Modeling Business Processes in Web Applications with ArgoUWE*

Alexander Knapp[1], Nora Koch[1,2], Gefei Zhang[1], and Hanns-Martin Hassler[1]

[1] Ludwig-Maximilians-Universität München, Germany
{knapp@pst, kochn@pst, zhangg@pst, hassler@cip}.ifi.lmu.de
[2] F.A.S.T. GmbH, Germany
koch@fast.de

Abstract. The CASE tool ArgoUWE supports the systematic design of Web applications using the UML-based Web Engineering (UWE) approach. The design methodology of UWE is based on a metamodel which is defined as a lightweight extension of the UML metamodel in the form of a profile and comprises the separate modeling of the different aspects of a Web application: content, structure, layout, and business logic. ArgoUWE is implemented as a plugin into the open-source tool ArgoUML. In this paper, we focus on the latest improvements of the ArgoUWE tool: On the one hand, ArgoUWE supports the design of workflow-driven Web applications where business logic can be captured by process structure and process flow models. On the other hand, ArgoUML's design critic mechanism has been extended to indicate deficiencies and inconsistencies in UWE models based on the UWE metamodel and its OCL well-formedness rules.

1 Introduction

Most of the existing approaches supporting the development process of Web applications were born with the aim to develop Web Information Systems (WIS) [3,6,9,14], whose main focus is to retrieve, adapt and present information to the users. However, Web applications are evolving to software systems supporting complex business processes in the same way as non-Web software does. To address the complexity of these business processes software development methods usually include the definition of an explicit business process view. In contrast, such business processes have been only tangentially tackled in most existing Web Application development approaches. This work centers its attention on such a business process view in the design of Web applications.

Web engineering always implies the employment of systematic, disciplined and quantifiable methods and tools in the construction of the applications. UWE and ArgoUWE are such a method and such a tool, respectively. UWE is the acronym for "UML-based Web Engineering" [9]. The model elements of UWE are defined in a UML profile—a conservative extension of the UML [13]—and the iterative and incremental development process is based on the Unified Software Development Process [7]. The extension comprises appropriate elements to permit separate modeling of

* This work has been partially supported by the EU project AGILE (IST-2001-32747), the DFG project InOpSys (WI 841/6-1), and the BMBF project MMISS (08NM070D).

the conceptual, navigational, business logic and presentational aspects of Web applications, guaranteeing the consistency between these models.

We have extended the CASE tool ArgoUML into a tool for UWE-based Web application development, called ArgoUWE[3] [8]. The distinguishing features of ArgoUWE are support of visual modeling, UML conformance and open source characteristics. This tool provides on the one hand tailored editors for the UML-based notation used for the conceptual, navigation, presentation and business process modeling of Web applications. On the other hand, it offers several semi-automatic model transformations that are defined in the UWE development process [9]. As these model transformations are based on the UWE-metamodel, both consistency between the different models and integrity of the overall Web application model with respect to UWE's OCL constraints are ensured by the tool. In particular, in a conceptual model classes can be marked for navigation and navigation classes can be derived from annotated conceptual classes. Furthermore, in the navigational model ArgoUWE can add automatically access primitives such as queries and menus. ArgoUWE is implemented as a plugin module of the open-source ArgoUML [16] modeling tool. It fully integrates the UWE metamodel [10] and provides an XMI extension. The construction process of Web applications is supported by incorporating the semi-automatic UWE development steps as well as the OCL well-formedness rules of the UWE metamodel which can be checked upon request.

We present in this work a new version of ArgoUWE, which includes new modeling facilities to support the design of process-driven Web applications. A business process model and new stereotyped classes are included as well as a set of additional constraints that are used for consistency checks and semi-automatic generation of process models. In contrast to the old version, consistency of models is now checked in the background during modeling. This way the developer is supported but not constrained in his modeling activities with suggestions for corrections and improvements.

In contrast to the Web application development tool VisualWADE, which is based on the Web engineering method OO-H [6], UWE and ArgoUWE use the UML throughout the complete development process. Furthermore, ArgoUWE does not rely on proprietary standards, like the Web modeling language WebML [1] that is used in the WebRatio tool. ArgoUWE can be viewed as complementary to work focusing on the technological aspects of Web applications, such as [2,12].

This paper is organized as follows: Section 2 provides an overview of the UWE approach. Section 3 describes how ArgoUWE supports the systematic design of Web applications focusing on the modeling of the business logic. Section 4 presents those special features of ArgoUWE implemented to guarantee model consistency. Section 5 presents the novel implementation aspects of business logic and the so-called design critics in the current version of ArgoUWE. Section 6 gives an overview of related work. Finally, in the last section some concluding remarks and future work are outlined.

