Science of Computer Programming 78 (2013) 2282–2291

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Bridging the gap between information architecture analysis and software engineering in interactive web application development

Luis A. Rojas, José A. Macías *

Escuela Politécnica Superior, Universidad Autónoma de Madrid, Tomás y Valiente 11, 28049 Madrid, España, Spain

ARTICLE INFO

Article history:
Received 15 January 2012
Received in revised form 11 July 2012
Accepted 26 July 2012
Available online 20 August 2012

Keywords:
Human–computer interaction
Information architecture
End-user development
Software engineering

ABSTRACT

Web development teams comprise non-computer experts working on the conceptual modeling of non-functional aspects in software applications. Later on, such conceptual information is processed by analysts and software engineers to face the technical phases of the software project. However, this information transfer is often difficult to automate since the information processed by the different professionals involves different abstraction levels, as well as important cost and effort that need to be considered. The main aim of this research is to minimize these problems by increasing automation and interoperability in the development of interactive web applications. To take up this challenge, we have created and evaluated a tool that aims at bridging the gap between the conceptual definitions of web contents – i.e., the information architecture, and the UML elements for analysis and design required by software engineers, connecting functional and non-functional information to achieve the rest of technical activities during the software development process.

1. Introduction

Information architecture (IA) is a recent paradigm that has been gradually introduced in most web development projects today. IA is defined as the science of structuring, organizing and managing information, where the usability plays an important role in the solutions created [1]. IA is directly involved in the development of website, mobile devices, vending machine and electronic games interfaces, to cite a few. Its main objective is to facilitate the processing and assimilation of information, as well as the understanding of tasks performed by users in a defined information space [2]. The way people interact with digital information environments is directly influenced by the information architecture [3].

The information architect is the person in charge of the IA. S/he works in the early phases of interactive web application development, attempting to connect the conceptual knowledge supplied by users and the technical information (i.e., analysis and design) needed by software engineers responsible for implementing the final web application. However, it is very common that the roles of information architect and software engineer rarely match, as the information architect may have a non-technical profile more oriented to conceptual tasks or ergonomics. This makes it necessary to ensure some degree of interoperability and alignment between the output generated by the information architect and the input that the software engineer needs. If this flow of information can be done in an automatic way, time and effort can be drastically reduced in the software project, allowing each expert to concentrate on the task according to her/his knowledge, and so minimizing the time of the information transfer between both kinds of professionals [4].

The aim of this paper is to overcome this problem and bridge the gap between the tasks performed by the information architect and the ones achieved by the software engineer. To carry out this task, we propose a CASE (Computer Aided Software Engineering) tool called InterArch [5] (Interoperable Information Architecture), which allows experts in the

* Corresponding author. Tel.: +34 914976241; fax: +34 91 4972235.
E-mail addresses: luisalberto.rojas@estudiante.uam.es (L.A. Rojas), j.macias@uam.es (J.A. Macías).

0167-6423/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2012.07.020

problem domain to focus on content analysis and navigation while the tool automatically generates UML classes for software engineers, implicitly supplying elements in the solution domain.

Specifically, our research is based on the following objectives:

- Generate analysis and design information for analysts and software engineers from the conceptual representations of contents earlier created by the information architect.
- Build an easy-to-use CASE tool for the information architect that enables to automatically generate analysis and design information for analysts and software engineers from the conceptual descriptions created by the information architect, and so bridging the gap between the initial project activities, in the problem domain, and the technical development activities, in the solution domain.
- Evaluate the usability of this tool by means of a user experiment, in order to obtain initial feedback for improvement through an iterative and incremental end-user-centered development process.

This paper is structured as follows. Section 2 introduces the related work. Section 3 presents our approach in detail. Section 4 describes the transformation rules used in our approach. Section 5 provides a use case to show the functioning of our tool in detail. Section 6 reports on an evaluation with real users to measure the usability of our approach. Finally, Section 7 discusses conclusions and future work.

2. Related work

There are a great variety of tools for the creation of diagrams representing the information architecture [6], and also for analyzing and evaluating the information architecture in websites [7]. These tools correspond to desktop and online software applications commonly used by information professionals to draw *blueprints* and create *wireframes* and *content models*, such as Axure, Visio Professional, OmniGraffle, Denim, ConceptDrawPro, SmartDraw, Pencil Project, MockFlow, iPlotz, Pidoco, Lovely Chart, Mockingbird and Lumzy, to cite a few. These tools include libraries comprising graphical elements for web prototyping, which enables managing and publishing information elements as well as incorporating new graphical components. Commonly, some of these tools include annotations, footnote facilities, collaborative authoring and dynamic prototyping.

Other approaches address non-functional requirements representation like informal architecture documentation, UML diagrams, and Architecture Description Languages (ADLs). These tools provide abstraction from implementation details, as well as data structures and relationships between different components [8]. Although these approaches present some drawbacks in connecting architectural descriptions and implementations, this has been solved in [9] by presenting an ADL-based solution that supports the modeling of system architectures at different levels of abstraction, linking architectural concepts to different technologies, immediate conflict detection, and continuous synchronization of both architecture and implementation. However, none of the approaches included in [9] concrete explicit relationships between IA content models and UML class diagrams, or even mechanisms to represent non-functional information and transform it into UML descriptions to be exploited in the software project.

