solution.

IDM Editor: Component Architecture

The reference requirements of the *IDM Editor* can also be used to identify the architectural specifications needed for the implementation of the editor itself. In addition to fully supporting the design phases provided by the corresponding methodology, the *IDM Editor* must also be able to manage the related diagrams and models. The *IDM Editor* architecture is essentially made up of an Eclipse workbench infrastructure, integrated by three visual development environments, in the form of Eclipse plug-ins. In the following, some technical aspects about such Eclipse plug-ins are reported.

C-IDM Editor: represents a visual development environment for the C-IDM diagram editing. The designer uses the C-IDM Editor in order to identify every macro-argument to present to the user, along with its corresponding relationships. The outputs generated by the editor are the C-IDM diagram and the corresponding XMI model.

L-IDM Editor: allows generation of the L-IDM diagram using the XMI model produced previously by the C-IDM Editor as input. The designer can still change such a diagram manually, by using the concepts residing in the knowledge base. Indeed, the L-IDM editor allows the designer to consult the knowledge-base content in order to retrieve the concepts to provide within the UX prototype. The designer also uses the L-IDM Editor in order to perform the L-IDM-Rich-IDM mapping. Such an operation allows the designer to define how many and which Context Views to create in the Rich-IDM diagram, especially how to distribute the Topics provided by the L-IDM diagram among these.

The outputs produced by the editor are the L-IDM diagram and the corresponding XMI model.

Rich-IDM Editor: the designer uses the editor to automatically generate the Rich-IDM diagram, starting from the L-IDM model. In order to proceed with the automatic generation, the Rich-IDM Editor provides a dialogue box, which the designer uses to enter the following configuration parameters:

- the URL relative to the web console of the BPM solution;
- the access credentials relative to the web console of the BPM solution;
- the file system path relative to the home directory of the reference application server for the BPM solution.

While the first two parameters are needed to retrieve the list of references related to the business processes defined in the workflow engine repository, the third is used to retrieve the lists of users and roles that are authorized to access the workflow engine web console.

After the generation of the Rich-IDM model, the designer can still change the diagram by associating a Context View or a User Experience Core with a reference to a business process, but also with a role and/or a user authorized to access the web console. Obviously, two such possible associations produce different effects during the RIA prototype generation. In particular, an association

between a Context View/User Experience Core and a web console role and/or user allows the designer to control the visibility of the corresponding information content, with respect to the selected role and/or user. In contrast, the association between a Context View/User Experience Core and a business process generates an access point to the web console of the BPM solution within the prototype UX. By using this access point, the prototype user can automatically start a new instance of the corresponding process, as well as access the web console in an integrated way, directly from the prototype user interface.

The designer can still use the Rich-IDM editor to consult the knowledge base and to retrieve all the relevant concepts to include in the RIA prototype UX. In this way, the designer can define specific information-integration points that may guide the prototype user to carry out their work activities. Such an editor prerogative is particularly productive, especially when the designer needs to refer to rules and regulations within the UX prototype. The information integration points represent the chance for the designer to characterize the prototype contents using general nature information, which can be complementary to the specific domain contents. So, the content retrieved from the knowledge base represents the real plus compared to the UX engineering of the prototype. If the designer is able to appropriately match an information integration point with a workflow engine access point, the UX prototype may condense the aspects related to both the technical know-how and the reference regulations for the business process into a single context.

The outputs produced by the editor are the Rich-IDM diagram and the corresponding XMI model.

5.3. RIA Prototype Generator

In the following some technical aspects about the design and the architecture of the *RIA Prototype Generator* component are reported.

RIA Prototype Generator: Component Design

The *RIA-PG* represents the *WebAE* component that is responsible for the RIA prototype generation, starting from the UX schema obtained by the designer using the *IDM Editor*. Such a component essentially matches the one already introduced in [18], albeit with a few small differences. In order to accomplish its function, the *RIA-PG* needs the Rich-IDM model that the designer has produced previously using the *IDM Editor*.

Before proceeding with the generator design details, it is important to introduce the following requirements that the RIA prototype has to fulfil. It must:

- be a Java servlet-based application;
- be a Java Enterprise Edition-JEE-complaint application;
- adopt the Model-View-Controller (MVC) pattern;
- be a RIA application;
- be able to interface with the corporate knowledge base in order to retrieve all the concepts provided by the designer in the UX schema;

- be able to interface with the reference workflow engine in order to allow starting a new instance of a business process. In this context, the prototype must also identify the UX controls whose values have to be used as process start-up parameters;
- be able to retrieve the list of tasks that is in the direct charge of the prototype user from the web console of the workflow engine;
- provide a graphical interface that is able to integrate the web console of the BPM solution.

