

# Combining Agile Practices with UML and EJB: a Case Study in Agile Development

**Richard Paige**

Department of Computer Science  
University of York  
Heslington, York  
YO10 5DD, United Kingdom  
+44 1904 433242  
paige@cs.york.ac.uk

**Priyanka Agarwal**

Department of Computer Science  
University of York  
Heslington, York  
YO10 5DD, United Kingdom  
priag16@rediffmail.com

**Phillip Brooke**

School of Computing  
University of Plymouth  
Drake Circus  
Plymouth, Devon PL4 8AA  
+44 1752 232712  
philb@soc.plym.ac.uk

## ABSTRACT

An agile methodology that integrates selected XP practices, UML modeling and Enterprise Java Beans is described. A case study in the domain of web-based systems is outlined, which applies and assesses the utility of the methodology. The applicability of agile modeling to the domain of web-based e-commerce systems implemented using Enterprise Java Beans is discussed.

## Keywords

agile modeling, XP, UML, EJB, case study

## 1 INTRODUCTION

Agile methodologies [2] and processes have been developed to support rapid development of high-quality code. Such methodologies follow a number of principles, such as short iterations between deliverables, minimal modeling, test-first development, and simplicity in terms of models constructed, code written, and processes followed. Agile methodologies are well suited to domains where system requirements are changing rapidly, where client feedback is feasible to obtain, and where the priority is obtaining a working system as soon as possible.

This paper reports on a case study in developing and applying an agile methodology for building web-based e-commerce systems. The methodology is novel for two main reasons: it integrates several of the key principles and practices of Extreme Programming (XP) [4] with UML modeling, attempting to deal with the incompatibilities between these techniques in the process. As well, it targets code written in the Enterprise Java Beans (EJB) [11] framework, which is particularly suited for building web-based systems, but which has yet to be fully investigated in

terms of its compatibility with agile development. The goal of the case study is to assess the practicality and effectiveness of using both UML and EJB together in an agile manner, given a project that must be completed under strict time constraints. We claim that the technologies are compatible and can be highly effective when used together to build web-based systems under substantial time constraints.

The paper provides an overview of the methodology and some details of its application to the development of a web-based e-commerce system. Full case study details of the iterations carried out can be found in [1].

## 2 AGILE METHODOLOGIES

Agile methodologies aim at establishing strong communication links between developers and customers, and accept volatility in requirements. They focus on rapid delivery of high-quality software. Examples of agile methodologies include XP-based approaches, the Crystal family [8], and Feature-Driven Development [6]. These methodologies all differ in many, but they all aim to follow a number of principles and practices, such as:

- *early* and *continuous delivery* of working software via iteration and incremental change.
- *simplicity*, i.e., maximizing the amount of essential work that is carried out.
- *minimizing* the deliverables, e.g., modeling artifacts and documentation.
- *involve the customer*
- *testing* and *validation* throughout the development lifecycle.

Different agile methodologies emphasize these (and other) practices in different ways, depending on application domain, project-specific and management constraints, etc. The methodology that we have devised integrates some of these practices with techniques from UML modeling and

EJB development. Compatibility of the techniques and practices is, of course, an issue that must be addressed.

### 3 OVERVIEW OF THE METHODOLOGY

A motivation for creating the methodology was the need to construct a small web-based e-commerce system. Rapid development was essential, given specific and fixed time constraints of less than a few months for complete development. The need for careful, yet lightweight modeling for *requirements understanding* was also identified. At the same time, it was desired to assess the utility of EJB in development. The methodology aims to allow developers to use UML modeling and EJB together in a lightweight manner. These technologies are therefore integrated with XP practices.

EJB was selected because it is, in effect, a framework for distributed systems development. As such, it is useful for rapidly constructing systems in this domain: developers typically need implement only business logic in the EJB framework in order to construct a system. This is entirely compatible with agile practices, and is a key reason for selecting the technology.

It was desired to use UML modeling to carry out initial planning and requirements analysis – particularly, for dealing with vague customer goals, wants, and needs – and to promote understandability. We argue that it is difficult to commence development by writing code directly from informally stated user requirements. A characteristic of Enterprise Java Beans systems is that the applications are often difficult to explain to customers because of the complex interactions between components (e.g., session and entity beans) that are usually hidden within the EJB infrastructure. Thus, UML modeling was also chosen to help with explaining the system under development to customers. Particularly for EJB-based development, it is useful to construct simple models to help abstract away details and to help explain the system. At the same time, it was desired to avoid use of excessive modeling, and to carry out any modeling in a lightweight fashion.