On the other hand, online tools are becoming more popular than desktop stand-alone versions, due to their availability and free-of-charge facility. However, online tools are generally less expressive and complete than desktop versions in terms of functionality. On the other hand, most common tools experience difficulties in connecting the output generated by the information architect and the input needed by the software engineer. This problem has been traditionally addressed by generating different graphical formats and HTML code to interchange information. However, this solution does not consider issues related to semantic analysis included in IA diagrams, and it lacks interoperability among different professionals working together, making it difficult to manage and share the knowledge generated by different tools and professionals in the software project.

3. Proposed solution

Generally, it is difficult to identify the operational limits of the information architecture, sometimes requiring the use of different tools and standards. However, it is possible to summarize the most common products that the information architect creates to carry out the analysis of the information architecture in interactive web application. Those are *blueprints*, *wireframes*, *content models* and *controlled vocabularies* [1,2]. These products provide important knowledge regarding the analysis, organization, managing and structuring of information for the professionals involved in the development of web projects. However, for all these products, content models are particularly transcendental for analysts and software engineers, as they represent non-functional aspect of the web application and are susceptible of being automatically processed to generate content classes and object diagrams that will define the software application in the solution domain. In fact, our approach is focused on these core elements for automating the output of the IA analysis.

This way, we have designed a CASE tool called InterArch that is based on two essential principles. First, since the information architect usually has a non-technical profile, more oriented to information design and organization, InterArch allows the information architect to concentrate on conceptual analysis tasks in the problem domain. Second, based on the initial analysis carried out by the information architect, InterArch automatically generates UML diagrams for analysts and software engineers, identifying elements that have a direct correspondence with class diagrams and content objects used by software professionals. In order for the information to be processed by any common CASE tool and provide continuity for

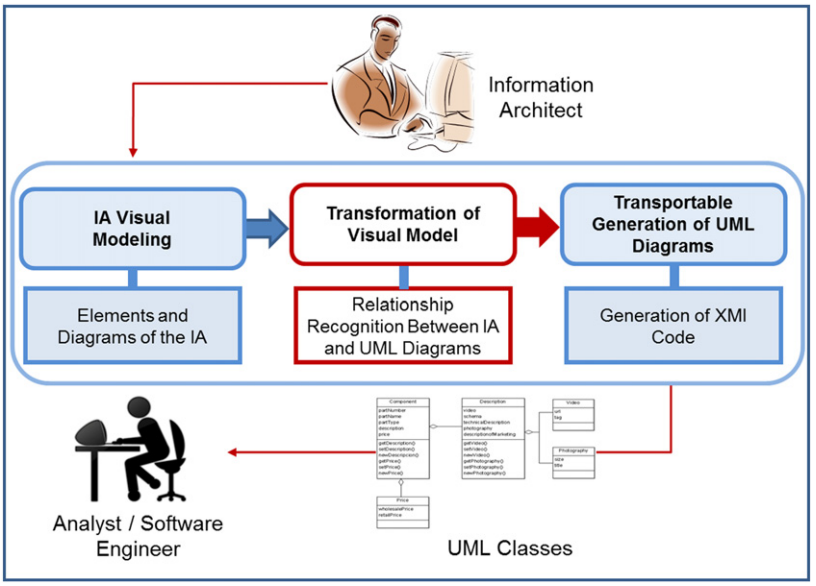

Fig. 1. Architectural details of the InterArch CASE tool.

other activities in the software development lifecycle, content information is generated in a textual and transportable XML format.

3.1. Architectural design of the proposed solution

InterArch comprises of a set of processes that are responsible for the management and transformation of models in a visual environment intended for the information architect. As shown in Fig. 1, these processes include: the visual modeling of the conceptual elements required by the information professionals, the transformation of the visual model into an intermediate model, and the generation of technical information in form of transportable UML diagrams. These processes are designed to take the input from the visual diagrams created by the information architect and generate UML diagrams for the analyst and software engineer as an output.

The main idea behind these architectural components is enabling the information architect to work on the visual modeling in a transparent way, but also incorporating a powerful interpretation layer that recognizes the different correlations between the IA diagrams and the UML classes required by software engineers. The transformation of visual model is based on a set of relationship and association rules that are applied to the conceptual model produced by the information architect, generating a set of UML diagrams in a transportable XML format called XMI.

The visual modeling of the conceptual IA elements is the first process shown in Fig. 1 (from left to right), which is carried out using the main user interface of InterArch. Such an interface is the main working environment for the information architect, and it is composed of different toolbars used to draw and manage diagrams on a functional environment. Fig. 2 shows this user interface, where the main sections are labeled with capital letters (A, B and C). This process allows the information architect to develop the different diagrams for the information architecture. The second process shown in Fig. 1 is the transformation of the visual model that includes identifying each of the visual elements produced by the information architect for composing, later on, the UML diagrams used by analysts and software engineers. This is accomplished through the association and relationship rules that are applied to the visual elements individually or in groups. This process allows the relationship recognition between information architecture diagrams and UML class elements (classes, operations, attributes, associations, etc.)

