From Requirements to Web System Design. An Automated Approach using Graph Transformations

Sergio Segura, David Benavides, Antonio Ruiz-Cortés and María José Escalona
Department of Computer Languages and Systems
University of Seville
Av. de la Reina Mercedes S/N, 41012 Seville, Spain
e-mail: {sergiosegura,benavides,aruiz,mjescalona}@us.es

Abstract
Building Web design models from requirements specification is recognised as a time-consuming and error-prone task. In this context, some MDA-based approaches propose using metamodels and CIM to PIM transformations in order to provide a systematic method to transform Web requirements models into Web design models. However, the specific tool support for such transformation is still very limited. In this paper we illustrate how graph transformations can be used as a suitable technology and associated formalism to automate the transformations from Web requirement models to Web design models. In particular, we clarify our proposal by detailing how transforming instances of the WebRE metamodel to instances of the UWE metamodel using the AGG System.

1 Introduction
The success of software development mainly relies on the ability to implement programs fulfilling all the identified requirements. Such task is traditionally delegated to skilful developers who base the design on its knowledge and previous experience. However, this frequently results on a time-consuming and error-prone activity incrementing costs and reducing software quality.

Model Driven Development (MDD) [20] proposes rising the abstraction level during software development from code to models. The main goal is to concentrate on the high-level concepts while automating as much as possible the low-level details. Roughly speaking, MDD is based on the definition of metamodels and model transformations. A meta-model is a special model establishing the constructs and constraints to build models within a concrete domain. On the other hand, model transformations set how a model can be transformed into another model or even code. In this context, model-to-model transformations are usually defined by means of metamodel mapping, i.e. transformation rules relating pattern from the source metamodel to patterns in the target metamodel.

In [13] Koch et al. propose filling the gap between requirements and Web design by using model transformations in the context of MDD. In particular, they propose transforming Web requirements model into Web system design models by means of model transformations. In this context, they illustrate their proposal by defining a set of formal transformation rules between models defined as instances of the Web Requirements Engineering metamodel (WebRE, [6]) and instances of the UML-based Web Engineering metamodel (UWE, [11]). For such purpose, they use the standard Query View Transformation language (QVT, [10]).

Graph grammars and graph transformations are a very mature approach used since more
than 30 years ago for the generation, manipulation, recognition and evaluation of graphs [19]. Most of visual languages can be interpreted as a type of graph (directed, labelled, etc.). This makes graph transformations to be a natural and intuitive way for transforming models [7, 8, 14, 15]. In contrast with other model transformation approaches [7], graph transformations are defined in a visual way and are provided with a set of mature tools to define, execute and test transformations.

In this paper we take a first step toward automated tool support for the mapping from requirement to Web design. For such end, we show how graph transformations can be used as a suitable technology and associated formalism to implement the transformations from Web requirements models to Web design models. In order to illustrate our proposal we base on the formal transformation rules proposed by Koch et al. in [13]. In particular, we introduce our proposal by detailing how implementing one of such rules in the AGG system.

The application of graph transformation to the Web engineering domain is not a novel idea. Cáceres et al. [5] propose a method to obtain the navigation model of a Web Information System starting from the conceptual data model and the user requirements. OOWS [22] also proposes graph transformation as a good strategy to transform requirements models into navigation design models. In [21] Valderas et al. go even further using graph transformations to produce Web prototypes from Web requirements models. Compared to them, we base our transformations on MOF metamodels [17] and QVT transformation rules clarifying how graph transformations can fix with the standards used in the context of MDA.

The remainder of this paper is structured as follows: Section 2 gives an overview of graph grammars and graph transformations. An introduction to the WebRE and UWE metamodels is presented in Section 3. In Section 4 we detail our proposal by using an example inspired in the work of Koch et al.. Finally, we describe our future work and summarize our main conclusions in Section 5.

2 Graph Grammars and Graph Transformations

Graph Grammars are a very mature approach used since 30 years ago for the generation, manipulation, recognition and evaluation of graphs [19]. Since then, graph grammars has been studied and applied in a variety of different domains such as pattern recognition, syntax definition of visual languages, model transformations, description of software architectures, etc. This development is documented in several surveys, tutorials and technical reports [1, 2, 3, 8, 14, 15].

Graph grammars can be considered as the application of the classic Chomsky’s string grammars concepts to the domains of graphs. Hence, a graph grammar is composed by an initial graph, a set of terminal labels and a set of transformation rules (sometime also called graph productions). A transformation rule is composed mainly by a source graph or Left Hand Side (LHS) and a target graph or Right Hand Side (RHS). The application of a transformation rule to a so-called host graph, also called direct derivation, consists on looking for an occurrence of the LHS graph in the host graph. If such matching is found, the occurrence of the LHS in the graph is replaced by the RHS of such rule. Thus, each rule application transforms a graph by replacing a part of it by another graph. The set of all graphs labelled with terminal symbols that can be derived from the initial graph by applying the set of transformation rules iteratively is the language specified by the graph grammar.

