Blekinge Institute of Technology
Computer Science 41-60 points
Spring 2005

2005-05-31

A comparison of UML and WAE-UML for the design of Web applications

Authors:
Heinz Andersson
Mikael Gustavsson

Supervisor:
Peng Zhang

Abstract

Since Web applications are very complex, compared to traditional client/server applications, Web application design with the UML can be obtrusively hard for a modeller. The grounds are that the UML does not define the correct semantics to be able to visualize a web application correctly. This is a qualitative reduction study where we have used interviews and our own experience during the redesign of a UML-modelled e-commerce application with WAE-UML. Using the flow of a case study we have tried to see if we can improve three quality attributes of a complete design. *Stakeholder communication* reflects the need of unambiguous design artefacts that are easy to understand and that mediate the real message of the use-case. The *condition* of the design artefacts should provide artefacts that resemble reality and that not are misleading and provide for verification and validation of the requirements. The last attribute *maintainability* should provide means for easy maintenance and updates. We found that WAE-UML can improve these quality attributes in a design but the impact it has on them is dependent on two major aspects. The first aspect concerns the designers' judgment of detail in a design. A detailed design can be good considering requirements and use-case traceability and verification, but prohibit communication. Maintainability can also be improved in a detailed design because the diagrams are less abstract and a truer picture of the application. The second aspect is that it depends on the knowledge possessed of the semantics by the people in contact with the design documents. Due to the time aspect the people working in the industry that we interviewed were reluctant to modelling a Web application at all. They thought it would take a long time to learn WAE-UML but also for executing a design phase.

Keywords

Web application, Design, UML, WAE-UML, Maintainability, Stakeholder communication, Condition.

Acknowledgement

Hereby we want to forward our gratitude to our supervisor for positive feedback and for meeting our questions in an informative way. We also would like to show our gratitude to the interviewees that gave of there valuable time and opinions. Finally, we send our thanks to all other people that have been a support during the execution of this thesis.

Table of contents

1 Introduction

In this chapter, we present the background problem and hypothesis for executing the thesis along with our background in the area of Web application development.

1.1 Our background

We are students at Blekinge Institute of Technology from the field of computer science and software engineering. The work in this thesis is based on questions which aroused during project courses we attended within our programmes.

The theoretical background and our previous knowledge are based on the pre-existing knowledge of the UML (Unified Modelling Language) as taught in the course object-oriented systems development. We do not claim to be experts in the UML neither in Web application development. With this experience and these non-expert aspects in mind we set the ground for approaching the thesis.

1.2 Problem description

Involvement in past web application development has risen a few questions about how suitable UML is for the modelling of Web applications. A web application is more complex to model than a traditional desktop application. It is more complex because it includes many elements and a mixture of different programming techniques. It is very hard, if not impossible, to use the standard UML for modelling a Web application. Many researchers agree on this, for example Conallen (2002), Baresi, Garzotto & Paolini (2001) and Ceri, Fraternelli & Bongio (2000).

Since a Web application is composed of highly distributed logic content and the fact that it is not only built with objects but also HTML (Hypertext Markup Language) web pages, scripts, hyperlinks and forms, a Web application cannot in an easy way be modelled with standard UML. We think that the cause of this is that the standard UML, developed by OMG (Object Management Group), is developed with object-orientation in mind and the fact that a Web page is not object-oriented in nature, which Conallen (1998, Conclusion section) also agrees on. Some will see a Web page as an object in compliance with the DOM (Document Object Model) but we still can not easily model a document based on the DOM with the UML. To put it simply, the UML does not define enough or

the correct semantics to describe the complexity of a Web application in a design artefact. Conallen approves to this statement.

I found both OMT and UML inadequate to express the things I thought were important in a Web application. (Conallen, 2002, Preface xvi)

The difficulties during previous design efforts, using standard UML, have circulated in five major problem areas.

Ambiguous design and abstract design - Since the UML does not define the correct semantics for modelling Web applications, efforts on modelling has led to that the designers used ad-hoc solutions for modelling, maybe not using the correct semantics to describe the complex behaviour. The designers have to describe what they describe in the model with additional text documents which means a lot more work. The design will be ambiguous and abstract to the readers of it and require a lot of effort to understand.

Complex to perform analysis and design - Due to the ambiguousness and abstractness it is complex for the engineering team to perform the analysis and design phases because it is hard to express in the design the message they want to mediate.

Not confirmed requirements and use-case - In an abstract design the requirements and use-cases might not be fulfilled by the design artefacts. Even if the use-cases and requirements should be fulfilled, it is not easy to trace them in the design artefacts and the developers have to make guesses and use the requirements specification and use-cases as an aid during the implementation phase.

Difficulties to communicate among the stakeholders - Different stakeholders might not understand an abstract design and think that it does different things instead of the intended tasks. It is not only important to communicate the design to customers and executives but also to communicate it to programmers and other people in the development team. An abstract design does not mediate between stakeholders and requires additional documents, such as requirements specification and use-cases, and other descriptions of the system. This requires more interaction, for a programmer, with the documentation because the programmer need to interpret a lot of text and descriptions and therefore takes more time for the programmer to understand the design. This effects the communication between the designer and the programmer. The stakeholders

that are interested in reading the design documents are customers, project team members, executives and the people who will maintain the system post-delivery. These persons all have different knowledge of UML semantics and therefore might understand a design in different ways.

Maintaining the design artefacts - If the design artefacts are complicated and abstract the implementing people are less willing to follow the design and not likely to do adequate and necessary updates to them. This leads to that the finished product does not map to the design artefacts. In a later stage when the system goes through maintenance it will be a complex task. The maintenance staff might not be the same persons as the developers implementing it and as such are dependent on good and descriptive design artefacts for easy maintenance.

From these five problem areas we can deduct three attributes defining good design artefacts according to our experience. The condition of the design artefacts should provide verification and validation of the use-cases and requirements it reflects. The condition of the design artefacts should also provide artefacts that resemble reality and that are not misleading. In the aspect of communication among stakeholders we need unambiguous design artefacts that are easy to understand and that mediate the real message of the use-case. The maintainability aspect should provide means for easy maintenance and updates.