Finally, the last process shown in Fig. 1 is the transportable generation of UML diagrams. This process takes as input the relationships created in the aforementioned transformation step and generates UML class diagrams in XMI format for software engineers. XMI provides a de facto standard for serializing, editing and customizing UML diagrams by analysts and software engineers, in order to be incorporated and reused in a software project. The goal is to carry on the analysis and design phases in the project and combine these diagrams with the functional part of the interactive web application by means of other CASE tools used during the rest of the project's technical development phases.

3.2. General description of the InterArch CASE tool

In general, InterArch allows manipulating, formatting and linking visual content elements for the development of information architecture diagrams, which allows the information professional to create and manage visual models for IA by means of the direct manipulation of visual elements.

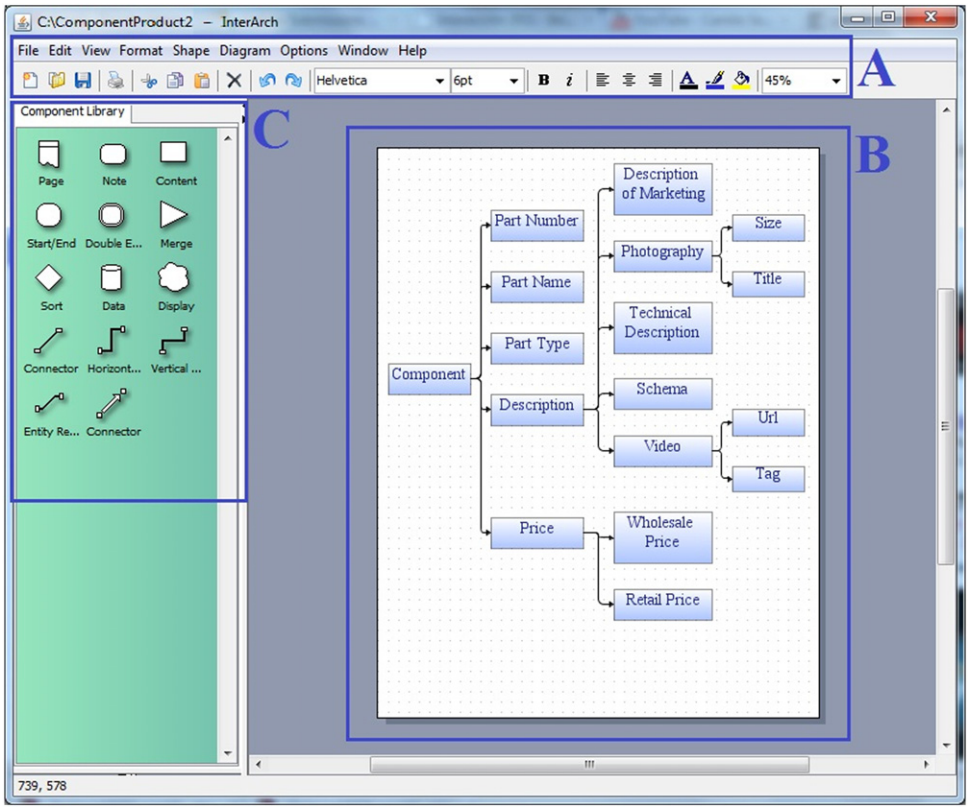

Fig. 2. User interface of InterArch divided into three sections: A, B and C.

Section C in Fig. 2 corresponds to the visual icons to draw diagrams, that is, graphical objects for composing different visual elements and enriching the creation of diagrams by the information architect. There are two main authoring elements for the visual modeling:

- Information elements for the visual conceptual modeling (first three rows of icons). These elements enable manipulating and interpreting the content entities for the visual-conceptual modeling of the IA. The information elements include different shapes and visual styles. The main idea behind these elements is to allow the information architect to define visual objects representing content entities that will be linked to others.
- Linking elements for creating associations and relationships between content items (last two rows of icons). Linking elements also have different shape and visual styles, but the purpose is similar – i.e., these elements enable the information architect to create relationships between different content elements and define a proper hierarchy from them.

This classification of the visual elements plays an important role in the activation of transformation rules to subsequently accomplish an adequate generation of UML class diagram.

Section B in Fig. 2 depicts the main working environment for manipulating and linking visual elements. In the example depicted in Fig. 2, relationships between content elements, which describe the structure and prices of each product in an online shop, are shown. The manipulation styles allow handling elements collectively in the diagram, creating visual element properties and linking content elements in a hierarchical way on the working environment.

Some of the manipulation styles supported by InterArch are the following:

- Connecting information elements. This is achieved by dragging the source element and dropping it onto the destination one. This automatically generates a link between both information elements.
- Grouping information elements. Elements can be grouped and manipulated as a block. This is achieved through the selection of various visual elements together.