The application of transformation rules to a given graph is called Graph Transformations. Graph transformations are usually used as a general rule-based mechanism to manipulate graphs. Most of visual modelling languages can be interpreted as a type of graph (directed, labelled, attributed, etc.). This make graph transformations to be recognized as a suitable technology for the specification and application of model transformations [7, 8, 14, 15]. Hence, as documented in the literature, the reasons to select graph transformations as a suitable approach for model transformations
Graph transformations are a natural and intuitive way of performing pattern-based visual model transformations.

The maturity of graph transformations has provided it with a solid theoretical foundation in form of useful properties. Hence, for instance, the "invertability" property details under what conditions a transformation rule can be inverted.

There is a variety of mature tools to define, execute and test transformations rules. Fujaba\textsuperscript{1} and the AGG System\textsuperscript{2} are two of the most popular general-purpose graph transformation tools within the research community. Nevertheless, other specific tools such as GReAT\textsuperscript{3} or VIA-TRA2\textsuperscript{4} are also starting to emerge as a consequence of the increasing popularity of graph transformations in the model-driven development domain.

3 Web Metamodels

Model Driven Development (MDD) is mainly oriented to the separation between platform independent design and platform specific implementations of applications. This is one of the main reasons that explain the increasing popularity of MDD in the Web Engineering domain\cite{12} in which the problems caused by the changing technologies are a routine. In this context, the popularity of MDA\cite{16} has motivated the development of new MOF\cite{17} metamodels and UML profiles\cite{18} making possible the interoperability between different development tools.

In\cite{13} Koch \textit{et al.} focus on the early step of the MDA approach and uses two of these metamodels to define the transformation from requirement models (CIM) to Web design models (PIM). In particular, they use the so-called Web Requirements Engineering metamodel (WebRE, [6]) and the UML-based Web Engineering metamodel (UWE, [11]) as the source and target metamodels respectively. In the next sections an overview of both metamodels is presented.

3.1 Web Requirements Metamodel

The Web Requirement Engineering metamodel (WebRE, [6]) (Figure 1) summarizes the main activities and elements used when modeling Web system requirements. It is composed by two packages: the WebRE Behaviour package and the WebRE Structure package.

The WebRE Behaviour package includes the metaclasses to model the Web System behaviour requirements. It comprises the general concepts of Web user, navigation, browse, search, Web process and user transaction. A WebUser is any user who interact with the Web System, registered or not. The navigation of the Web user through the system is modelled by means of the Navigation metaclass which consists of a set of browse activities that the Web user performs to reach a

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{webre_metamodel.png}
\caption{WebRE Metamodel (extracted from [12])}
\end{figure}
target node. A Browse activity represents the action of following a link. The special browse activity Search models that the Web user performs a query to the Web system. Finally, the WebProcess metaclass is used to model navigations including user transactions like changing a password or providing payment data.

The WebRE Structure package includes the metaclasses to model how the Web System manages and displays the information. Such information is modeled at three levels: nodes, pages and content. A Node is a point of the navigation where the user finds information. Nodes will be usually associated to one or more Web pages modeled by means of the metaclass WebUI. Finally, each piece of information of a Web system is represented as an instance of the metaclass Content.

3.2 Web Design Metamodel

The UML-based Web Engineering metamodel (UWE, [11]) defines the elements to model the high-level design concepts of Web systems related to content, navigation structure, business process and presentation. As an example, Figure 2 shows the metaclasses and associations of the UWE navigation package. The main navigation elements are navigation nodes and links modeled as instances of the classes NavigationNode and Link respectively. There are several kinds of navigation nodes and links defined. Hence, for instance, navigation classes present the content of the Web application to the user meanwhile navigation links are the link which take to them.

4 From Web Requirements to Web Design

In this section we detail our proposal by detailing how graph transformations can be used to automate the mapping from Web requirements models to Web system design models. Firstly, we introduce the AGG System. Next, we present our proposal by detailing an example.

4.1 The AGG System

In order to test our proposal we implemented it in one of the most popular tool within the graph grammar community: The Attributed Graph Grammar System (AGG). Roughly speaking, AGG is a free Java graphical tool for editing and transforming graphs by means of graph transformations. The AGG System is a prototype implementation of the algebraic approach to graph transformation supporting Contextual Layered Graph Grammars (CLGGs, [4]). In CLGGs the set of productions is classified into ordered layers. To transform a graph, productions are applied layer by layer from layer 0 to layer N (cyclically if needed) until no rules can be applied. AGG provides a flexible graph editor and a useful component to apply user-selected productions to a given graph. In addition, the AGG system can be used as a general purpose graph transformation engine in any dedicated Java applications employing graph transformation methods. All this reasons made us to select AGG as a suitable tool to implement our proposal.