1.3 Hypothesis

Since our background is working with UML and development of Web applications we wanted to investigate as to what extent the UML was at all suitable for modelling Web applications. Experience in past projects also raised questions about what methods commonly are used in the industry.

The choice of using WAE-UML(Web Application Extension to UML) as an aid in modelling Web applications made us formulate further questions regarding if, and to what extent, this method can help us improve our pre-existing and future designs. As the problem description suggests, the task was to examine how the new design choice could help us in producing a better design improving the quality attributes; and as such aid in the communication with stakeholders and improve the condition and maintainability of the design artefacts. These questions define the framework for the thesis and have since the start evolved to conclude the following hypothesis:

Using WAE-UML for Web application design improves the condition and maintainability of the design artefacts and the communication among the stakeholders.

1.4 Goals and motivations

We have several goals with the thesis. The first goal is to bring light to the possibilities to use previous knowledge of UML and apply it on the modelling of Web applications. Since most designers and Web application developers have previous knowledge of UML; using UML instead of a new way of modelling (such as W2000 and WebML) is according to us the best way to attend than to learn a new way of modelling. We noticed during the execution of the thesis that the developers in the industry are not using any particular development method or design methods at all. They are dependent on descriptive documents such as specifications and requirements. We are under the opinion that other methods than UML, like WAE-UML, needs to be more widespread in the industry. They can provide a more genuine way of development and reduce the development costs. Conallen has something to say in this matter also.

Current development environments make it so easy to produce simple web applications that they have the unfortunate side effect of encouraging us to develop and evolve applications in the absence of serious analysis and design. Any system with non-trivial complexity needs to be designed and modelled. (Conallen, 1998, Introduction section.)

The second goal is to be able to create better design artefacts that are unambiguous and clear and which correctly describes the Web application under development. Most importantly the design artefacts should confirm the requirements and use-cases, so that a meaningful verification and validation can be performed.

As a consequence of the two first goals we hope to be able to develop more Web applications instead of the more traditional client/server applications. Web applications are architecturally a specialization of the client/server architecture at a high level (Conallen, 2002, p. 141). The differences appear at the lower levels and typically are related to the communication protocol, HTTP, and vendor specific extensions of the client browser (Conallen, 1998, Web sites). In this context we define the difference as when developing a client/server application we need to develop software executables for both the client and the server side.

In a client/server application these executables does not include java applets, or similar technologies, meant to execute in the browser. In a Web application the Web browser acts as the client side program and as such we need only develop software for the server side, and sometimes extensions for the browser. We can use the communication protocols already defined. This means that if we have a good way to model a Web application we preferably will choose to develop an application as a Web application instead of a client/server application. The arguments for the development of Web applications instead of client/server applications are that a Web application provides us with many parts of the system. For instance, to our opinion, it can reduce the development cost and effort because the client side uses only a browser and the protocols are already there for the communication. On the server side we have http servers, application servers and DBMS' (Database Management System) already implemented and tested. The focus of the development can then be concentrated on trimming business logic and improving the presentation to the user.

We think that bringing to light the possibilities of using WAE-UML for the modelling of Web applications we can help future Web applications projects to design and implement better software.

1.5 Delimitation

Due to the complexity of the problem we have chosen to restrictedly look at the stakeholder communication and the condition and maintainability of the design artefacts. We think that these are the most important aspects of a software design. The quality and performance of the resulting software product is certainly valuable to evaluate. There is no existing formal method, to our knowledge, for testing how the finished system performs given the design artefacts for it. The focus will not be on performance and quality of the resulting software product. However, as performance and quality of the system is an important aspect this will be mentioned briefly if it adds any value to the rest of the discussion.

2 Theoretic background

In this chapter we will start with a definition of what we mean with a Web application. The concept of Web application architecture and the .NET architecture are important to grasp to understand if a design artefact resembles reality. Thereafter we will describe the different technologies we have looked at

for the modelling of a Web application and why or why not they are suitable for this kind of modelling.

The literature regarding modelling of Web applications is not extensive. However, the efforts for developing a new way to model a Web application is still an evolving area of interest in the industry, and as such there exists three interesting methodologies namely, W2000 (Baresi, Garzotto & Paolini 2001), WebML (Ceri, Fraternelli & Bongio 2000) and WAE-UML (Conallen, 2002).

A fourth possibility is to use the standard UML for the modelling of Web applications. However, as described in the problem description this way is not a good way because the UML lacks the correct semantics and elements to model the complexity of a Web application correctly and accurate.

To our opinion, a method for the design of web applications should prove to have four properties:

P1. **Build on the UML**. It should incorporate or build upon the UML. In the industry, we believe that most developers and designers are familiar with the UML since it is de facto standard for modelling of traditional applications. Providing them with a new method decreases the learning efforts if it builds on the learners pre-existing knowledge.

P2. **Formal reference specification.** In a development team and among the stakeholders, not all are required to have knowledge of the UML. If the method provides a formal reference specification it should reflect the ability to understand and read the design artefacts. It is also easier for the designers and implementers to reference a specification.

P3. **Reveal the architecture.** If the method provides means to reveal the architecture, the understanding of the concept and domain should increase in the project team and among the stakeholders.

P4. **Increase the condition.** Increase the condition of the design artefacts means that they should describe the application in such a way that it can not be misleading; provided that the reader of the document possess adequate knowledge of the UML and knowledge of the formal specification for the method.

These properties are based on the qualities we want a method for modelling Web applications to provide.

2.1 Introduction to Web applications

We have adopted Conallens definition (2002, p. 31) where we distinguish a Web application from a Web site through the users influence on the business logic. The user executes business logic with the browser through the input and navigation he does on the physical web page that is rendered by his browser. The navigation and input of information on the client side browser is typically handled and transported to the server side through the definition of a posted form. This communication normally takes place through the use of the standard connectionless HTTP (Hypertext Transport Protocol) protocol. Due to this connectionless property, the client state management on the server is a challenge of Web applications since each client request to the server opens a completely new connection each time (Conallen, 2002). This is usually solved through session management or through the use of cookies. A web site contains static contents that the user can not change or interact with. However the methods for modelling Web applications discussed later can be applied on both smaller Web sites with static content and Web applications with dynamic content.