2 UML-based Web Engineering

UML-based Web Engineering (UWE) has been continuously extended since 1999; for a summary see [10]. UWE supports Web application development with special focus on

[3] `http://www.pst.ifi.lmu.de/projekte/argouwe`

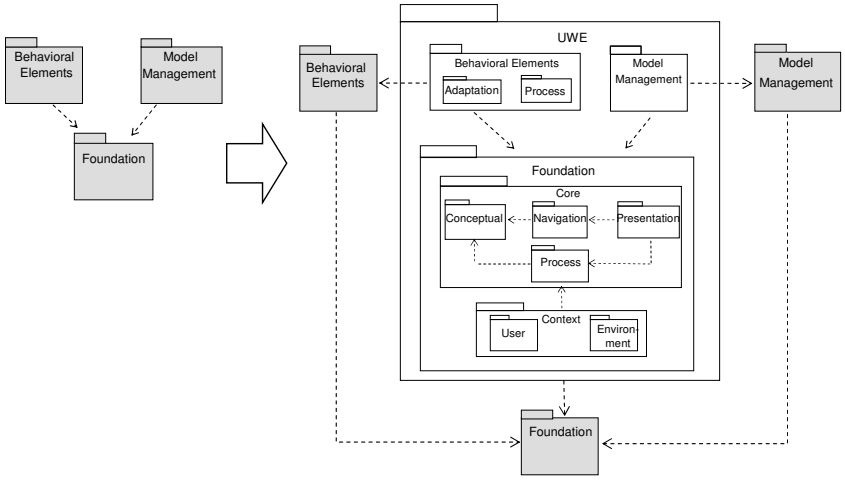
systematization. Being a software engineering approach it is based on the three pillars: process, notation and tool support. The focus of this article is the tool support and we will only give a brief overview of the UWE approach and the underpinning UWE metamodel here. Section 3 gives some details about the UWE process and the notation used when showing how the ArgoUWE tool supports the design of Web applications.

The UWE process is object-oriented, iterative and incremental. It is based on the Unified Process [7] and covers the whole life cycle of Web applications focusing on design and automatic generation [10]. The UWE notation used for the analysis and design of Web applications is a lightweight UML profile [13] published in several previous articles. It is a UML extension based on the extension mechanisms defined by the UML itself, i.e., the extension is performed by the definition of stereotypes, tagged values and OCL constraints. These modeling elements are used in the visual representation of the requirements and in the design of the conceptual model, the navigation structure, the business logic and the presentation aspects of Web applications (see Sect. 3).

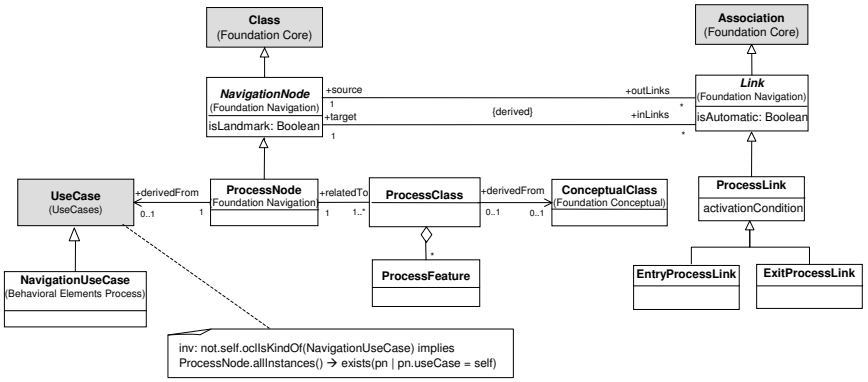
We defined the UWE metamodel as a conservative extension of the UML 1.5 metamodel. Conservative means that the model elements of the UML metamodel are not modified. Instead, all new model elements of the UWE metamodel are related by inheritance to at least one model element of the UML metamodel. We define additional features and relationships for the new elements. Analogous to the well-formedness rules in the UML specification, we use OCL constraints to specify the additional static semantics of these new elements. The resulting UWE metamodel is profileable, which means that it is possible to map the metamodel to a UML profile. In particular, UWE stays compatible with the MOF interchange metamodel and therefore with tools that are based on the corresponding XML interchange format XMI. The advantage is that all standard UML CASE tools which support UML profiles or UML extension mechanisms can be used to create UWE models of Web applications. If technically possible, these CASE tools can further be extended to support the UWE method. ArgoUWE presents an instance of such CASE tool support for UWE based on the UWE metamodel.

We only briefly review the UWE metamodel here; for further details see [10]. The extension of the UML metamodel consists of adding one top-level package UWE to the three UML top-level packages. This UWE package contains all UWE model elements, which are distributed in sub-packages (see Fig. 1(a)). The structure of the packages inside the UWE package is analogous to the UML top-level package structure (shown in grey). The package Foundation contains all basic static model elements, the package BehavioralElements depends on it and contains all elements for behavioral modeling and finally the package ModelManagement, which also depends on the Foundation package, contains all elements to describe the models themselves specific to UWE. These UWE packages depend on the corresponding UML top-level packages. Figure 1(b) shows part of the UWE metamodel classes in UWE::Foundation::Core::Process. Note that the separation of concerns of Web applications is reflected by the package structure of the package UWE::Foundation of the UWE metamodel in Fig. 1(a).