AGG graph transformation rules consist on three parts: a left-hand and a right-hand side graph, a mapping morphism between nodes and edges on both sides and a set of Negative Application Conditions (NACs). NACs are graph patterns establishing under what conditions the rule will not be applied. Figure 3 shows a screenshot of the AGG GUI. On the left hand side a tree view display the working graph and the rules of the proposed grammar. In the upper central area the NAC (if any) and the LHS and RHS graphs of the selected rule are displayed. Finally, the central area is reserved for the host graph (also called working graph).

4.2 Automating Transformations

In order to detail our proposal we use an example based on the work of Koch et al. [13]. In particular, we present the set of AGG transformation rules needed to transform automatically a Search activity from the requirement model (instance of WebRE) into a Query
Figure 2: UWE Metamodel: Navigation package (extracted from [12])

Figure 3: The AGG System
in the design navigation model (instance of UWE). Figure 4 shows the formal definition of such transformation in QVT. As detailed in [13], each search activity at the requirement level is transformed into a Query, an Index, a NavigationClass and two Links relating the query, the index and the navigation class. The NavigationClass in the navigation model represents the results of the query. In a similar way, the parameters of the search (Content) are transformed into attributes (Navigation-Attribute) of the query.

An important aspect to consider when working with AGG is that it works exclusively with the graphs created using its editor and not with external models. Thus, input models must be represented as AGG graphs before applying transformations. In a similar way, once the transformation is performed, the obtained graph must be translated to the target model. The translation of a model to an AGG graph and vice versa is out of the scope of this paper. We refer the interested reader to the work of Valderas et al. [22] who propose using XML as a suitable strategy for the translations model-to-graph and graph-to-model in the context of AGG.

Figure 5 show the set of AGG transformation rules implemented to perform the sample QVT rule showed in Figure 4. From left to right, the NAC, LHS and RHS graphs of each rule are presented.

Objects used in requirement models can be part of different rules creating different structures in the design model. This is known in the literature as one-to-many transformations [14]. When implementing this kind of transformation the source pattern must not be replaced by the target one. Instead of that, it is recommended using a helper structure to link the source and target graph 'marking' the transformation [8, 14]. Such structures are later used in the NACs to check if the transformation rule has been already applied. The oval nodes and dashed lines used in the rules of Figure 5 represent such structure.

Rule 1 depicts the transformation of a Search object into a Query. Firstly, AGG look for a search object in the host graph (LHS). Next, the transformation engine verify that the search object is not linked to a query with the same name by means of a 'S2Q' node (NAC), i.e. the transformation rule has not been already applied. Finally, if the NAC is not fulfilled, AGG create a query node with the same name than the search object and link them using a 'S2Q' node (RHS).

Rule 2 illustrates how the parameters of the search (Content instances) are transformed into navigation attributes with the same name representing the attributes of the query. Finally, Rule 3 shows how the results of the search is transformed into an index, a navigation class and two links relating the search, the index and the navigation class.

Figure 6 depicts the result of executing the set of AGG transformation rules to a sample input requirement model.

5 Conclusions and Future Work

In this paper we take a first step toward automated tool support to transform Web requirements models into Web system design models. For such end, we propose using graph transformation as a suitable technology and associated formalism for implementing such transformations. In order to illustrate our proposal we base on the transformation rules formally defined by N. Koch et al. in [13]. In particular, we present the set of AGG transformation rules needed to transform a Search activity in the WebRE metamodel (requirements level) to a Query in the UWE metamodel (design level). In contrast with other related proposals, we base our transformations on MOF metamodels and QVT transformation rules illustrating how graph transformations can be used with the standards used in the context of MDA.

Many challenges remain for our future work. AGG is a suitable tool but it still has some drawbacks. AGG does not work with standard notations such as MOF or UML making necessary the translations of the metamodels to its internal notation. Additionally, in contrast with the QVT language, AGG rules can not be structured by using inheritance and composition mechanisms making it a not scalable
Figure 4: QVT Transformation Rule: Search TO Query (extracted from [12])

Figure 5: AGG Transformation Rules: Search TO Query
Figure 6: Sample Transformation

proposal. In order to get over these difficulties, we are studying other more powerful but complex approaches such as MOMENT\(^5\).

Finally, we consider that integrating our proposal in a CASE tool such as ArgoUWE\(^6\) or NDT-Tool [9] would be useful and we plan to work in that direction too.

References


\(^5\)http://moment.dsic.upv.es/
\(^6\)http://www.pst.informatik.uni-muenchen.de/projekte/uwe/argouwe.shtml