2.1.1 Architectures

The typical architecture of a Web application consists of the client browser, a web server, an application server and a DBMS. The role of the client browser is to render the html page that is sent from the server and to handle the communication with the Web server.

Typically an application server executes the code in the server side pages that affects the business logic and is usually located on the same machine as the server (Conallen, 2002, p. 147). It is also used for executing already compiled modules and classes which affects the state of the business content, such as the communication with a database server.

2.1.2 .NET concept

The Microsoft .NET architecture does not define a single responsible web page; rather a web page is composed of the virtual web page (.html) and a corresponding physical server page (.aspx) which has a super class (the code-

behind) that contains the majority of event handling mechanisms. Figure one illustrates this concept where the server page (.aspx) inherits the event handling and other methods from the code-behind super class (.cs) and builds the web page code that is eventually sent to the client (Conallen, 2002, p. 248). Even if this architecture is correct from a developer perspective it is not the same view as a user has. Figure one shows the .NET architecture from a designer and programmer perspective not from a user perspective. Since a user is unaware of the .html and sees only the .aspx extension in the browser.

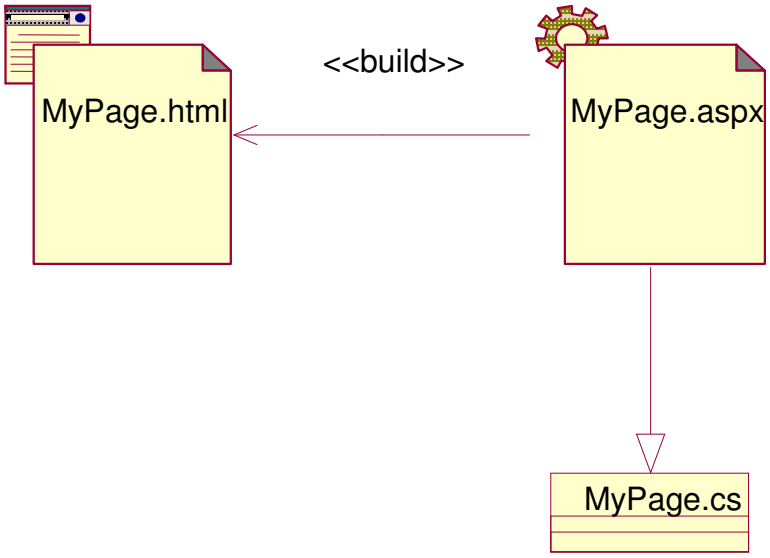


Figure 1: .NET Architecture (Conallen, 2002)

2.2 Why we model software?

We model software because it is not possible for a human being to keep every detail in mind in a complex model. We try to understand the complexity of a system by modelling it. In that way we can focus on the important tasks and not keep every detail in mind (OMG, 2001, p. 1-3). We use the model to communicate what to build with the stakeholders, to communicate with the people that are building it, and to communicate with the people who will maintain the system (Conallen, 2002, p. 4). With this in mind we understand that a design model must possess certain properties to be useful. With a model we can abstract the details so it is easier to understand. In the end it is up to the modeller to decide in what way the model is seen as enough. A model can neither be too abstract nor show too much detail. The role of the modeller is to abstract the model enough to

capture the meaningful parts of a system and make it easy for the stakeholders to understand it (Staroń, 2003, p. 3).

2.3 Standard UML

The UML was founded by Grady Booch, Jim Rumbaugh and Ivar Jacobson as a way to utilize and standardize the many different modelling languages that existed in the past. They emerged the UML from Booch, OMT (Object Modeling Technique) and OOSE (Object Oriented Software Engineering) modelling languages. The UML is standardized by OMG (OMG, 2001, p- 1-6).

Even though we think that the UML lacks the semantics for modelling Web applications it is still an option to use. The UML is now in the version 2.0; however, for the modelling of the diagrams that are used as a fundament of this thesis the version 1.5 has been used and accordingly described here.

2.3.1 Description

The UML defines a unification of semantics and a common notation to be able to visualize and specify complex software systems. It provides the modeller with a standard way and a formal description of how to document the evolution of any piece of software (OMG, 2001, p. 1-1).

With UML many things can be expressed but the common use is for designing software systems. It defines a Meta model and a notation for object-oriented modelling and documentation of artefacts for software systems. These artefacts are then mapped to object oriented languages.

The UML does not promote any special development process, however, the authors of UML address that object oriented modelling with UML should be performed in the context of a process, such as the UP (Unified Process), and which is use-case driven, architecture centric and either iterative or incremental (OMG, 2001, p. 1-11).

The UML should be able to be used in many areas such as embedded systems, desktop application development and Web application development. To address this universal behaviour UML defines and extensibility mechanism which includes stereotypes, tagged values and constraints (OMG, 2001, p. 1-9).

2.3.2 UML notation

As a complete UML notation reference is beyond the scope of this thesis this chapter provides only an introduction to the main concepts, the diagrams and the notation that we think is necessary to follow the discussions in the thesis.

The vocabulary of the UML can be classified into three components, namely elements, relations and diagrams. The elements are the most important and can for example be classes and packages. Relations are used to bind elements together and for executing the functional requirements. Relations are dependencies, associations, generalizations and realizations. In advanced modelling these relations can further be extended with aggregation and navigation. Diagrams are used for the visualization of the structures being modelled (Strand, 2003, p. 15).

The UML specifies semantic rules for how the elements and relations can be connected and how they are specified. The meaning of these rules is to provide a uniform understanding and to provide support for generating code from the model by CASE tools. A CASE tool is able to automatically generate stub code from a design.

2.3.3 UML extendibility

The UML defines tools for extending its semantics and the elements used for modelling any type of application. These tools are defined as stereotypes, constraints and tagged values (OMG, 2001, p. 1-9).

