Proceedings of the 8^{th} International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010. Vol. 2. pp. 305–312.

Modeling Business Processes in Web Applications

Attila Adamkó, Lajos Kollár

Department of Information Technology, University of Debrecen
e-mail: adamkoa, kollarl@inf.unideb.hu

Abstract

Business process modeling in Web applications is an emerging field. The early methods of Web Engineering only dealt with the structural, navigational and presentational part of an application but business processes were completely missing. The demand for these processes resulted in several modeling techniques. One of them uses stereotyped class diagrams while others apply state machines for defining the transitions between the steps.

In this paper the pros and cons of each approach will be discussed, highlighting the differences and how can these methods work together in order to capture different aspects of business processes and how can they help us for effective and complex software development processes.

These processes and workflows could be expressed in a set of simple UML diagrams like activity and state machine diagrams. We will discuss the benefits and disadvantages of this model-driven software development approach.

Keywords: Web Application, Business Process, BPM, UML, Spring Web Flow

Categories and Subject Descriptors: D.2.10 [Software Engineering]: Design; D.2.11 [Software Engineering]: Software Architectures; H.4.3 [Information Systems Applications] Communications Applications

1. Introduction

Business Process Modeling (BPM) is a modern term and methodology which has evolved through different stages and names. Confusingly, the acronym BPM can mean different things, some of them are closely related to Business Process Modeling while others are not. 'Business Process Management' is an example of a different and related meaning. Business Process Modeling is a method for improving organisational efficiency and quality.

The increasing transparency and accountability of all organisations, including public service and government, together with the modern complexity, penetration and importance of ICT (information and communications technology), for even very small organisations nowadays, has tended to increased demand for process improvement everywhere. This means that Business Process Modelling is arguably more widely relevant than the earlier efficiency methodologies (e.g., Time and Motion Study or Total Quality Management were.

In common sense, Business Process Modeling aims to improve business performance by optimising the efficiency of connecting activities in the provision of a product or service. Business Process Modelling techniques are concerned with 'mapping' and 'workflow' to enable understanding, analysis and positive change. Diagrams – essentially 'flow diagrams' – are central features of these methodologies.

The term Business Process Model (also abbreviated to BPM) is the noun form of Business Process Modelling, and refers to a structural representation, description or diagram, which defines a specified flow of activities in a particular business or organisational unit.

A Business Process Model (BPM) is commonly a diagram representing a sequence of activities. It typically shows events, actions and links or connection points, in the sequence from end to end. Sequence is significant and essential to most aspects of business process modelling.

Over the years, the scope of business processes and BPM has broadened. Less than a decade ago, BPM, known then as "workflow", was a groupware technology that helped to manage and drive largely human-based, paper-driven processes within a corporate department. BPM today is an enterprise integration technology complementing Service-Oriented Architecture (SOA), Enterprise Application Integration (EAI), and Enterprise Service Bus (ESB).

However, in our point of view, in the field of Web applications the BPM is a supporting tool to describe the business logic inside the application. In order to express these rules we can apply several standards as it can be seen in Table 1.

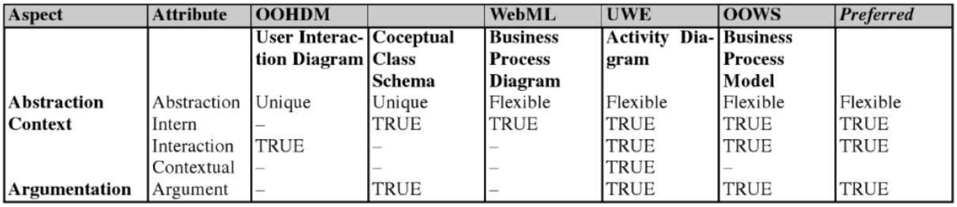
Standard	**Organization**	**Type**
Business Process Execution Language (BPEL)	OASIS	Execution Language
Business Process Modeling Notation (BPMN)	Business Process Management Initiative (BPMI)	Notation language
Business Process Modeling Language (BPML)	BPMI	Execution language
UML Activity Diagrams	OMG	Notation language
XML Process Definition Language (XPDL)	WfMC	Execution language