The UWE methodology provides guidelines for the systematic and stepwise construction of models for Web applications. The precision can be augmented by the definition of constraints in the OCL.



(a) UWE metamodel embedded into the UML metamodel.



(b) Part of UWE metamodel classes in UWE::Foundation::Core::Process.

Fig. 1. UWE metamodel.

3 Modeling with ArgoUWE

We present the main steps of modeling Web applications using the CASE tool ArgoUWE by means of a simplified example of an e-shop. The user of this e-shop can search for and select products, add them to his shopping cart, and view the content of the shopping cart. If he has signed in as a customer, he can also order the items stored in his shopping cart following a checkout process.

Working with ArgoUWE is intuitive to ArgoUML users. In particular, ArgoUWE makes use of ArgoUML's general graphical user interface. The *project browser* is di-

vided into several panes, see Fig. 4. The *navigation pane* (1) lists all diagrams and model elements of the current project in a tree view. A single UWE diagram is edited in the *multi-editor pane* (2). In the *critique pane* (3) a list of design critics issues is shown. The *detail pane* (4) serves several different purposes: besides a to do list, details of the currently selected model element are shown and edited here: meta attributes, stereotypes, tagged values, and OCL constraints. Even code skeletons can be generated for a selected model element and shown in the detail pane.

3.1 Starting with the Use Case Model

The first step towards modeling a Web application using UWE is the elicitation of requirements which is laid down in a use case model, see Fig. 2.

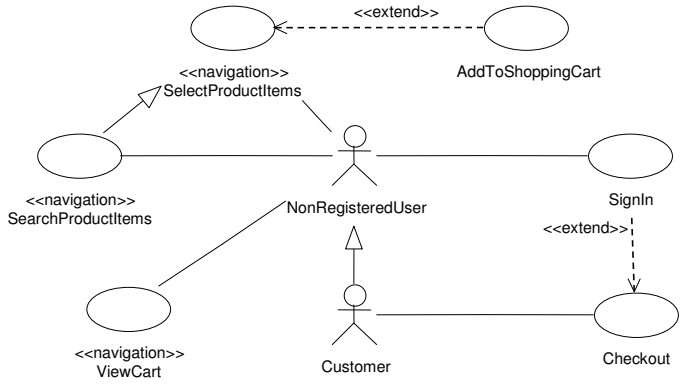


Fig. 2. Use case model of the e-shop example.

UWE distinguishes between two kinds of use cases: ≪navigation≫ use cases represent navigational activities of the user, like the selection of product items and viewing the shopping cart. On the other hand, standard UML use cases capture the business logic of the Web application. These use cases will be refined later in the step of process modeling. Both types of use cases can be integrated in the same use case model.

For implementation reasons of ArgoUWE as a plugin module a new type of diagram UWE use case diagram is defined, which only differs from the UML use case diagram in that it allows for ≪navigation≫ use cases.

3.2 Building the Conceptual Model

Based on the requirements analysis, the content of a Web application in UWE is modeled in a conceptual model, represented by a UML class diagram. The main model elements are conceptual classes, associations, and packages. Figure 3 shows the conceptual model of the e-shop example, where a Customer has one or more ShoppingCarts which contain a set of ShoppingItems that can be ordered by the customer.

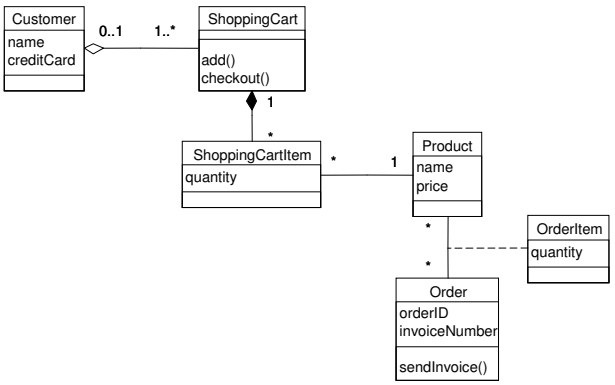


Fig. 3. Conceptual model of the e-shop example.

The conceptual model does not contain any information pertaining to the navigational aspects of the Web application, but it forms the basis for constructing the navigation model. In fact, by marking conceptual classes as relevant to navigation the designer indicates that the instances of these classes contain information that will be shown to the user. In the e-shop example, all conceptual classes are marked as navigation relevant; ArgoUWE supports the selection via a pop-up menu.

3.3 Generating and Enriching the Navigation Model

Based on the requirement analysis and the content modeling, the navigation structure of a Web application is modeled. Navigation classes are navigable nodes of the hypertext structure, navigation links represent direct links between navigation classes. Alternative navigation paths are handled by menus. Access primitives are used to reach multiple instances of a navigation class (≪index≫ or ≪guided tour≫), or a selection of items (≪query≫).