The UML elements applied in the methodology are use case diagrams, class diagrams, and collaboration diagrams. Use case diagrams are applied minimally; it is the use case scenarios and templates that are more important for understanding and explaining requirements. The methodology itself is use case driven: an identified use case is developed through to implementation and testing in EJBs. Class diagrams are produced after use case modeling, and are used for specification, rather than conceptual, modeling. Since the resulting implementation will be in EJB, the classes in the class diagram will be identified as EJB session and entity beans. Collaboration diagrams are used when trying to understand complex business logic that must be implemented in the individual beans.

### Testing

A hallmark of XP approaches is their emphasis on test-based development; this is particularly valuable for building systems with substantial reliability requirements. In the methodology, we develop acceptance tests from the use cases. This means that tests act as *specifications* for the methods that must be implemented in Beans. We write unit tests for every method. We then fill in the logic of the test case with the implementation details as we would for that method. For validation, the methodology uses JUnit [9] and Java's `Assert` class. This class helps to validate the output of every method against the expected output. Only when this passes, do we move on to work on the next method. Once all the unit tests have passed, acceptance tests can be executed.

Testing is integrated with *both* EJB coding and with modeling. Developers may wish to carry out some modeling when working out the internal logic of methods or entire beans; to this end, UML collaboration diagrams might be applied. These diagrams can then be used to further generate tests, e.g., following the work of Briand [5] and Paige & Ostroff [10]. Ongoing work is focusing on tool support to automatically generate test cases and test plans from UML diagrams.

### The iterative nature of the methodology

The methodology by design is iterative; an activity diagram depicting the process can be found in [1]. The second iteration starts with the need to add new functionality. The system design at the end of iteration 1 becomes the initial design to iteration 2, which introduces new classes and interactions leading to system design 2. In addition there is flexibility between coding and design within an iteration itself. This flexibility is supported via tools that provide forward and reverse engineering capabilities.

The methodology views models as deliverables in order to help explain the EJB implementation to customers. As such, consistency between implementation and models is important. CASE tools can be used to help establish consistency, by forward and reverse engineering.

### 4 THE CASE STUDY

The case study that was used to experiment with and assess the methodology involved the construction of an Internet call centre. The system aimed to provide facilities for retrieving quantitative statistical data for a call management system, and to support customers' requests for information, via a web page. The system to be developed maintained a database of various IT, Manufacturing, Banking, and Accounting organisations. It would offer the following features; exact requirements for each system service were highly changeable.

- **Search:** users could search the call centre for organisation details by a variety of criteria, such as name, category, location, etc. The search would result in contact details, where available.

- **Statistics:** users could view statistical data by city, category and organisation name.
- **User authentication:** Users would have to be registered in order to gain access to the facilities. They would enter their details and thereby use email and password as login parameters.

#### **Suitability of the system for agile development and EJB**

The system was an e-commerce application to be used in a domain with changing functional requirements. This makes it poorly suited for traditional development methods that consider a set of stable requirements with a definite plan. In order to facilitate making changes and incorporate new features, we need to follow an iterative approach that considers one piece of functionality at a time rather than build the whole system at once. This makes an agile approach suitable. The strict time constraints for completion also made an agile approach suitable.

The call centre itself was a distributed web based system where the clients and the database were located remotely on a network. This is an event-driven system, where processing is triggered by the client. Technology options for this system include pure Java as well as component-based middleware technologies, such as CORBA and EJB. These alternatives are analyzed in detail in [1]. EJB was chosen in part because of the framework it provides, as discussed earlier.

#### **Details of the case study**

It is not possible to include full details of the case study so herein we provide a summary. The first stage was to decide on an architecture. For such an event-driven system, the model-view-controller architecture was selected. Work proceeded by constructing a use case diagram and carrying out release and iteration planning, prioritising the use cases based on risk, priority, and criticality. The *Search* use case was initially chosen for development through to implementation. At the same time, acceptance tests for this interaction were constructed from the use cases.

The next step was to follow through with development of the *Search* use case. For this, UML class diagrams were created. At the same time, EJB beans were determined to avoid later refactoring. Classes were matched with beans. Thus, a number of classes, such as session beans, entity beans, and UI classes, were determined and detailed informally first using CRC cards, and then more formally using class diagrams. Only minimal modeling was carried out at this stage, in part because a layered architecture had already been selected, EJBs had been chosen as an implementation technology, and only informal details were needed at this stage.

Collaboration diagrams were constructed next to depict internal business logic of use cases and methods and to

show connections between classes. Modeling followed the AM principle of *modeling with purpose*: incomplete collaboration diagrams were constructed, sufficient to continue EJB development. The collaboration diagrams helped us to expand the search use case in terms of messages. This proved useful in understanding the internal logic of search. However we expanded only the normal flow of events; the alternate flow of events were only a slight deviation, which is typical with this kind of application; thus we did not feel the need to do unnecessary modeling.