The stereotype is, according to us, the most important tool. A thorough examination of stereotypes and its use have been researched by Staroń (2003). Stereotypes can be used to change the vocabulary of UML and change the semantics of the notation so that it fits any need (OMG, 2001, p. 1-10). For example, the meaning of an association dependency can be changed so that the meaning of it is completely different and fits the modellers' needs. Constraints describe semantic restrictions to an element in UML. Enclosed by brackets, sentences can introduce restrictions for an element so that something will happen only if the restriction clause is validated true (OMG, 2001, p. 2-64). Tagged values are used to create a new definition of an elements specification (OMG, 2001, p. 2-29).

2.3.4 Web application modelling with UML

Because the UML is uniform, provides an extensibility mechanism and is well known in the industry then using UML in some way for the modelling of Web applications is preferred. It adheres only to properties P1 (Build on the UML) and P2 (Formal reference specification). Since the UML provides an extensibility mechanism for modelling any application it might adhere to property P3 (Reveal the architecture) but this means that a formal method and description must be developed by the designers for the communication aspects. From experience we can deduce that the property P4 (Increase the condition) is not fulfilled.

2.4 W2000

Baresi, Garzotto & Paolini (2001) describes a framework consisting of a combination of UML and HDM (Hypermedia Design Model) for defining Web applications. The framework defines methods for requirements analysis, state evolution design, hypermedia design, functional design and visibility design which together defines the full model of the Web application.

The W2000 framework can be used to build personalization into a model, which is how different users use and sees the Web application and to what extent they are allowed to use its functionality. It also focuses on content management, information design, and navigation design (Baresi, Garzotto & Paolini, 2001, p. 1).

2.4.1 Web application modelling with W2000

Even though this method provides us with some interesting properties such as personalization we think that it is not suitable for modelling large Web applications because it does not define any formal description (it seems that the method and promoted CASE tools are not actively maintained at the date of writing). To facilitate communication in a model there must exist a formal description so that different stakeholders can learn it and adapt the message that a model provides. The example models described by Baresi, Garzotto & Paolini (2001) also does not seem to reveal the architecture which is something that should be of large concern as an aid in communicating the message of the model.

2.5 WebML

WebML, as promoted by Ceri, Fraternelli & Bongio, uses orthogonal dimensions for specifying web applications at the conceptual level and addresses platform independent specification of data-intensive Web applications (Ceri, Fraternelli & Bongio 2000, p. 138). WebML provides one-to-one personalization of content addressing the users and also targets the delivery of content to multiple devices, such as handheld and mobile devices (Ceri, Fraternelli & Bongio 2000, p. 138).

The orthogonal dimensions constitute a structural model which describes the data content in web pages and the relationships among the content. The pages that compose the Web applications are modelled in a composition model. Link topology or navigation among the pages in the Web application is modelled in the navigational model. When it comes to the layout of the pages and the graphical requirements for the Web application these are modelled in a presentation model (Ceri, Fraternelli & Bongio 2000, fr. p. 138).

A Web application design modelled with WebML is, along with its graphical representation, represented as XML structured information for the structural model which facilitates moving to different platforms and design tools (Ceri, Fraternelli & Bongio, 2000, p. 141). Several CASE tools are available, for example on http://www.webml.org. Here is also provided specification and documentation for how to use the language.

2.5.1 Web application modelling with WebML

WebML only describes a Web application at the conceptual level. In many cases this is not suitable for a Web application that incorporates extensive business logic. According to Ceri, Fraternelli & Bongio (2000, p. 138), WebML is compatible with UML in the structural model; however, new concepts and entities are introduced in the composition model, navigation model and the presentation model. These entities are not part of the specification of the UML so we conclude that this method does not have the properties of P1 (Build on the UML).

The WebML provides us with a formal specification but not the ability to reveal the architecture, as it defines an application from a conceptual point of view (Ceri, Fraternelli & Bongio 2002, p. 138), and thus have the properties of P2 (Formal reference specification) but not P3 (Reveal the architecture).

In WebML there is no clear definition of how to integrate the business logic objects of compiled code into models developed with WebML when a more detailed design of it is needed.

WebML could be suitable for integration into the UX (User Experience) part of the WAE-UML as its main focus is on information and navigation design which is a part of the User Experience.

2.6 WAE-UML

Both W2000 and WebML lack some of the properties we think a method must provide and their use is rather restricted due to the non-extensive information about some of them. However, with efforts from the authors of W2000 and WebML they could be substitutes or aid in the User Experience modelling which is incorporated in the latest updates to the WAE-UML.

With WAE-UML, engineered by Conallen, we can express logical design and connect it to the physical web pages in an architecture-centric and correct way. It allows hypermedia and information design as part of the User Experience modelling.

WAE-UML uses extensions to UML which allows the construction of full models of any Web application and supports the most dominant architectures, such as .NET, Java framework and PHP, but still provides an object-oriented view of an application (Conallen, 2002, p. 7).

Even though we speak about the WAE-UML as a method, what it does is only building on the UML with the definition of several stereotypes, tagged values and constraints for the correct modelling of a Web application. Since the WAE-UML extends the UML in this way we think that using this method for the design of Web applications is suitable considering that most developers already have knowledge of UML.

2.6.1 Notation

WAE-UML defines several stereotypes, tagged values and constraints to allow us to model a Web application. Appendix four introduces a small collection of the important stereotypes used for the redesign of the e-commerce application. A full reference and specification can be found in Conallen (1999, (2002, Appendix A)).

2.6.2 Unified Process with WAE-UML

Conallen (2002, p. 97) advocates that a prospective software engineering process could be the Unified Process (UP) or Rational Unified Process (RUP). However, any process that incorporates UML can be applied such as the Extreme Programming (XP) process.

User Experience with WAE-UML

An important introduction in the latest WAE-UML, as a part of the process that Conallen uses for modelling, is the UX (User Experience) modelling. With UX we can model the behaviour of the application from a user perspective in compliance with the use-cases. UX captures the design and distribution of both dynamic and static content for information design. Hypermedia design is also captured in the navigational map (Conallen, 2002, p. 187).