ArgoUWE supports the Web application designer in generating a navigation model from the conceptual model. When the designer selects Navigation Diagram from the menu Create Diagram, ArgoUWE generates navigation classes and associations for all "navigation relevant" conceptual classes and all associations between them into a new navigation model. The modeler can add some links after automatic creation of the navigation diagram. ArgoUWE automatically generates indexes and menus when one of the encircled buttons in Fig. 4 is selected: An index is added between two navigation classes related by an association whenever the multiplicity on the target end is greater than one. Menus are added to every class that has more than a single outgoing association (for a more detailed account see [8]). For instance, CustomerOrders and MainMenu will be automatically included in the e-shop example (see Fig. 4).

In Web applications which are business logic intensive the business processes have to be integrated into the navigation structure indicating entry and exit points to the process nodes, see Fig. 1(b). In UWE, each business process that is modeled by a use case can be added to the navigation model as a corresponding ≪process node≫.

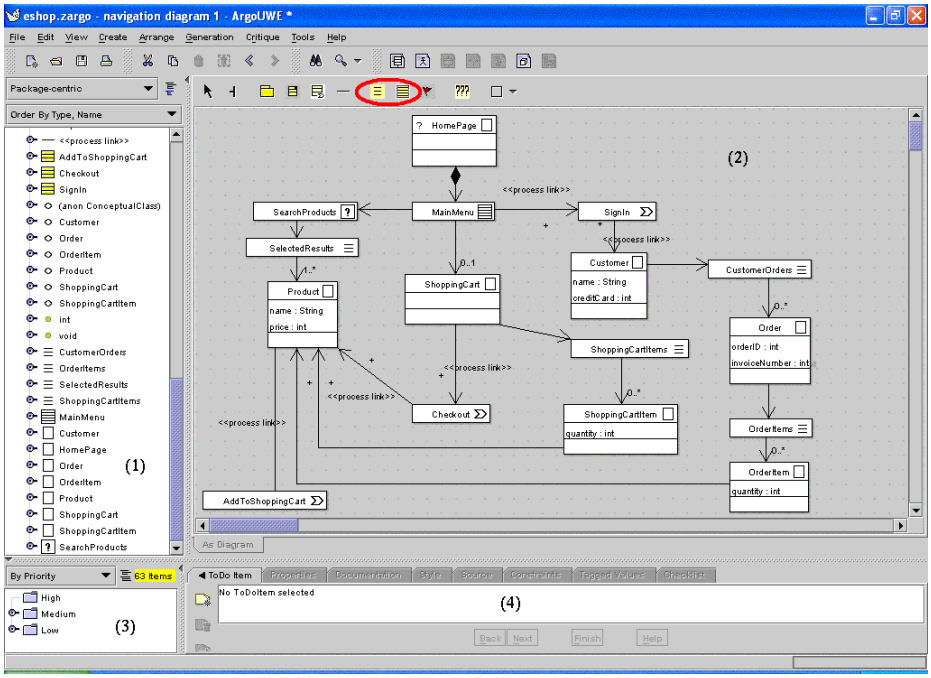


Fig. 4. Navigation model of the e-shop example after integration of business processes.

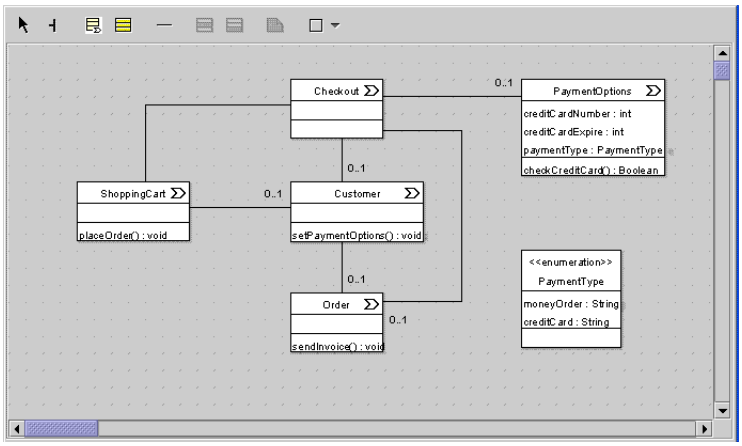
A link connecting such a process node to a navigation node is called a ≪process link≫. Figure 4 shows the enriched navigation model of the e-shop example where SignIn, AddToShoppingCart, and Checkout have been integrated.