Visual elements have inheritance properties that can be exploited in the workspace. This lets the user create new information elements inheriting features from the source visual element selected.

Section A in Fig. 2 shows formatting and editing options, which allow the manipulation of elements in the working environment. The most important functionality in this part is the one for saving diagrams to disk. Specifically, the option "UML diagram FILE (.xmi)" transforms the content model developed by the information architect into UML class diagrams

in XMI format. This option starts up the processing of the IA model and applies the corresponding transformation rules, as will be described bellow.

4. Transformation rules

InterArch includes an interpretation layer comprising a set of transformation rules that analyze association and hierarchy relationships in the content models, developed by the information architect, in order to be transformed into UML code. Transformation rules are divided into *Hierarchy* and *Configuration* rules:

4.1. Hierarchy rules

In order to have formal criteria to validate the automatic processing of content models, and the further transformation into UML class diagrams, a set of rules has been defined. Hierarchy Rules deal with structure and hierarchy of the content-model diagrams produced by the information architect. Such rules have been inspired by the process model appearing in [10], which proposes the transformation of non-functional content information, initially developed by analysts, into classes that will be included later in the functional application classes for the design of web applications, fostering continuity in the design of non-functional elements that takes place in early phases of the web development process. More specifically, our research has formalized, improved, implemented and evaluated this previous approach by means of a CASE tool, identifying the potential roles of information architect and software engineer, and building a set of rules that allow relating content models developed by the information architect and the UML classes required by analysts and software engineers. Additionally, our approach incorporates new features such as generation of methods and the configuration of relationships and rules to successfully generate UML class diagrams. These rules are applied to every element in the content model to perform corresponding UML transformations. That is, the rules consider the structure and hierarchy of content to create classes, attributes, operations and associations in the resulting UML class diagrams.

Specifically, InterArch includes five main hierarchy rules that can be defined as follows:

*R*1: A content element containing other descendant elements is directly considered as a UML class.

*R*2: A descendant content element is considered as an attribute, which is included in the class elements from which it descends.

*R*3: The main element of the content model diagram will be the main class in the UML class diagram.

*R*4: A descendant content element corresponding to a new class generates a direct association with the element from which it descends.

*R*5: For each of the associations generated in the UML class diagram, three methods (get, set and new) are created and included in the source class.

4.2. Configuration rules

Configuration rules are a set of specific properties concerning the level of visibility, access and navigability in classes, attributes, methods and associations of the UML class diagrams. Unlike hierarchy rules, configuration rules do not consider the structure of the content diagrams produced by the information architect. By contrast, configuration rules deal with properties that will affect the generation of UML class diagrams. Another difference with respect to hierarchy rules is that configuration rules are principally focused on software engineers due to the technical knowledge required for manipulating UML properties. This means that information architects can use InterArch to carry out content modeling in the problem space, while software engineers can use the configuration rule facility to configure the UML that better fits the design requirements in the solution space.

Fig. 3 shows the InterArch module for managing configuration rules. Also, the figure shows the rules selected by default, which are grouped into class, attribute, method, and association rules. These configuration rules corresponds to those proposed by the OMG [11] in order to customize the schemas and documents produced using XMI as an interoperable textual XML-based language for representing UML. The categories of configuration rules are the following:

- Class: This group provides options to configure the visibility (public, package, protected and private) and the access type (active, abstract, leaf and root) for classes. It is configured by default that class attributes have a public visibility and no defined access.
- Attribute: This group provides options to configure the visibility (public, package, protected and private) and the access type (owner scope) for class attributes. By default, attributes have a public visibility and no defined access.
- Association: This group involves the configuration of different features such as navigability between classes, aggregation (aggregation, composite and none) and access (root, leaf and abstract) for all UML relationships in the class diagram. By default, associations are standard and have no defined access or specific navigation between classes.
- Operation: This group provides configuration facilities for the visibility (public, package, protected and private) and access (query, root, leaf, owner scope and abstract) for class operations. By default, operations have public visibility and no defined access.