Conallen (2002, p. 193) introduces the notion of a <<screen>> stereotype for the construction of the design diagrams in UX. A screen captures static and dynamic content as seen by the user and is also used to define the navigational flow in the application. In addition, adornments can be used to express special properties of screens such as accessibility from other screens and scrollable screens.

2.6.3 Web application modelling with WAE-UML

Baresi, Garzotto & Paolini (2001, p. 9) gives criticism on this method suggesting that it privileges client-server interactions and underestimates the design of information and navigation structures. With the release of the WAE-UML, in Conallen (2002), it is introduced information and navigation design as a part of UX modelling. Since the UX modelling is tightly coupled and mapped to the analysis and design these problems are now arranged for.

The WAE-UML method introduces extensions to UML for the modelling of web applications. It provides a formal specification of the extensions used. While modelling with WAE-UML, according to our experience, we reveal the architecture of the behind the application. Since the method uses a use-case centred approach to modelling Web applications we think that it is a prospect to improve our quality attributes of the design artefacts.

In WAE-UML, the UML notation is extended with additional semantics and constraints to permit the modeling of Web-specific architectural elements as a part of the system's model. (Conallen, 2002, p. 4)

WAE-UML allows us to model the complex behaviour of a Web application while paying particular attention to the business logic behind it. Since large Web applications can contain large and complex business logic this is an important property.

3 Method

In this chapter we describe our scientific approach to the hypothesis and research questions. We also describe the methods for approaching the practical work and the evaluation.

3.1 Scientific method and approach

3.1.1 Case study

A case study incorporates an intense study of a phenomenon of interest and is used for in-depth study of problems to understand processes or a situation in context (Martella, Nelson & Marchand-Martella, 2003, p. 20). The emphasis is on understanding the phenomena. The most important aspect of a case, though, is that it provides us with means to examine different methods and models (Bengtsson, 2004, p. 13). Case studies mostly deal with qualitative data collection and can have multiple sources for the collection (Martella, Nelson & Marchand-Martella, 2003, p. 20). A case study contains five phases. The first phase constitutes means for analyzing and understanding the context. The next phase should identify the problems and set diagnoses. The third phase should provide options for approaching the problem. In the next phase we analyze the information and choose an option for the approach. The last phase presents the solution in accordance to the problem description. These phases are described by Bengtsson (2004).

We use the flow of a case study early in our thesis. In the first phase we try to understand the context in which we have worked in earlier Web application development. In phase two we identify the problems that have occurred in past projects and understand why they occurred. The literature study of different methods provided us with alternatives for using UML in Web application

designs. We choose WAE-UML as a way to approach the practical work. As a part of phase five we executed the practical work and presented the result as design diagrams with WAE-UML. These provide us with means to test the hypothesis that constitutes the thesis.

3.1.2 Qualitative vs. quantitative research

Quantitative research is based on statistical sampling and gathering of data with the primary purpose of describing present processes. The theories are developed in an inductive manner with the concern of studying natural events with no interference (Martella, Nelson & Marchand-Martella, 2003, p. 93).

Opposite to quantitative research we have qualitative research. Instead of describing present processes we determine the effects of a phenomenon. In a deductive way, theories rule the purpose of the investigation (Martella, Nelson & Marchand-Martella, 2003, p. 257).

According to Martella, Nelson & Marchand-Martella (2003, p. 256) there is no preferred method. Instead the choice is dependent on the type of questions to be investigated and answered. We have used a qualitative research method where we have tried to gather qualitative data by conducting interviews to get a personal contact and insight in the problems regarding Web application modelling with WAE-UML. We also tried to be open and give subjective personal opinions and experience to get insight and get in contact with the problem area. This gave us an objective and critical-eye approach for executing the practical work.

The practical works influences emphatic neutrality as a major role for us. As complete objectivity is impossible (Martella, Nelson & Marchand-Martella, 2003, p. 263), we have tried to have a neutral non-judgemental concentrated investigation governed by the hypothesis.

3.1.3 Analytical approach - reductionism

Complex problems can be better understood if they are reduced to smaller pieces. This is the fundamental idea of the research method reductionism. The idea is that theories for the whole can be derived from lower levels (Robson, 2002, p. 551). Since a piece of software and the blueprints describing it is indeed complex we seek to divide the problem of improving the software designs into smaller pieces.

We look at the design from three major views. From the condition of the design artefacts where we want to find improvements in the matter of verification and validation of the design diagrams according to the use-case and requirements specification. Next we look at the maintainability of those design artefacts and try to understand the maintainability issues regarding the UML and WAE-UML designs. From a software engineer and customer point of view it is important that the created artefacts do not only validate the requirements and use-cases, but also that they can tell the story about the software to the people involved. There are many different stakeholders in a software project and they all have different technical knowledge and can therefore understand a model in different ways.

3.1.4 Approach

The overall method is working by the defined flow of a case study where we are trying to see in what way, and how, an e-commerce design can be improved by using WAE-UML. In the case-study we take a qualitative and reductionism approach and hope to be able to conduct a general statement for whether or not the design artefacts can be improved by using WAE-UML. All our three viewpoints, the condition, maintainability and communication of the design artefacts, are equally important. From these viewpoints we try to deduce a conclusion whether we can improve the software design artefacts of Web applications as a whole.

3.2 Practical work

A comparison of the UML and WAE-UML variants of modelling Web applications is not possible without some practical influences. The practical work provides us with a way of structuring the information and presents the case in the form of a redesigned e-commerce system. The e-commerce system was originally developed in spring 2004 as a part of the course Small Team Software Engineering Project, PAB005 at Blekinge Institute of Technology. As a result of the initial case study we approach the practical work as the last phase in the case study.

3.2.1 E-commerce background

The customer and the main stakeholder of the e-commerce system is a large company dispersed among several different countries. The intended users of the e-commerce system are small and large retail stores. The management visions an increase among the retail stores using a computer and the Internet for ordering their supplies and needs to offer an alternative way of ordering their products.