3.4 Constructing the Process Model

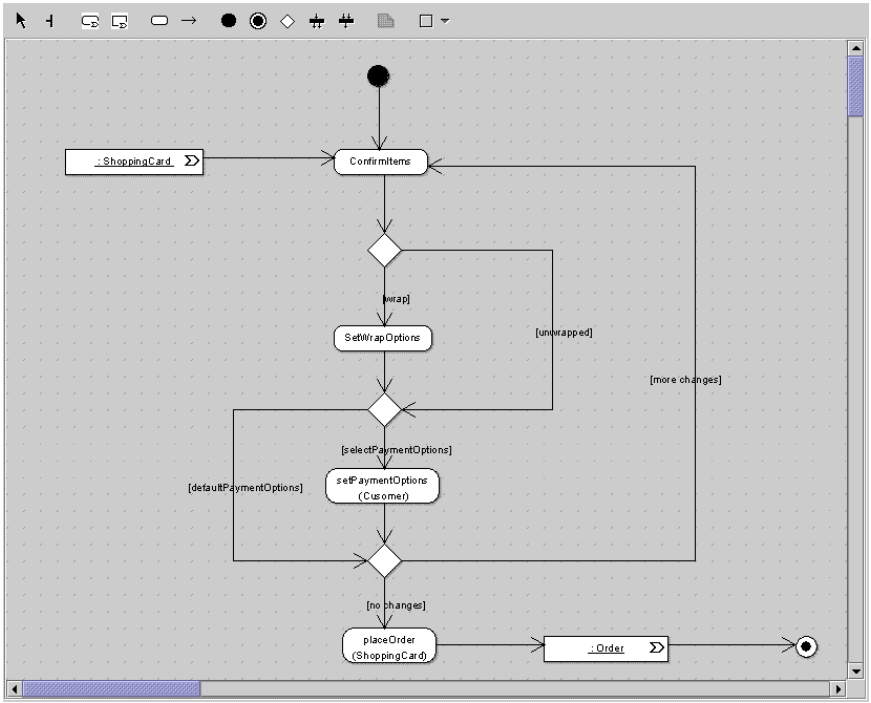
In this step of the UWE design process each process node is refined in a process model, consisting of a process structure model and a process flow model. A process structure model is represented by a UML class diagram and describes the relationship of a ≪process node≫ and other ≪process class≫es whose instances are used to support this business process. The logic of the business process is described by a process flow model visualized as a UML activity diagram.

ArgoUWE generates a process node in the navigation model for each (non-navigational) use case that is manually selected by the modeler. Thereby, a process class is generated for the process node of the selected use case and automatically included in the process structure model. Auxiliary process classes can be added manually. Figure 5(a) shows that the process node Checkout is supported by the process classes Customer, ShoppingCart, Order, and PaymentOption.

Figure 5(b) shows a part of the process flow model of the check out process in the e-shop example. In the business process Checkout a user can confirm the product items he gathered in his shopping cart. Additionally, he can set wrapping and payment



(a) Process structure model.



(b) Process flow model.

Fig. 5. Process model of checkout of the e-shop example.

options. The activity setPaymentOptions has been included in the process flow model by selecting the corresponding operation of class Customer from the process structure model. Similarly, the objects ShoppingCart and Order are instances of classes in the process structure model.

3.5 Sketching the Presentation Model

Based on the navigation model, a presentation model is created in the UWE design process to model the abstract layout of the Web pages. In ArgoUWE, a logical presentation model can be inferred from the navigation model by choosing the menu item Presentation Diagram from the menu Create Diagram. For each navigation node, each access primitive, and each process node a corresponding presentation element is created and added to the presentation model (for a more detailed account see [8]). Such a presentation model only sketches the Web pages. Layout details like relative position, colors and size of layout elements cannot be represented in presentation diagrams.

4 ArgoUWE Design Critics

One of the distinguishing features of ArgoUML compared to other modeling tools is the support of cognitive design critics offered to the designer. During run time, a thread running in the background keeps checking if the model built by the user shows deficiencies. For each deficiency found a design critique item will be created and added to the design critique pane. Design critics not only warn the user that his design is still not perfect but can also, by means of a wizard, help the user improve the design. The design critics range from incompleteness, such as lacking names of model elements, to inconsistency, such as name collisions of different attributes or operations in a class. Furthermore, design critics also suggest the use of certain design patterns such as the singleton pattern [5]. Design critics are not preemptive and never interrupt the designer. Instead, they simply give warnings and post items to the designer's "to do" list shown in the detail pane.

ArgoUWE inherits the design critics feature from ArgoUML. In fact, all well-formedness constraints of UWE have been fully integrated and are continuously checked by ArgoUWE in the background at runtime. In Fig. 6 the highlighted design critique item in the design critique pane indicates that a process use case does not show a corresponding process node yet; this critique corresponds to the following UWE metamodel constraints:

```
context ProcessNode
inv: not self.useCase.oclIsKindOf(NavigationUseCase)

context UseCase
inv: (not self.oclIsKindOf(NavigationUseCase)) implies
     ProcessNode.allInstances()->exists(pn | pn.useCase = self)
```

In the detail pane, a two-step wizard designed for this critic is leading the user to correct this defect by creating a process model. In the first step, a new process structure model has to be created and in the second step, a new process flow model.