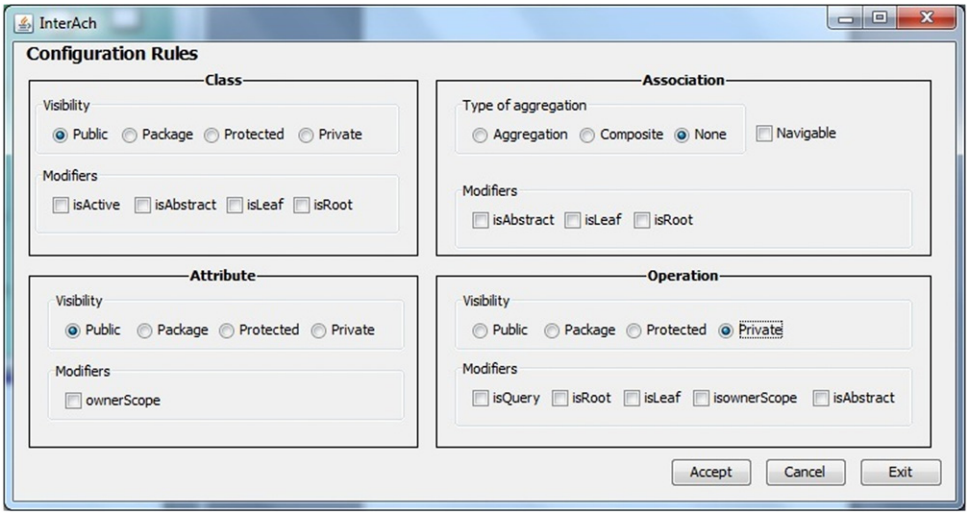

Fig. 3. Configuration rules module of InterArch.

5. Use case

In order to show in detail how our CASE tool works, we provide a specific use case.

5.1. Task description for the use case

Let us suppose that the information architect desires to work on a content model representing product information for an online shop. This information is depicted in Fig. 4, which represents a medium-fidelity prototype – i.e., mock-up obtained in the requirements elicitation phase with the sales manager. As shown, the mock-up contains information regarding name, type, product number, technical description and marketing, as well as an image and video describing the features of the product visually. The information architect would carry out the conceptual modeling achieving the following tasks:
(a) *Segmentation*: The first task is content segmentation using the mock-up provided in Fig. 4. This task identifies different components and structures of information, obtaining the composition and hierarchy of the different information elements contained in the mock-up. Results of the content segmentation allow the information architect to have the necessary information to prepare the content model by using InterArch.
(b) *Content Modeling*: This is carried out by using InterArch and considering the information previously obtained during the segmentation, which generates a hierarchical information diagram as depicted in Fig. 2 – Section B, where the content object *Component* is described by other five different content objects (*Part Number, Part Name, Part Type, Description* and *Price*), in which *Description* and *Price* are compound data that are defined according to the content objects that hierarchically descend from them. Once created, the content model can be automatically transformed into UML diagrams by activating the corresponding transformation rules. This step is completely transparent for the information architect, and it is run by simply clicking on the corresponding menu option.

5.2. Association and relationship rules to process visual elements

To generate the UML diagrams from the IA content model, the rules explained before are applied. This is achieved by saving the content model to disk using the option "UML diagram File (.xmi)". In this case, a main class is generated from the principal element (*Component*). Also, the descendant elements generate the following classes: final elements and direct descendants of the main element are transformed into attributes in class *component*. If these elements are both descendant elements then they are transformed into new classes directly related to the element *component*. If descendant elements generate attributes and they are compound elements then new classes are recursively generated and related to the element from which they descend. Applying this rule, a class *component* would be generated containing five attributes: *partNumber, partName, partType, description* and *price*. Also, applying the rules, element *price*, descending directly from the main element, and at the same time containing descendant elements *wholesale price* and *retail price*, is transformed into a new class, and its descendant elements *wholeSale price* and *retail price* in attributes for class *price*. Regarding class methods, three methods are generated by default for each attribute that represents an aggregate class (get, set and new).

(c) Generation of XMI code

The execution of the rules automatically generates a UML class diagram in a XMI textual format. The following code fragment represents the generated XMI code for the class *component*, containing some of the related attributes, methods and relationships with the class *price*, according to the previous example and the output diagram depicted in Fig. 5. This way,

Component	
Part Number	857789
Part Name	LED Screen
Part Type	TV
Description	
Schema	Model UN40D5800VGXZS
Description of Marketing	Full HD. Certified Product
Technical Description	Resolution On Line 1920 X 1080.
Video	
Url	\products\TV\FullHD\LS857789.mpeg
Tag	TV, Full HD, LED Screen.
Photography	
Size	411 x 334
Title	Full HD LED Screen
Price	
Wholesale Price	US 315
Retail Price	US 380

Fig. 4. Mock-up of a product for an online shop.

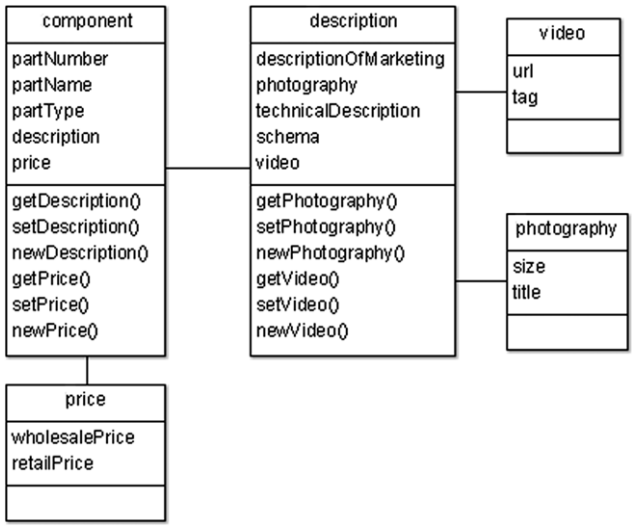

Fig. 5. UML class diagram obtained from the transformation of the IA content model.

the class *component* is represented by the tag *<UML:Class>*. Also, the attribute *partNumber* in class *component* is represented by the tag *<UML:Attribute>*. Regarding class methods, the method *getDescription* in class *component* is represented by the tag *<UML:Operation>*. All these tags contain a unique identifier and the name of the tag.