For the retail stores, the e-commerce system features placement of orders, searching for articles, following the delivery progress, receive special offers and discounts and easy tracking of orders and invoices. As the retail stores possesses article catalogues and mostly orders the same supplies the e-commerce system does not contain any article tree structure for presentation of the articles. The retail stores personnel know the article numbers on their most popular articles so the need for an article tree is not required.

3.2.2 Delimitation

The e-commerce system is a large piece of software and as such constitutes a large design. We have chosen to delimit the remodelling efforts of the e-commerce system to two specific functions; the article search and place order functions. We think that we can still show the concepts and important aspects of WAE-UML modelling.

As we will focus on the analysis and design of two particular use-cases in the e-commerce system, only the requirements needed for these two functions will be presented. As the requirement specification is still exhaustive, it excludes the source, purpose and dependencies as we think that these are not important nor required for the understanding of modelling.

When referring to "the customer" in the requirements it means the user of the system, the retail stores. Appendix one contains the use-cases we choose to redesign and appendix two contains the requirements for these use-cases. Appendix three shows the design artefacts for the e-commerce system.

3.2.3 Conduction

The redesign of the e-commerce application was performed according to the process outlined in Conallens book about WAE-UML. Since we choose to use the method WAE-UML this process of work was the most reasonable. Conallen

describes the method as a mixture between the Rational Unified Process (RUP) and ICONIX Unified Process. The method also incorporates the User Experience (UX) into the process (2002, p. 97).

3.3 Evaluation

The evaluation consists of our own evaluation and conducted interviews with relevant people working in the industry.

3.3.1 Own evaluation

The fundamental idea of evaluating a design is to evaluate the correctness and accuracy as it conforms to the requirements specification and use-case scenarios, provided that the SRS (System Requirements Specification) and use-case scenarios are accepted as correct. In this thesis we presuppose that the requirements and use-cases, which we base our model on, are correct in the matter of stakeholder acceptance and indecency.

There are several ways for assessing software architectures but these methods main focus is on assessing the quality attributes (non-functional requirements) of software architectures. The methods suggested in Bosch (2002, p. 91-103) are scenario-based assessment, simulation-based assessment, mathematical model-based assessment and experience-based assessment. However, since the focus for these methods is on quantitative and qualitative measurements of the quality attributes of the software architecture these methods do not apply to the thesis; we go deeper into an application than to the architecture level. However, we have borrowed the idea for our inspection evaluation from the definition of experience-based assessment which is briefly mentioned in Bosch (2002, p. 103). What we are trying to do is to see how the designed system fulfils its functional requirements and use-case scenarios, not the architecture quality attributes (the reason is that the quality attributes as regarding to the original system was minimal) and hereby see if we can get good measurements for making a comparison of the two designs.

The idea of experience-based assessment is to use an external software architecture assessment team that evaluates the architecture (or design) by objective reasoning based on earlier experiences and logical argumentation (Bosch, 2002, p. 103). We have tried to do this reasoning ourselves based on the old and new design.

There is no single method or mechanism established for evaluating a software design. The current practice is evaluations made by walkthroughs and inspection of code or designs. These tasks are time consuming and demand the reviewers to control a lot of information. As the evolution of the design proceeds during UX, analysis and design there is a need for a stepping through of the produced artefacts for recognizing compliance with the requirements and use-case scenarios.

Trung Thanh Dinh Trong (2003) suggests a procedure for systematically testing UML design models. However, this method does not solve the incompleteness nature of a design model. Since this thesis evaluates an old and a new design model and the fact that the new design resolves the issues of incompleteness in the old design this method can not easily be applied since the old design model is incomplete.

We found that an appropriate way to evaluate the two designs was to follow the IEEE standard for software verification and validation regarding the design phase. The standard suggests traceability analysis, software design evaluation, interface analysis, critically analysis and component V&V (Verification and Validation) test as part of Design V&V activity (IEEE, 1998, p. 31). We found that a sufficient level of evaluation and a quality measure for comparison between the two designs could be provided by using the traceability analysis in compliance with the standard.

To verify the condition of the two designs according to the IEEE standard we went through an inspection process (Sommerville, 2001, p. 425) of the two designs. The condition of a design according to us would be to grade the relationships in the traceability analysis and the software design evaluation. A useful value is also the detection of faults and inaccuracies and of course the expert opinion. Since it is a hard task to detect faults and inaccuracies in a design that we have made ourselves, we focus on providing an expert opinion.

In this evaluation of the designs we will use a mixture of experience-based assessment (Bosch, 2002, p. 103), inspection process (Sommerville, 2001, p.425) with the help of the IEEE standard for software verification and validation (IEEE, 1998). Even though the old design is not that detailed we think that this method of evaluation will give us a fair value of the two ways of modelling. In our own evaluation we will look at the traceability of the requirements and use-cases to

the design artefacts and see if the requirements and use-cases are fulfilled by the design.

3.3.2 Data collection method

Interviews can be conducted for qualitative analysis. Formal interviews are planned in advanced and conducted in an environment where it is possible to go deeper into the subject. The interview is lead by a person with the purpose of gathering information (Ely, et. al., 1993, p. 65). An interview can be structured, semi-structured and unstructured (Robson, 2002, p. 270). The style that applies most to the thesis and interviewees is the semi-structured variant where questions can be changed and added depending on the interviewees' perception and knowledge. Robson (2002, p. 271) describes the circumstances in which a qualitative research interview is most appropriate. In qualitative research this approach is the most widely used according to Robson (2002, p.271). We think that this method is most appropriate and can provide us with meaningful statements from the interviewees as qualitative data.

The data collection provided by interviews can be described as qualitative. The data collection is performed with in depth interviews. The personal contact and insight is important in qualitative data collection as it captures direct quotations of people's personal perspectives and experiences (Martella, Nelson & Marchand-Martella, 2003, p. 263). This is the reason why we chose to conduct semi-structured interviews with people, in the industry, and experts to collect qualitative data.