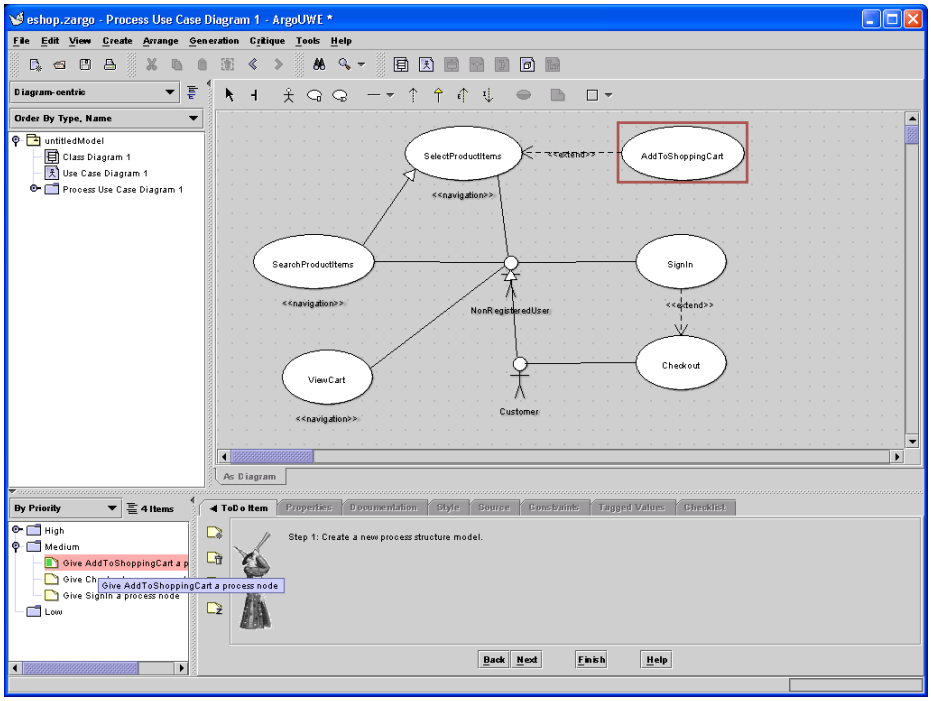


Fig. 6. Design critique concerning the checkout use case.

5 Architecture of ArgoUWE

The ArgoUWE tool is implemented as a plugin into the open-source UML modeling tool ArgoUML (version 0.15.7), both written in Java. ArgoUML provides a suitable basis for an extension with UWE tool support by being based on a flexible UML metamodel library (NSUML [18], version 1.3/0.4.20) and a general graph editing framework (GEF [17], version 0.10.2), as well as featuring an extendable module architecture. These feature characteristics and also the fact that ArgoUML is an open-source tool with an active developer community lead us to favoring ArgoUML as development basis over other commercial tools—although the open source code of ArgoUML has sometimes to offset its rather poor documentation. However, tools like Rational Rose™ or Gentleware's Poseidon™ would also afford the necessary extension prerequisites, perhaps with the exception of metamodel extensions.

5.1 ArgoUWE Metamodel

The "Novosoft UML library" (NSUML), on which ArgoUML is based, not only provides a library for working with UML 1.3 models in terms of Java objects, but also contains an XML/XMI-based generator for arbitrarily changed and extended (UML) metamodels. As UWE uses additional modeling concepts targeted onto Web applications, ArgoUWE uses NSUML to generate an extended UML/UWE metamodel that

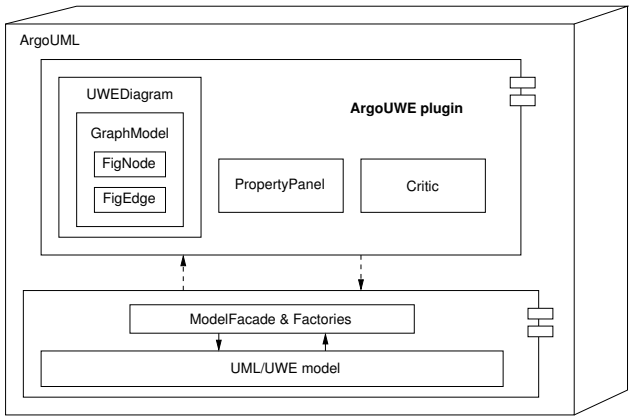


Fig. 7. Overview of the ArgoUWE plugin architecture.

again allows the programmer to handle UWE entities in a seamless manner. In particular, we chose a "heavyweight extension" for the physical metamodel that is generated by NSUML. Alternatively, we could employ the UWE lightweight UML profile directly. However, stereotyping and tagging is not compatible with the concept of overloading in object-oriented programming. For the current ArgoUML version, the adaptations of the UML metamodel merely consist of extending the NSUML generator resource files by the UWE metaclasses ConceptualClass, NavigationNode, ProcessNode, etc.