As for the associations between classes in the XMI class diagram, they are represented by *<UML:Association>*, along with *<UML:AssociationEnd.participant>* and *<UML:AssociationEnd>* tags that allow specifying the association type (aggregation, composite and none) and the classes involved in the association. Finally, all the tags defining classes, attributes, methods and associations have specific properties that can be customized, as described before, by the configuration rule module of InterArch.

<**UML:Class** xmi.id = '**x232**' name = '**component**'> <UML:Classifier.feature>

<**UML:Attribute** xmi.id = 'x232:87B' name = ' **partNumber**' visibility = 'public'></UML:Attribute>...

<**UML:Operation** xmi.id = 'x232:02C' name = **'getDescription'** visibility = 'public'></UML:Operation>...

<**UML:Class** xmi.id = '**x235**' name = '**price**'>...

<**UML:Association** xmi.id='868'> <UML:Association.connection> <**UML:AssociationEnd** xmi.id='889' aggregation= **'aggregate'**>...

<UML:AssociationEnd.participant><UML:Class xmi.idref='**x232**'/></UML:AssociationEnd.participant></UML:Association End><UML:AssociationEnd xmi.id = '874' aggregation=**'none'**><UML:AssociationEnd.participant><UML:Class xmi.idref= '**x235**'>...

The file generated in XMI format is portable and can be used in any UML diagramming tool supporting XMI – to cite a few: ArgoUML, StarUML, BOUML, VisualParadigm, Circa and Mia-Generation, among others.

(d) UML class diagram

Fig. 5 depicts the final UML class diagram automatically generated by InterArch. The class diagram consists of a root class *component* that contains five attributes: *partNumber, partName, partType, description* and *price*. Methods created for this class, denoting relationships with *description* and *price* classes, are getDescription, setDescription, newDescription, getPrice, setPrice and newPrice. In turn, class *description* is directly related to *video* and *photography* classes.

Additionally, it could be necessary for a specific design solution to concrete the cardinality in relationships or composition relationships (strong aggregation). This is the case for classes *price* and *description*, both related to class *component*, since it does not make sense that the price exists without the component (i.e., does not make any sense that the part exists without the whole). The type and level of dependence between relationships, as well as the cardinality and navigability, can be modified by the analyst or software engineer by means of the configuration rules explained before.

In general, the information created by InterArch can be adapted to more specific requirements, importing the generated XMI files in other CASE tools. This helps take advantage of the features provided by other tools for dealing with UML code, such as reverse engineering, database integration and OCL constraints, among others.

In short, the implementation of the methodology focused on the conceptual modeling in the problem domain, and the underlying transformation into a model closer to the solution domain, helped us reach the first and second objective specified at the beginning of the paper. However, it is also necessary to evaluate the tool to have an early idea of its usability. In the next section, an experiment with real users is carried out to determine the degree of the user's satisfaction concerning the InterArch CASE tool.

6. User experiment

In order to have some clues about the usability of InterArch, we have carried out an early evaluation with real users.

6.1. Participants and resources

To evaluate the tool, 12 users were enrolled. They regularly work for IT companies as project consultants specifically related to IA. They were 9 men and 3 women, aged between 24 and 43 ($M = 32$, $SD = 8.062$). In general, these users had previous knowledge about analysis, documentation, structuring and categorization of website contents. All users had experience in the use of similar tools oriented to content modeling, but they have never used InterArch before.

A retrospective analysis was used to obtain video recordings of the user experiment to further analyze the information later on [12]. On the other hand, the thinking aloud protocol was also used at the same time to observe the user while s/he interacts with the tool, so obtaining the main behavior observed. This protocol consisted in asking end-users to think aloud while they interact with the system in order to understand how they see the tool, which makes it easier to identify misconceptions and errors [12]. Both protocols helped measure and analyze user interaction with the tool, showing different issues and facilitating the measure of time and the analysis of events occurred during the interaction recorded in video sessions to be analyzed later on in detail.

6.2. Experimental task description

The evaluation consisted in a controlled experiment comprising the following steps:

(a) First, the different functionalities and features of the tool InterArch were shown to users. This tutorial took an average of 7 min ($SD = 142$ s).

(b) Next, users were asked to develop a content model about products for a second-hand online shop. Specifically, users were given a medium-fidelity mock-up extracted from one of the products included in the printed version of a second-hand magazine, and they were requested to create the content model using InterArch. This process, including the aforementioned segmentation stage, took in average less than 12 min ($SD = 194$ s). User interaction was recorded by sessions in order to be further analyzed, using the protocols described before.