Testing was the next stage to be carried out. Data was generated for populating the client database, and then JUnit was used for functional testing. After the code passed the tests, stress testing was carried out. Testing Enterprise JavaBeans is not straightforward since it involves traversing multiple layers in the software architecture. Thus, simulation of clients is used in, e.g., testing entity beans. Code for these tests and for the Beans themselves can be found in [1].

We found that EJB was particularly well suited for this development since it meant that we did not have to write many methods in the implementation – the EJB framework provided them for us. This promoted rapid development and also helped to minimise the overall amount of modeling that we needed to carry out.

The results of this iteration were then delivered to the customer, who offered feedback and constructive criticism. The second iteration then could begin, focusing on the development of the remaining use cases. These proceeded along the lines described above. Full details can be found in [1].

#### **5 LESSONS LEARNED AND CONCLUSIONS**

A number of issues are worth discussing in the context of the web-based case study and the agile methodology.

- *Was agility achieved?* Modeling was carried out before coding, since development could not start without understanding the use cases. Minimal modeling was effected in developing business logic for the use cases, and in modeling system architecture. In part, the latter effect was due to the use of EJB as an implementation technology, since it imposes a strict architecture on implementations. Modeling was primarily helpful for deriving use cases that helped us understand customer requirements: these models drove development, and were particularly useful in building test cases and in test planning and resource allocation. Agility was therefore achieved at the level of use case modeling and building test cases. We also achieved agility in the use of class diagrams: simple class diagrams were constructed and mapped immediately to EJB code; the class

diagrams were thereafter useful in explaining the system.

- *Success of the methodology:* the methodology aimed at integrating XP practices, UML modeling, and EJB. The XP practices that it applied included customer involvement, emphasis on testing, simplicity, and short iterations. XP's emphasis on testing was the most useful element for this case study. As well, the emphasis on rapid development was appropriate, since the case study was completed under substantial time constraints. In carrying out the case study, the primary goal was to assess the utility of the methodology, particularly how the three technologies fit together; thus the suitability of the case study problem domain to agile development was not considered rigorously. Success can only be determined by carrying out numerous different case studies and by quantitative measurement. The case study that we carried out was successful, in the sense that the system was constructed within deadlines and satisfied its key requirements, but further case studies must be performed. We will attempt to measure success by measuring time-to-delivery, the number of defects removed from code over time, amongst other measures.
- *Lessons learned.* Modeling is essential, even in XP-based developments. It was essential in this development for capturing use cases, object interactions, and simple associations among

classes. These models were thereafter useful in explaining the system to the customer. EJB was particularly useful in this domain for rapid application development. The framework that it provides was useful in dealing with several iterations of development, since the overall architecture of the system to be developed was not about to change. As well, the fact that EJB effectively provides a framework for development helped in producing a working system more quickly.

Ongoing work includes further case studies as mentioned, but with making greater use of CASE tools, particularly Together's ControlCentre. We are also exploring different uses of UML in the methodology, particularly the use of class diagrams for conceptual modeling. Also, the methodology places a requirement for models and code to remain consistent, in order to use models when speaking with customers. It would be ideal to be able to use forward and reverse engineering techniques, as supplied by ControlCentre, to keep code and models consistent. Due to time constraints and the in-flux nature of EJB support in existing CASE tools, this was not explored in the case study, but we aim at considering it in detail in the future.

## REFERENCES

1. Agarwal, P. A case study in agile development using Enterprise JavaBeans and the Unified Modeling Language, M.Sc Thesis, University of York, U.K., September 2002.
2. Ambler, S. The official Agile Modeling Site. <http://www.agilemodeling.com>. Visited October 18, 2002.
3. Aoyama, M. Web-based agile software development. *IEEE Software*, November/December 2000.
4. Beck, K. *Extreme Programming Explained*, Addison-Wesley, 1999.
5. Briand, L. and Labiche, Y. A UML-Based Approach to System Testing. In *Proc. UML 2001*, LNCS 2185, Springer-Verlag, 2001.
6. Coad, P., and Palmer, S. *Feature-Driven Development Guide*. <http://thecoadletter.com/download/fddguide>. Visited October 30, 2002.
7. Cockburn, A. *Agile Software Development*, Addison-Wesley, 2002.
8. Cockburn, A., and Highsmith, J. Crystal Methodologies. <http://www.crystalmethodologies.org>. Visited October 28, 2002.
9. JUnit Testing Framework. On-line at <http://www.junit.org>. Visited October 25, 2002.
10. Paige, R.F. and Ostroff, J.S. A Proposal for a Lightweight Rigorous UML-Based Development Method for Reliable Systems. In *Proc. pUML Workshop*, GI-Series 7, Lecture Notes in Informatics, 2001.
11. Roman, E., Ambler, S., and Jewell, T. *Mastering Enterprise Java Beans*, Wiley, 2002.
12. Turk, D., France, R., and Rumpe, B. Limitations of agile software processes. In *Proc. XP 2002*, Alghero, Italy, 2000.