5.2 Plugin Architecture

In ArgoUML, the model is encapsulated in an instance of the (extended) NSUML library class ru.novosoft.uml.model_management.MModel and accessed through the facade org.argouml.model.ModelFacade. Thus, manipulations of the model have a single access point, all effects of model manipulations are disseminated to other components by a general observer mechanism following the Model-View-Controller paradigm. Figure 7 summarizes the general structure of the ArgoUML model, view, and control devices as used in ArgoUWE. The UWE diagram kinds: conceptual diagram, navigation diagram, process structure diagram, process flow diagram, and presentation diagram are straightforwardly supported by introducing new subclasses of org.argouml.uml.diagram.ui.UMLDiagram, more specifically the common superclass org.argouml.uml.diagram.ui.UWEDiagram.

A UMLDiagram captures the graphical presentation and manipulation of model elements. It is based on a bridge, the graph model, between the model realized by NSUML library classes and the graphical presentation using the "Graph Editing Framework" (GEF) library classes. The bridges for the structural UWE diagrams are all derived from the class org.argouml.uml.diagram.static_structure.ClassDiagramGraphModel whereas the bridge for the process flow diagram inherits from the class org.argouml.uml.diagram.state.StateDiagramGraphModel.

Each UWE model element is linked to a figure node (org.tigris.gef.presentation.FigNode) or a figure edge (org.tigris.gef.presentation.FigEdge) of the GEF library. In addition to manipulating the model elements graphically, they can also be changed and edited by using property panels that are implemented as subclasses of org.argouml.uml.ui.PropPanel. ArgoUWE adds property panels for conceptual classes, navigation classes, etc., which are installed on the ArgoUML platform automatically with the diagram counterparts by reflection. Finally, the ArgoUWE semi-automatic editing functionalities induced by the UWE method (see Sect. 3) are triggered in a UWEDiagram.

The ArgoUWE consistency checking mechanisms are packaged into ArgoUML design critics: Each well-formedness rule of a UWE model is implemented as a class inheriting from CrUWE, i.e., the abstract UWE model critic that extends org.argouml.cognitive.critics.Critic. By registering a critic with org.argouml.cognitive.critics.Agency a so-called designer thread of class org.argouml.cognitive.Designer will continuously check whether the critic applies to the current model; the critic will then post a critique in form of a org.argouml.cognitive.ToDoItem to the designer thread.

The UWE extensions are packaged in a plugin module. The original ArgoUML user interface is extended by a class implementing the extension point class org.argouml.application.api.PluggableMenu, registering the new diagram types and their support. This extension, when put into the extension directory (./ext) of ArgoUML, is loaded automatically on ArgoUML start-up. However, it must be noted that the UWE extension of ArgoUML is not completely orthogonal to ArgoUML as the underlying metamodel has been changed. Nevertheless, packaging the UWE extensions as a plugin module insulates these extensions from the continuous changes to ArgoUML.

6 Related Work

Many methods for the development of Web applications have been proposed since the middle of the nineties. An excellent overview is presented by Schwabe [15] where the most relevant methods, such as OOHDM [14], OO-H [6], WSDM [3], and UWE [9] are described on the basis of a same case study. But only some of these methods support the systematic development with a CASE tool. The most advanced tool support is offered for both, the method OO-H and the modeling language WebML.

VisualWADE is the tool, which provides an operational environment supporting the OO-H method. In contrast to our ArgoUWE, it uses the UML only in the first phase of the development process. A default presentation is obtained from the navigation model similarly as in ArgoUWE. VisualWADE includes authoring tool features that allow the designers to render the final look and feel of the application. VisualWADE provides model compilers for PHP/mySQL, PHP/Oracle and PHP/SQL server technologies.

In WebRatio, Web applications are specified using an entity-relationship (ER) model for data requirements, and the proprietary Web Modelling Language (WebML [1]) for the functional requirements. This approach differs from UWE as it does not perform a clear separation of the navigation and presentation aspects. The WebRatio development architecture includes a graphic interface for editing ER and WebML schemas, and customizable code generators for transforming ER specifications into relational table definitions for any JDBC or ODBC compliant data source, and WebML specifications into

page templates for the J2EE and .NET architectures. WebRatio internally uses XML and XSL as the formats for encoding both the specifications and the code generators: XML is used for describing data and hypertext schemas, whereas XSL is used for generating the graphic properties and layout of the page templates, for validity checking, and for automatic project documentation.

A more architecture-oriented approach is proposed by Conallen [2]. It extends the UML to support the design of Web applications focusing on current technological aspects of the implementation and is based on the generic RUP [11] development process. The notation is supported by the Rational Rose™ tool, but in contrast to ArgoUWE it neither supports a systematic development process nor guides the developer through the Web-specific process.

Another metamodel for modeling Web-based user interfaces and an associated notation is proposed by Muller et al. [12]. The modeling elements of this approach are defined as specializations of the very general UML ModelElement. A kind of graph is used to represent the interface behavior and a class diagram is used to represent business classes. The action language Xion that augments OCL with Java-like structures is used to describe the behavior of the methods. The main disadvantage of such an approach is that it requires the implementation of its own editor and development environment.