The *RIA-PG* must also implement a RIA prototype capable of providing the following services:

- identification of the UX content to retrieve from the knowledge base;
- regular update of the content values that reside in the knowledge base;
- recovery and regular updating of the list of tasks in the charge of the user and/or the profile logged on the prototype, directly from the jBPM web console;
- access management to the content provided by the prototype, according to the user account and/or the role relative to the user logged on the prototype itself;
- *REST* calls management related to the creation of a new jBPM process instance.

Considering that the prototype must be an MVC pattern complaint, the *RIA-PG* has to implement the View and Model component artefacts of the RIA application. The *RIA-PG* does not produce the prototype from scratch, but it still uses a template application that already implements the Controller component, some basic features (such as login, logout and session timeout management) and the business logic to interact with the web console of the BPM solution.

In contrast, the artefacts related to the Model and View components must be generated dynamically, since their implementation specifications must be retrieved directly from the UX schema. In order to implement the Model component of the prototype, the *RIA-PG* still uses Plain Old Java Object (POJO) classes that are essentially JavaBean specification-compliant. With regard to the View component, in order to implement the user interface artefacts the *RIA-PG* now uses FreeMarker templates (*freemarker.org*). Such a choice is justified mainly for the sake of uniformity and conformity with the corresponding reference technology used by jBPM.

On completion of the View and Model component artefacts creation, the generator also proceeds with the prototype deployment on the application server that the designer has to specify. Before starting the prototyping process, the designer specifies the reference to an application server installation directory of where to deploy the RIA prototype. For the purpose of the project, the Apache Tomcat (*tomcat.apache.org*) represents the web container reference solution. On completion of the deployment operation, the generator also starts the web container instance and opens a browser window on the prototype homepage.

At this point, the designer can log into the prototype using one of the accounts authorized to access the web console. In this way, the designer can control the validity of the UX prototype, in the form of web application, in order to decide whether and how to act on the prototype UX. Therefore, the designer can intervene on both the UX model structure and the graphical aspects of the prototype. In the latter case, the designer can use a WYSIWYG HTML editor in order to intervene directly on the

View component artefacts of the prototype. In contrast, if the designer decides to intervene on the UX schema, they have to return at Design Time and to change any particular model appropriately, using the *IDM Editor*.

RIA Prototype Generator: Component Architecture

The logical architecture provided by the *RIA-PG* is essentially obtained by reusing and reorganizing the *RIA Prototyping Generator* component already introduced, designed and implemented in [18].

The technological infrastructure of the *RIA-PG* is still based on EMF and Acceleo (*eclipse.org/acceleo*) frameworks, in order to generate the View and Model components of the RIA prototype, respectively. Moreover, the *RIA-PG* also uses a Java library called CodeModel (*codemodel.java.net*) for the generation of the Model component classes.

The Model and View artefacts are implemented by the *RIA-PG* component using the FreeMarker templates and the corresponding Java classes, respectively. The View component artefacts uses the Model Java classes in order to manage their state and behaviour.

The *RIA-PG* provides a logical structure that consists of the following modules:

- *Model Navigator*: allows Rich-IDM model reading and the corresponding element retrieval, directly from the UX schema of the prototype;
- *Transformation Engine*: allows obtaining the components of the output model, starting from the corresponding Rich-IDM elements retrieved previously by the Model Navigator. For this purpose, the Transformation Engine module uses specific transformations made available by a component called the Transformation Repository;
- *Model-to-Text Generator*: generates the prototype artefacts written in source code, starting from the output model elements. For this purpose, the Model-to-Text Generator uses all the templates made available by the same generator. Besides characterizing the artefacts' look and feel, these templates also specify to the generator how to produce the target code, based on the attribute values provided by each element of the output model.

The main innovation with respect to the previous version is represented by the possibility to configure the prototyping process using a specific XML file. In order to obtain such a result, the last version of the *RIA-PG* architecture is re-organized as follows:

- a core component that is responsible for the user interface generation activities, which are also coordinated using the prototyping instructions retrieved from the correspondent XML file;
- n components of UX generation elements, related to the widgets and controls implemented according to a specific RIA technology.

Such architecture allows the *RIA-PG* to support the generation of View artefacts using different RIA technologies within the same prototype.