Business Process Definition Metamodel (BPDM)	OMG	Execution language and/or notation language, as MDA metamodel
Business Process Runtime Interface (BPRI)	OMG	Administration and monitoring, human interaction, system interaction, as MDA metamodel
Web Services Choreography Interface (WSCI)	World Wide Web Consortium (W3C)	Choreography
XLANG	Microsoft	Execution language
Web Services Flow	IBM	Execution language
Language (WSFL)		
Business Process Schema Specification (BPSS)	OASIS	Choreography (and collaboration)

Table 2: Common BPM Standards

At the heart of the architecture is a runtime engine that executes processes whose source code is written in a given language. What distinguishes between architectures is the choice of standards. In our case, to model business processes in Web applications there are several factors to take into account, like frameworks and methodologies, that can influence the development and design steps in these kinds of applications.

Based on a case study in [6], Table 2 shows that the most flexible methodology which supports business modelling for Web applications is the UML-based Web Engineering (UWE).



Aspect	**Attribute**	**OOHDM**		**WebML**	**UWE**	**OOWS**	***Preferred***
		User Interaction Diagram	**Coceptual Class Schema**	**Business Process Diagram**	**Activity Diagram**	**Business Process Model**	
Abstraction	Abstraction	Unique	Unique	Flexible	Flexible	Flexible	Flexible
Context	Intern	–	TRUE	TRUE	TRUE	TRUE	TRUE
	Interaction	TRUE	–	–	TRUE	TRUE	TRUE
	Contextual	–	–	–	TRUE	–	
Argumentation	Argument	–	TRUE	–	TRUE	TRUE	TRUE

Table 3: Comparison of several methodologies

The activity diagram provides good possibilities to model the flow of information inside Web applications. However, it does not contain or to be more precise, does not provide a way to define all different elements on the Web pages. But this is the closest method in the list which can fulfil our requirements.

On the other hand, we can think about a widely used framework's controller module to handle these requirements also. The Spring framework in version 2.5 has introduced a new controller module which uses a state machine diagram to capture business logic and processes in Web applications. In the following sections we would like to introduce these possibilities.

2. Modelling Business Processes

2.1. ArgoUWE

ArgoUWE is a CASE tool that 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.

As one can found in [1], the Process Model construction in the UWE design process includes a step where 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 classes 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 is automatically included in the process structure model. Auxiliary process classes can be added manually. These models can be seen in Figure 1 and Figure 2.

These diagrams are helpful to see the connections and relations between classes and business rules but it can be useful only if we choose to use the UWE methodology. Moreover, it has not got any code generations facilities to automatically generate a skeleton for the Web application.