Our approach focuses on the structures and the workflows of business processes in Web applications. Furthermore, ArgoUWE supports the integration of business processes into the navigation structure. In contrast, languages like ebXML [4] concentrate on the inter-process aspects—the collaborations and the choreography between business processes—rather than the intra-process aspects of business processes.

7 Conclusions and Future Work

We have presented the CASE tool ArgoUWE [8] that we have developed for the computer aided design of Web applications using the UWE methodology. The focus of this paper is to show how ArgoUWE was enhanced to support the development of more complex Web applications which include business processes like those that are frequently used in e-commerce systems.

ArgoUWE is built as a flexible extension of ArgoUML due to the plugin architecture facilities provided by the ArgoUML tool. We claim that the core of the CASE tool is the underlying UWE metamodel [10] defined as a conservative extension of the UML metamodel. ArgoUWE provides platform-independent models (PIMs) that will be used in the OpenUWE tool suite environment, which is currently under development, to achieve a model-driven generation of Web applications.

We outlined in this work the basic ideas behind the UWE methodology [9] and presented a running example to show how the tool supports the design of the main UWE models: use case model, conceptual, navigation, process and presentation models in a semi-automatic development process where user and tool-activities are interleaved. The support of the designer activities is also improved by the UWE well-formedness rules included in the design critics mechanism assisting the modeler in finding design errors or enhancing the models while designing with better modeling constructs. This consis-

tency checking mechanism allows the continuous verification of the rules in contrast to the former checking process that had to be explicitly triggered by the modeler.

We are currently working on minor improvements of the usability of ArgoUWE and to include better support for iterative and incremental modeling. Further, we will improve ArgoUWE to provide new UWE modeling elements as well as additional well-formedness rules needed in the design of personalized Web applications. Finally, we will enhance ArgoUWE to build platform-specific models (PSMs) for frequently used frameworks like Zope or Struts and architectures such as J2EE and .NET.

References

1. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan-Kaufmann, San Francisco, 2002.
2. J. Conallen. *Building Web Applications with UML*. Addison-Wesley, Reading, Mass., &c., 2nd edition, 2003.
3. O. de Troyer and C. J. Leune. WSDM: A User Centered Design Method for Web Sites. *Computer Networks and ISDN Systems*, 30(1–7):85–94, 1998.
4. ebXML. Business Process Specification Schema, Version 1.01. Specification, ebXML, 2001. `http://www.ebxml.org/specs/ebBPSS.pdf`.
5. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley, Boston, &c., 1995.
6. J. Gómez, C. Cachero, and O. Pastor. On Conceptual Modeling of Device-Independent Web Applications: Towards a Web-Engineering Approach. *IEEE Multimedia*, 8(2):26–39, 2001.
7. I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, Reading, Mass., &c., 1999.
8. A. Knapp, N. Koch, F. Moser, and G. Zhang. ArgoUWE: A CASE Tool for Web Applications. In *Proc. 1st Int. Wsh. Engineering Methods to Support Information Systems Evolution (EMSISE'03)*, Genève, 2003. 14 pages.
9. N. Koch and A. Kraus. The Expressive Power of UML-based Web Engineering. In D. Schwabe, O. Pastor, G. Rossi, and L. Olsina, editors, *Proc. 2nd Int. Wsh. Web-Oriented Software Technology (IWWOST'02)*, pages 105–119. CYTED, 2002.
10. N. Koch and A. Kraus. Towards a Common Metamodel for the Development of Web Applications. In J. M. C. Lovelle, B. M. G. Rodríguez, L. J. Aguilar, J. E. L. Gayo, and M. del Puerto Paule Ruiz, editors, *Proc. Int. Conf. Web Engineering (ICWE'03)*, volume 2722 of *Lect. Notes Comp. Sci.*, pages 495–506. Springer, Berlin, 2003.
11. P. Kruchten. *The Rational Unified Process — An Introduction*. Addison-Wesley, Reading, Mass., &c., 2nd edition, 2000.
12. P.-A. Muller, P. Studer, and J. Bézivin. Platform Independent Web Application Modeling. In P. Stevens, J. Whittle, and G. Booch, editors, *Proc. 6th Int. Conf. Unified Modeling Language (UML'03)*, volume 2863 of *Lect. Notes Comp. Sci.*, pages 220–233. Springer, Berlin, 2003.
13. Object Management Group. Unified Modeling Language Specification, Version 1.5. Specification, OMG, 2003. `http://www.omg.org/cgi-bin/doc?formal/03-03-01`.
14. G. Rossi and D. Schwabe. Object-Oriented Web Applications Modeling. In M. Rossi and K. Siau, editors, *Information Modeling in the New Millennium*, pages 463–484. IDEA Group, 2001.
15. D. Schwabe, editor. *Proc. 1st Int. Wsh. Web-Oriented Software Technology (IWWOST'01)*, 2001. `http://www.dsic.upv.es/~west2001/iwwost01/`.
16. `http://www.argouml.org`.
17. `http://gef.tigris.org`.
18. `http://nsuml.sourceforge.net`.