6 Evaluation

In order to evaluate the benefits of using the *WebAE*-based approach with respect to a more traditional methodology without the use of *WebAE*, a case study has been conducted. In particular, such a case study is related to the implementation of a system for academic exam management and aims at comparing the two approaches in terms of effort during the UX design of the prototype.

The evaluation was carried out on two distinct groups of people. We chose university staff for the system development using the traditional approach, and university students for the development of the same system using the *WebAE* approach. It is worth pointing out that the two groups may have different levels of expertise. The technological gap of the students with respect to the university staff is bridged by using the *WebAE*. However, in order to make possible the evaluation and comparison of both the approaches, the two groups are in line with the competences covered in terms of modelling.

6.1. Struts-based Application Design and Development

Regarding the development of the exam management system using traditional approach, the reference is a Struts-based application already implemented in-house by our university staff.

Table 1 shows the five canonical phases that distinguish a software development project, along with both the corresponding working hours and the effort in percentage terms.

A working hour represents an hour of productive work. So, in order to calculate the working hours amount it was excluded the number of hours devoted to holidays, illness and other activities in the project (training, meetings, etc.).

Table 1. Struts-based application development: effort and working hours

Project Phases	Effort %	Elapsed Time Working Hours – Wh (40/Week)
Analysis	12 %	240 WH
Design	8 %	160 WH
Development and Unit Test	63 %	1.280 WH
System Test	15 %	320 WH
Deployment	2 %	40 WH
Total on the phases	*100 %*	*2.040 WH*

The project had a duration of 11 months, which were required for the completion of all stages of the project.

Table 1 shows that most of the effort in percentage terms necessary to implement the examinations management system is related to the Development and Unit Test phase. The main reason for this result is represented by the need to write the source code that implements the system business logic and user interface, along with the system integration with knowledge bases and databases.

6.2. WebAE Framework-based Application Design and Development

For the evaluation of the *WebAE*-based approach, a student group attending the *Web Information Systems* course was arranged. This student group has a good knowledge of both the application domain and the IDM notation.

In order to facilitate the performance comparison, the student group had the possibility to reuse the project documentation already produced by the university staff. In particular, this project documentation consists of:

- The diagram, in BPMN standard notation, of the student exam verbalization process;
- The domain data model.

In this way, it was possible to ensure the comparability between the application developed by the academic staff and that implemented by the students, with the same domain and functional requirements.

Regarding the application developed using a traditional approach, the developer fulfils all the activities provided by the business process. In contrast, regarding the development of the application using the framework *WebAE*, the developer operates at both design time and run time. In particular, at design time the developer manages the integration of the Rich-IDM model of the RIA prototype, with the business processes defined in the workflow engine. Instead, the operations the developer has to perform directly at run time on the RIA prototype aim to:

- Manage the integration with the database in order to implement the operations of insert, update and delete. To this end, it was possible to reuse of all the business logic methods already implemented by the academic staff.
- Intervene on the user interface of the RIA prototype in order to make available to the user the operations of insert, update and delete.
- Intervene on the visual aspects of the RIA prototype application.

The learning curve related to the use of the *WebAE* framework has an average of two weeks, in comparison with the learning curve of the Struts framework.

The activities carried out during the trial in the classroom are as follows:

- Modelling of the UX IDM schema of the RIA prototype, using the *IDM Editor* provided by the *WebAE* framework;
- Validation and adaptation of the IDM models designed by students compared to that produced and used by the academic staff;
- Creating the RIA application, using the prototype generator provided by the *WebAE* framework.

Table 2 shows the data related to both the effort and the working hours of the canonical project phases, for the application development using the *WebAE* framework. Taking into account what has already been said, the phases on which it is necessary to focus attention are the Design and Development and Unit Test. In this case, the classroom trial had a duration of 3 months, required for the completion of all stages of the project.

Table 2. WebAE framework-based application development: effort and working hours

Project Phases	Effort %	Elapsed Time Working Hours – Wh (40/Week)
Analysis	43 %	240 WH
Design	22 %	120 WH
Development and Unit Test	14 %	80 WH
System Test	14 %	80 WH
Deployment	7 %	40 WH
Total on the phases	*100 %*	*560 WH*

Observing the results shown in Table 2 it is possible to note that the effort percentage related to the Development and Unit Test phase is significantly reduced with respect to the other phases. The *WebAE* approach focuses on the design phases of the system, while also trying to automate and reduce the development time and effort as much as possible. Such a prerogative of the *WebAE*-based methodology has actually helped to reduce the amount of the necessary Working Hours, in order to implement the system by using the two approaches, from 2,040 to 560.