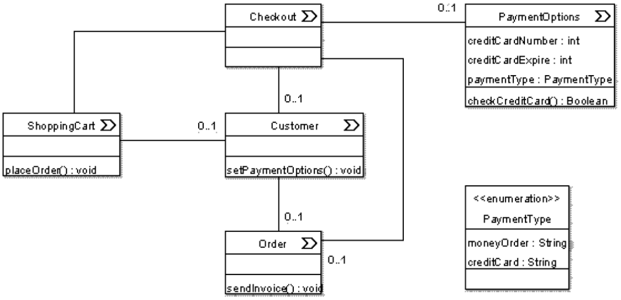


Figure 1: Process structure model

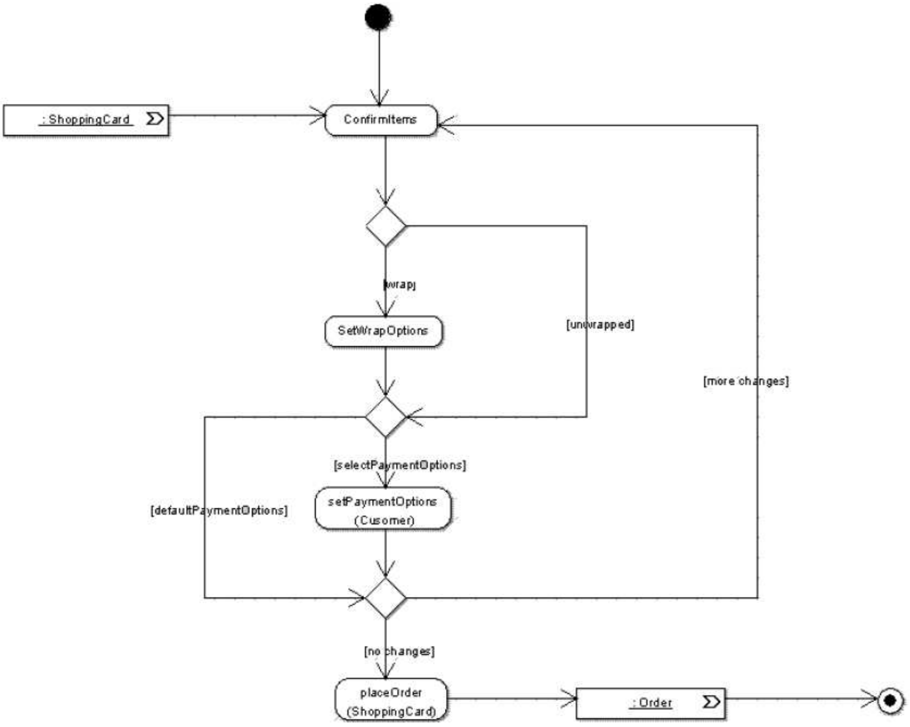


Figure 2: Process Flow model

2.2. Spring Web Flow

Another interesting direction can be found in the Spring framework. The basic idea behind Web Flow's method is the nature of the Web applications. More specially, how the users are navigating inside the application. Each task that a user can perform is expressed as a sequence of distinct steps, as a workflow. Moreover, to make a step there could be defined several conditions that must be met before the user can go to the next step – or as we will see, to the next page. In order to express these conditions UML's State Machine diagrams are applied because this diagram can be used to express all the necessary requirements need to be fulfilled.

In Spring Web Flow, a flow consists of a series of steps called "states". A flow encapsulates a reusable sequence of steps (states) that can execute in different contexts. A flow may call another flow as a subflow. The flow will wait until the subflow returns, then it responds to the outcome of the subflow. Flows are authored by Web application developers using a simple XML-based flow definition language.

Entering a state typically results in a view being displayed to the user. On that view, user events occur that are handled by the state. These events can trigger transitions to other states which result in view navigations. This can be seen in Figure 3 which is a typical booking sequence.

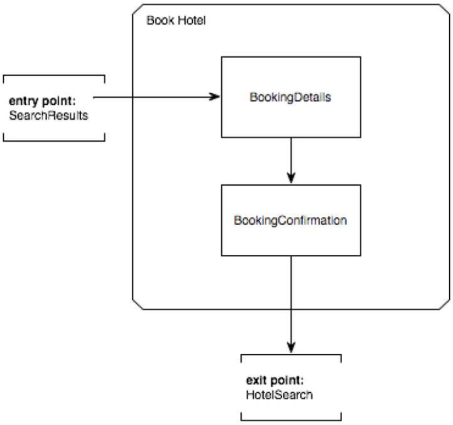


Figure 3: An example to show a flow (for booking a hotel)

Most flows need to express more than just view navigation logic. Typically they also need to invoke business services of the application or other actions. There are several points where actions can be executed, like flow start, state entry and exit, on view render and so on. Actions are defined using a concise expression language. Spring Web Flow uses the Unified EL by default. The example in Figure 4 shows the same flow as Figure 3 but expressed in the flow definition language.

```
<flow xmlns="http://www.springframework.org/schema/webflow"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/webflow
                          http://www.springframework.org/schema/webflow/spring-
webflow-2.0.xsd">
   <input name="hotelId" />

   <on-start>
       <evaluate expression="bookingService.createBooking(hotelId,
currentUser.name)"
                  result="flowScope.booking" />
   </on-start>
   <view-state id="enterBookingDetails">
       <transition on="submit" to="reviewBooking" />
   </view-state>

   <view-state id="reviewBooking">
       <transition on="confirm" to="bookingConfirmed" />
       <transition on="revise" to="enterBookingDetails" />
       <transition on="cancel" to="bookingCancelled" />
   </view-state>

   <end-state id="bookingConfirmed" />
   <end-state id="bookingCancelled" />
</flow>
```