4 Realization

In this chapter we describe how we realized the practical work.

4.1 Practical work

The e-commerce design we were looking at provided us with the use-case and requirements for the article search function and the place order function. It also provided us with mock-ups of the user interface; these can be found in Appendix five. We decided to work in a similar way when redesigning the Web application. Therefore we used the UP (Unified Process), which is almost the same as Conallen uses in his latest book (2002), in such a way as performing the analysis and design phase. An initial phase was introduced which incorporated

UX modelling as a first phase. The method we followed thus constitutes UX modelling, analysis and design.

In the User Experience phase we defined the screens through the use of mock-ups, use-cases and requirements for the functions article search and place order. With the help of the screens, the use-cases where then organized into storyboards. Storyboards are a way of telling a story about a particular use-case through discrete static pictures (Conallen, 2002, p.204). With the help of the storyboards and screen definitions we could construct a participants diagram for each of the use-cases and combine these two together to extract a navigational map over the whole system functionality (see chapter 11.2.1).

The analysis phases started with mapping the screens from the UX phase to domain boundary classes and then define a domain model from the boundary classes and additional controller classes and entities. Each use-case could then be visualized in a sequence diagram consisting of the cooperation of these classes and entities.

In the design phase the sequence diagrams, from the analysis phase, were developed further to show more details and for mapping them to the class diagrams (see chapter 11.2).

4.2 Evaluation

To evaluate the redesigned e-commerce system we are forced to make a comparison of the condition, maintainability and stakeholder communication between the redesign with WAE-UML and the original version designed with the UML. We choose to divide our evaluation in two parts, verification and validation of requirements and use-cases and experience-based assessment. These parts are performed with the original version and the redesigned version to provide measurement for comparison. In the experience-based evaluation part we put ourselves into observers and try to reason about the designs paying particular interest in the quality attributes we have specified. In these circumstances we see ourselves as experts doing a similar processing of the designs as in experience-based assessment. Since this method is a discussion it is likely to be similar reasoning in the result section (chapter 5) and discussion section (chapter 6).

To get additional external opinions from experts and from persons working in the industry we conducted interviews. The focus here is on all the quality attributes. We also try to see if the interviewees can trace the requirements and use-cases in the design.

4.2.1 Interviews

The interviews were conducted in two phases. The first phase was to let the interviewees conduct a design study to get familiar with the design diagrams from both designs, as suggested in Sommerville (2001, p. 427). The interviewees were presented with the article search use-case and requirements along with the diagram implementation using the UML and WAE-UML.

In the design study we asked the interviewees to get familiar with the diagrams of the UML and WAE-UML designs and read through the use-case and requirements to get prepared for the interview. The diagrams where navigational map, sequence-/collaboration diagrams and class diagrams. These diagrams are presented in Appendix three.

The interviews were planned to be held in a neutral environment because it makes the interviewees more relaxed. However, most of the interviews were conducted in the interviewees own working environment. The reason was that the interviewees had a limited time to offer and proposed that we would come to their offices. We gave in to their proposals because we felt that this could also help them relaxing because they were more familiar with and accustomed to the environment. In this context we were more emotional vulnerable as they had the advantage of being superior.

We tried to listen more than we spoke and ask the questions in a straightforward manner. We also tried not to force the interviewee to believe that any of the design alternatives where better than the other. These aspects of an interview is pointed out as important by Robson (2002, p. 274). In this way we get the interviewees to feel and speak more free and open according to Robson.

We had planned to record the interviews for permanent reference. Robson (2002, p. 289) argues that an interview should be recorded for future reference and that it allows the interview leader to concentrate on the interview. According to Freedman & Weinberg (1990, p. 128) it is not a good idea to record design evaluations and interviews because it can prohibit the interviewee to say what he or she wants. They are less eager to express their opinions. We choose to use a

laptop to write down the interviewees answers. We think that we have captured the most important statements in this way and received a more honest opinion from the interviewees.

In addition we have followed the four basic requirements for ethical research where the focus is on protecting the integrity of the persons involved in the research (Swedish Research Council, 2002, p. 6). In short these requirements provide means to protect individual participation in the research; for example, to inform the participants of the purpose of the research. Participants should also have the right of self-determination to participate in the research. Further, all personal information from the informants should be confidential and protected so that no unauthorized people can access it. The information collected should only be used for research means. These principles are all recommended by the Swedish Research Council (2002 p. 6).

Selection of interviewees

The interviewees were selected by us to receive different viewpoints. Two work in the industry of developing Web applications, two was a part of the team developing the original e-commerce system and one is a UML expert. We did not choose these four persons to lead the resulting data towards favouring WAE-UML but to get valuable statements seeing them as experts in their fields. We did not believe that a person without any knowledge of UML or software engineering methods would add any value to the data collection.

Interviewing two persons that more or less were involved in the development of the original e-commerce system was a deliberate choice. As such we were sure that they had experience in Web application development with UML. We wanted to see how they perceived the new design with WAE-UML. In addition we believed that they could provide valuable opinions about the problems with using UML and if WAE-UML could improve the design.

Questionnaire

Robson (2002, p. 275) lists some aspects that are important to think of when you design the interview questions. In short these are to avoid long and double-barrelled questions, questions involving jargon, leading questions and biased questions. In designing our questionnaire we have tried to follow these aspects.

The questions were divided into three major areas. First we wanted to check the background knowledge of the interviewees. We asked them if they were familiar with development of Web applications and what methods and technologies they had been used. We also asked if they had previously worked with or were familiar with UML, WAE-UML and stereotypes.

The second area of interest was the design artefacts. These questions were performed separately by the interviewees for the UML design and then the WAE-UML design. Because we knew that the WAE-UML design provided more information about the system functionality we started to ask the design questions about the UML design first and then the WAE-UML design. In this way we meant to capture additional aspects about the newer design since we knew it provided more detailed information.