From the comparison between the data introduced previously (Figure 4), it is possible to note a significant reduction in terms of effort and working hours corresponding to the phases of design and development.

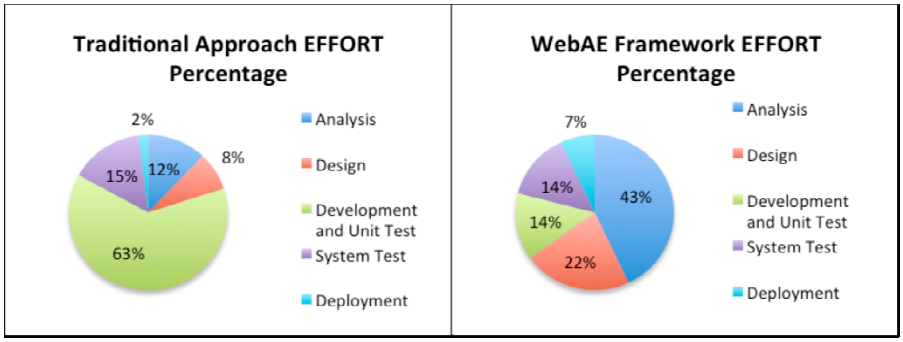

Figure 4. Struts-based and WebAE framework-based application development: effort percentage comparison

7 Conclusions and Future Work

The UX represents a critical factor for the success and the innovation level of an information system. As a consequence, it is important to model the UX effectively and efficiently, in order to develop a new application, or to re-engineer an existing one. It also important not to avoid considering the specific domain context, during the UX design. Such a context is characterized by different information (like company organization, business processes, regulations and laws, etc.) that can be managed using both knowledge bases and BPM systems.

This paper describes a methodological and technological framework for the engineering of the UX, along with the automatic generation of the corresponding RIA prototype. In particular, the paper provides the architectural aspects of the macromodules that constitute the *WebAE* framework; namely, the *IDM Editor* and *RIA Prototype Generator*.

Unlike solutions found in the literature, the proposed framework interacts with a BPM system and a domain knowledge base. In addition, the formal representation of the company application domain, in terms of concepts and relationships between them, can support the design experts in creating appropriate UX models that are flexible with regard to the changes in the business context.

Planned future work includes the extension of the framework in order to support the generation of prototypes that can run on different platforms. For this purpose, the development of a prototype that can be deployed on an Enterprise Content Management System is particularly interesting.

References

1. Kraft, C. User Experience Innovation: User Centered Design that Works. Apress, 2012.
2. Nagamachi, M. Kansei Engineering: A New Ergonomic Consumer-Oriented Technology for Product Development. International Journal of industrial ergonomics, 15 (1). 3-11.
3. Schutte, S. (2005). Engineering Emotional Values in Product Design. Kansei Engineering in Development, Unpublished doctoral dissertation, Department of Mechanical Engineering, Linköpings University, Sweden.
4. Levy, P. Beyond Kansei Engineering: The Emancipation of Kansei Design. International Journal of Design, 7(2). 83-94.
5. Abowd, G. Beale, R., Dix, A. and Finlay, J., Human-computer interaction. Prentice Hall, 1996.
6. Fleming, J. and Koman, R. Web navigation: designing the user experience. O'Reilly, 1998.
7. Peschl, M. F. and Stary, C. The role of cognitive modeling for user interface design representations: An epistemological analysis of knowledge engineering in the context of human-computer interaction. Minds and Machines, 8(2). 203-236.
8. Sadiq, S. and Governatori, G. Managing regulatory compliance in business processes.Handbook on Business Process Management 2, Springer Berlin Heidelberg, 2015, 265-288.
9. Liu, B., Chen, H. and He, W., Deriving user interface from ontologies: a model-based approach. in Proceedings of 17th IEEE International Conference on Tools with Artificial Intelligence, (Hong Kong, 2005).
10. Kleshchev, A. and Gribova, V. From an Ontology-Oriented Approach Conception to User Interface Development. Information Theories & Applications, 10 (3). 87–94.
11. Buriano, L., Marchetti, M., Carmagnola, F., Cena, F., Gena, C. and Torre, I., The role of ontologies in context-aware recommender systems. in Proceedings of 6th IEEE International Conference on Mobile Data Management, (Sydney, 2006).
12. Razmerita, L., Angehrn, A. and Maedche, A., Ontology-based user modeling for knowledge management systems. in Proceedings of the 9th International Conference on User Modeling 2003, (Johnstown, 2003).
13. Van Greunen, D., Van Der Merwe, A. and Kotze, P. Factors influencing BPM tools: The influence on user experience and user interfaces. International Journal of Computing and ICT Research, 4(1). 47-57.
14. Bolchini, D. and Paolini, P. Interactive dialogue model: a design technique for multichannel applications. IEEE Transactions on Multimedia, 8(3). 529-541.
15. Pandurino, A., Bolchini, D., Mainetti, L. and Paiano, R., Rich-IDM: extending IDM to model rich internet applications. in Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services, (Paris, 2010).
16. Kent, S., Model driven engineering. in Proceedings of the 3rd International Conference on Integrated Formal Methods, (Turku, 2002).