Figure 4: A sample flow

The reason why we have chosen Web Flow as a good candidate to model business processes inside these applications is the possibility to model stateful Web applications resulting a process driven approach. One can define all navigation in a central location and can treat the entire interaction as one algorithm. It is possible to define the flow programmatically.

Web Flow uses a Domain Specific Language (DSL) analogous to a flowchart to model processes, as it can be seen in Figure 4. These processes (flows) are reusable across different areas of the application. Similarly to method invocations, flows can be called by other flows, can accept input parameters and can return output values.

These possibilities make the Sping framework's Web Flow controller module as a recommended solution for capturing the business processes for Web applications.

3. Conclusion

One of the greatest advantages of the Model Driven Architecture is that we can build complex systems with a clear, structured project environment. The project participants (analysts, designers, developers) work on separated areas. The design plans in an MDA environment are UML documents. These documents change along with the development process.

In this paper we have focused on the business logic and process modelling possibilities inside Web applications. We have discussed the pros and cons of two approaches highlighting the differences and how these methods can work together in order to capture different aspects of business processes. We have also shown the technologies we found effective for complex software development processes.

These processes and workflows can be expressed in a set of simple UML diagrams like activity and state machine diagrams. We showed that UWE uses class diagrams and activity diagrams to capture business requirements while Web Flow applies state machine diagrams for the same purpose. Both of these approaches have advantages and disadvantages.

Activity diagrams can be exposed as typical business processes mapped to the Web environment. Developers grown up on UML can find this approach very useful and easy to understand. However, Web applications have a different working mechanism but if we apply the principles of UWE we can create these models and utilize its benefits.

On the other hand, Sping's Web Flow concept uses state machine diagrams, or more precisely, a DSL which is based on state machine diagrams. This approach deals with states and each flow can be expressed as a sequence of Web pages. From a practical viewpoint, it is easier to imagine a Web application this way.

Modifications can easily be introduced into the models in both cases. However, it is more comfortable to update only one model rather than two. It can lower the possibility of inconsistency inside the design plan. These approaches are usable for agile projects with high change request expectations.

This architecture is not recommended for small applications. The setup of the configuration and the whole generation process would only mean an overhead in the cost during the development. Finally, we found it important to underline that this software development method is effective in medium- and greater sized projects only.

Acknowledgement. This work is supported by TÁMOP 4.2.1./B-09/1/KONV-2010-0007/IK/IT project. The project is implemented through the New Hungary Development Plan co-financed by the European Social Fund, and the European Regional Development Fund.

References

[1] Alexander Knapp, Nora Koch, Gefei Zhang, Hanns-Martin Hassler: Modeling Business Processes in Web Applications with ArgoUWE. UML 2004: 69-83, 2004

[2] Schmidt, D.C., "Model-Driven Engineering." IEEE Computer 39 (2), 2006, pp. 25-31.

[3] Mendes, E., Mosley, N., "Web Engineering." Germany : Springer, 2006.

[4] Conallen, J., Building Web Applications with UML. 2nd Edition. Boston : Addision Wesley, 2002.

[5] Jan Machacek et al., Pro Spring 2.5, Apress, 2008

[6] Jurriaan Souer et al., Model-Driven Web Engineering for the Automated Configuration of Web Content Management Systems, 9th International Conference, ICWE 2009 San Sebastián, Spain, June 24-26 2009 Proceedings, Springer, 2009

Attila Adamkó, Lajos Kollár
Department of Information Technology
University of Debrecen
H-4010, P.O. Box 12, Debrecen
Hungary