(c) Finally, users were asked to complete a questionnaire to measure the usability perceived.

6.3. Questionnaire description

In order to measure the usability, and have a feedback about the user's satisfaction, we utilized the USE questionnaire [13], with some variations provided by the questionnaire of utility and perceived ease of use by Davis [14] and the Purdue Usability Questionnaire [15]. The questionnaire contained 31 closed questions divided into four groups to measure variables related to utility (8), ease of use (10), ease of learning (6) and satisfaction (7) concerning InterArch. These four variables correspond to the four dimensions for the estimation of the perceived usability. A numerical scale ranging from 1 (lowest) to 10 (maximum) was used to measure the answer to each question. Besides, we added four open questions in the questionnaire to enable users to include any other issue, such as positive and negative aspects concerning the tool.

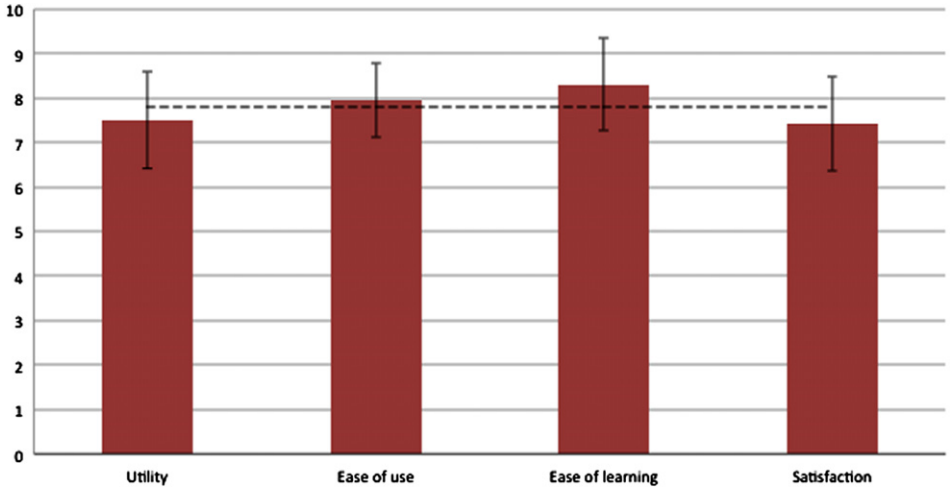

Fig. 6. Average score for each variable measured from 1 to 10, including error bars ($\pm\sigma$) and a dotted line representing the mean.

We used Cronbach's alpha to measure the internal consistency of the questionnaire. This indicator was calculated for the 31 closed questions in the questionnaire. The result shown a reliability value of 95.18% ($\alpha = 0.9518$), which indicates that the questionnaire had an excellent internal consistency level as it exceeds by 25.18% the threshold of acceptance.

6.4. Analysis and results

Fig. 6 shows the results obtained from the analysis of the questionnaire, indicating the average score obtained for each variable and an error bar corresponding to the standard deviation ($\pm\sigma$); the overall mean (horizontal dotted line) is also represented. As shown, the variable *ease of learning* obtained the highest average score ($M = 8.3$, $SD = 1.23$). The variable *ease of use* obtained the second highest average, with a score of 7.95 ($SD = 1.083$). It is followed by the variable *utility*, which obtained an average score of 7.5 ($SD = 1.089$). Finally, the variable *satisfaction* obtained the lowest average, with a score of 7.41 ($SD = 1.059$).

The overall mean for the four variables (dotted line in Fig. 6) was 7.79 ($SD = 1.14$). In general, all variables obtained scores over 74%, which can be considered as a good usability indicator of the tool InterArch according to the user's perception.

In addition, we also analyzed the sessions recorded by the aforementioned protocols, which reported valuable information about how users utilized the tool. For example, although the majority of users chose to use the functionality to select and drag on the content items (inheritance), it was found that this feature implied some difficulties, as users needed several attempts before successfully getting what they wanted to do. Also, we realized that users achieved several steps to find some of the tool's functionalities. This highlighted the necessity of having more shortcuts. All in all, no important errors were found during the experimental session. Furthermore, the results obtained will be used to improve the tool in the future so that it responds more efficiently and in accordance to the user's needs.

Also, the four open questions included in the questionnaire reported valuable information about strengths and further improvements. On the one hand, users highlighted the following positive aspects of InterArch: simplicity, ease of use, speed, intuition and similarity with other environments. On the other hand, users observed areas of improvements in the toolbar used to represent content elements, where more diversity was expected, also in element sizes and mouse grouping operations. All these issues will be taken into account in the future to improve InterArch.