17. Martella, A., Paiano, R. and Pandurino, A., A dialogue-based framework for the user experience reengineering of a legacy application. in Proceedings of 15th IEEE International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, (Las Vegas, 2014).
18. Caione, A., Martella, A., Paiano, R. and Pandurino, A., A Knowledge-Driven framework for user experience modelling and prototyping. in Proceedings of 6th IEEE International Conference on Internet Technologies and Applications, (Wrexham, 2015).
19. Paiano, R., Guido, A. L. and Pandurino, A. Designing Complex Web Information Systems: Integrating Evolutionary Process Engineering. IGI Global Publishing, 2009.
20. Del Nostro, P., Orciuoli, F., Paolozzi, S., Ritrovato, P. and Toti, D., A Semantic-Based Architecture for Managing Knowledge-Intensive Organizations: The ARISTOTELE Platform. in Proceedings of the 12th International Conference on Web Information Systems Engineering, (Sydney, 2011).
21. Preciado, J. C., Linaje, M., Comai, S. and Sanchez-Figueroa, F., Designing rich internet applications with web engineering methodologies. in Proceedings of the 9th IEEE International Symposium on Web Site Evolution, (Paris, 2007).
22. Bozzon, A., Comai, S., Fraternali, P. and Carughi, G. T., Capturing RIA concepts in a web modeling language. in Proceedings of 15th IEEE International Conference on World Wide Web, (Edinburgh, 2006).
23. Urbieta, M., Rossi, G., Ginzburg, J. and Schwabe, D., Designing the interface of rich internet applications. in Proceedings of 5th IEEE Latin American Web Congress, (Santiago, 2007).
24. Meliá, S., Gomez, J., Perez, S. and Diaz, O., A model-driven development for GWT-based Rich Internet Applications with OOH4RIA. in Proceedings of 8th IEEE International Conference on Web Engineering, (New York, 2008).
25. Machado, L., Filho, O. and Ribeiro, J. UWE-R: an extension to a web engineering methodology for rich internet applications. WSEAS Transactions on Information Science and Applications, 6(4). 601-610.
26. Martinez-Ruiz, F. J., Arteaga, J. M., Vanderdonckt, J., Gonzalez-Calleros, J. M. and Mendoza, R., A first draft of a model-driven method for designing graphical user interfaces of rich internet applications. in Proceedings of 4th IEEE Latin American Web Congress, (Cholula, 2006).
27. Valverde, F. and Pastor, O., Applying interaction patterns: Towards a model-driven approach for rich internet applications development. in Proceedings of 7th International Workshop on. Web-Oriented Software Technologies (New York, 2008).
28. Paulheim, H. and Probst, F. Ontology-enhanced user interfaces: A survey. Semantic-Enabled Advancements on the Web: Applications Across Industries: Applications Across Industries. Information Science Reference, 2012.
29. Tang, L. A., Li, H., Qiu, B., Li, M., Wang, J., Wang, L., ... and Tang, S., Wise: a prototype for ontology driven development of web information systems. in Proceedings of 8th Asia-Pacific Web Conference on Frontiers of WWW Research and Development, (Harbin, 2006).
30. Brambilla, M., Preciado, J. C., Linaje, M. and Sanchez-Figueroa, F., Business process-based conceptual design of rich internet applications. in Proceedings of 8th IEEE International Conference on Web Engineering, (New York, 2008)..
31. Ceri, S., Fraternali, P., and Bongio, A. Web Modeling Language (WebML): a modeling language for designing Web sites. Computer Networks, 33(1). 137-157.