6.5. Verification and validation

InterArch has been developed using an iterative and incremental prototype-oriented end-user-centered development process. Main software requirements have been elicited from a comparative analysis carried out with other similar tools, identifying drawbacks and areas of improvement, as well as considering the opinion of the different stakeholders related to IA. In addition, the theoretical formalism, comprising creation, interaction and transformation of visual models, has been fully analyzed, designed and implemented by means of a CASE tool called InterArch, This way, the research has been explicitly verified according to the early requirements and objectives stated. Furthermore, InterArch has been evaluated to measure its general usability according to the user's perception. This provides an implicit validation of both the research and the formalism conceived. On the other hand, information architects have been considered as the potential users of InterArch, this is why the usability experiment has been carried out with such users, obtaining acceptable feedback and results from the information professionals overall. However, InterArch can be used by software engineers to customize the UML code generated by the tool. In this sense, we have carried out some evaluations using experts in software engineering to validate the output generated by InterArch. This validation has been achieved by both manual inspection and using existing UML

tools, such as ArgoUML, StarUML, BOUML, VisualParadigm, Circa and Mia-Generation, that allow importing the UML code generated by InterArch without major problems, ensuring also the compatibility and the suitability of the UML generated by the tool, mainly intended for software engineers.

7. Conclusions and future work

In this paper, we have presented an approach consisting of a formalism to represent knowledge from conceptual definitions of the IA, and also a mechanism to transform this knowledge into analysis and design information to be processed by software engineers in order to develop interactive web applications. This approach has been validated through the construction and evaluation of a CASE tool called InterArch.

The main aim of InterArch is bridging the gap between high-level conceptual representations of the IA and non-functional representation of software, providing analysis and design classes that are necessary to implement interactive software in the solution domain. To carry out this task, the tool automatically generates UML class diagrams from content-model definitions of interactive websites, using XMI as intermediate language of representation that can also be processed by other different CASE tools. This increases interoperability in integrating functional and non-functional classes in the engineering process of interactive web application development.

Results obtained in the early usability evaluation of the tool shown positive and acceptable ratings about the user's perception on utility, ease of use, ease of learning and satisfaction. In addition, the sessions recorded through the retrospective analysis and thinking aloud protocols allowed us to obtain more detail regarding the user interaction with the tool. Also, the open questions included in the questionnaire reported valuable information to know the strengths and areas of improvement for InterArch.

As future work, in addition to improve InterArch with the early results obtained, a promising line would be incorporating semantic features in the tool – i.e., the inclusion of comments in the content elements by the information professional. This would provide further semantic information [16] for software engineers, so that more advanced constraints for the solution domain would be automatically generated. Also, another interesting line to consider is modeling explicit accessibility and usability properties in the early phases of the software project using InterArch.

Acknowledgments

This work has been supported by the founded projects TIN2011-24139 and TIN2011-15009-E.

References

[1] E. Erlin, Y. Yunus, A. Rahman, The evolution of information architecture, ITSim 2008, International Symposium on Information Technology 4 (2008) 1–6.
[2] P. Morville, L. Rosenfeld, Information architecture for the world wide web, O'Reilly Media, third ed., O'Reilly Media Inc., 2006.
[3] G. Elaine, Information interaction: providing a framework for information architecture, Journal of the American Society for Information Science and Technology 53 (10) (2002) 855–862.
[4] J. Macías, Intelligent assistance in authoring dynamically generated web interfaces, World Wide Web 11 (2) (2008) 253–286.
[5] L. Rojas, J. Macías, End-user support for information architecture analysis in interactive web applications, Interact 2011, in: LNCS, vol. 6949, Springer, NY, 2011, pp. 515–518.
[6] J. Garrett, A visual vocabulary for describing information architecture and interaction design, www.jjg.net/ia/visvocab, 2002.
[7] C. Katsanos, N. Tselios, N. Avouris, AutoCardSorter: designing the information architecture of a web site using latent semantic analysis, in: proceeding of the twenty-sixth annual SIGCHI conference on human factors in computing systems, CHI'08, ACM, New York, NY, USA, 2008, pp. 875–878.
[8] M. Babar, T. Dingsyr, P. Lago, H. Vliet, Software architecture knowledge management, Springer, 2009.
[9] G. Buchgeher, R. Weinreich, Connecting architecture and implementation, OTM'09, On the Move to Meaningful Internet Systems: OTM 2009 Workshops, in: Lecture Notes in Computer Science, vol. 5872/2009, 2009, pp. 316–326.
[10] R. Pressman, Software engineering: a practitioner's approach, McGraw-Hill, 2005.
[11] OMG, MOF 2.0/XMI mapping specification, http://www.omg.org/spec/XMI/2.1/PDF, 2005.
[12] J. Nielsen, Usability engineering, Morgan Kaufmann Publishers, 1993.
[13] A. Lund, Measuring usability with the USE questionnaire, Usability and User Experience Special Interest Group 8 (2001).
[14] F. Davis, Perceived usefulness, Perceived Ease of Use and User Acceptance of Information Technology, MIS Quarterly 13 (1989) 319–340.
[15] H. Lin, A proposed index of usability: a method for comparing the relative usability of different software systems, Behaviour and Information Technology 16 (1997) 267–278.
[16] E. Chavarriaga, J. Macías, A model-driven approach to building modern semantic web-based user interfaces, Advances in Engineering Software 40 (12) (2009) 1329–1334.