In the third part we wanted to see how much they had grasped about the concept of WAE-UML and if it could be any help to them in their work. In this way we wanted to measure the popularity among the two methods. Additionally, we asked them which design they would choose being in a certain situation as a customer, executive and programmer. The complete questionnaire can be found in appendix six.

5 Result and analyze

In this chapter we present the result of our own evaluation which includes a summary of our own opinions about the diagrams developed with WAE-UML along with the verification and validation of the requirements and use-cases. Further, we present the material collected during the interviews.

When the word design is mentioned it is in regard to the navigational map, class diagrams and sequence diagrams as presented in the appendixes, chapter 11.2.

5.1 Own evaluation

Our own evaluation is divided into two main sections. In the first section we present the verification and validation of the requirements and use-cases.

The second section lists our own comments on the properties of condition, maintenance and stakeholder communication, regarded to the WAE-UML variant and in opposite to the UML variant, as the result of the experience-based

assessment. The statements in this sub-chapter does not have any scientific evidence but is based on previous design experience and the practical redesign of the e-commerce system with WAE-UML with the help of objective reasoning as observers. A critical discussion of these statements is provided in subsequent chapters.

5.1.1 Verification and validation

The verification and validation of the requirements for both the UML variant and the WAE-UML variant, separately, are summarized in diagram 1. Here we present the percentage of fulfilled requirements.

The criteria for a requirement to validate to "fulfilled" is that it can be traced to the design and that all details in the requirement is visible in the design. A requirement that validates to "partly fulfilled" can be traced to the design but not all details in the requirement is visible in the design. A requirement that is "not fulfilled" cannot be traced to the design at all. These criteria are not part of the standard but set by us to provide means for measurement.

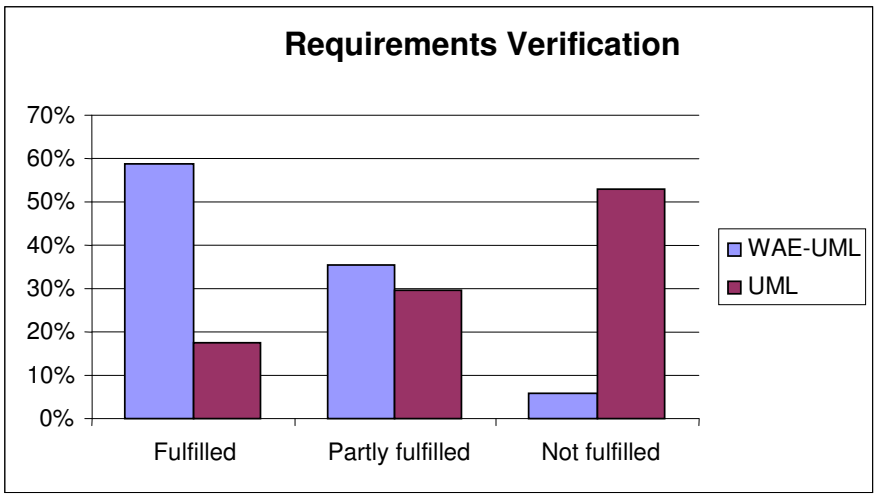


Diagram 1: Experience-based assessment with requirements verification of WAE-UML and the UML design.

This result can be a bit misleading. Most of the people that were involved in the modelling of the original e-commerce application with the UML were not very interested in modelling at all. However, the cause of this view among the designers can be traced to the fact that the UML does not provide semantics enough to model Web applications. It then becomes an exhaustive task to model it.

Opposed to the requirement verification the use-case verification can either be "fulfilled" or "not fulfilled". In the UML design we did not manage to trace nor validate any of the use-cases basic flows. The same use-cases basic flow could be traced to the WAE-UML design. The reason that the use-cases are not fulfilled by the UML design can be the same as for the requirements not being fulfilled. These criteria are set by us to provide means for measurement.

5.1.2 Experience-based assessment

This chapter is based on our evaluation of the WAE-UML design. The diagrams mentioned are the navigational map, class diagrams and sequence diagrams. These diagrams are presented in appendix three, chapter 11.2 - E-commerce redesign.

Condition

Our definition of the condition of the design artefacts means that the design artefacts should prove to fulfil the requirements and use-cases, resemble reality and not be misleading.

According to us, the most important document produced by the UX is the navigational map. It shows the hypermedia and information design in the form of screens. The navigational map developed for the e-commerce application redesign does not include all of the information necessary though. As Conallen (2002, p. 187) points out; the UX modelling team and design team work in parallel and most of the UX modelling has its roots in the human- computer interaction and should not be a part of the tasks assigned to the design team. Thus, the focus of the latter stages in UX modelling should focus on improving the hypermedia and information design. However, since UX provides a good start for the rest of the development and that the navigational map and the screens are direct input to analysis a less detailed UX effort should be required before the analysis begins. As the UX modelling is use-case centric a good assurance can be assumed that the use-cases will be fulfilled at least at the start of the analysis and design phase. For a more secure assurance in this matter verification and validation of both the use-cases and requirements can be applied between the process phases.

Assuming that a stakeholder possesses adequate knowledge of the stereotypes used for modelling screens and for creating the design diagrams, the documents should not be misleading. The navigational map shows simple things in the form of screens but gives a good view of the navigational structure and information distribution in the system and cannot easily be misinterpreted. The class diagrams and sequence diagrams from the design phase does also adhere to this property. In fact, the design diagrams reveal the underlying architecture of the system and as such are a prospect for generating code with a CASE tool provided that the correct semantics has been used.

We can not draw any conclusion if the design diagrams resembles reality as the aspect of reality is to the observers own view. We think, though, that the fact that the design diagrams reveals the architecture, it does resembles reality if adequate knowledge of the stereotypes used are possessed. The design resembles reality in the aspect that WAE-UML divides a web page into its real components. The web page is divided into a server page, a client page and its contents in the way of forms and scripts. In this way we can see the actual architecture and the design is more likely to resemble reality. Figure two shows the concept of a revealed architecture. In the WAE-UML extraction we can clearly trace the .NET architecture. In the UML variant these classes are modelled as a single entity.

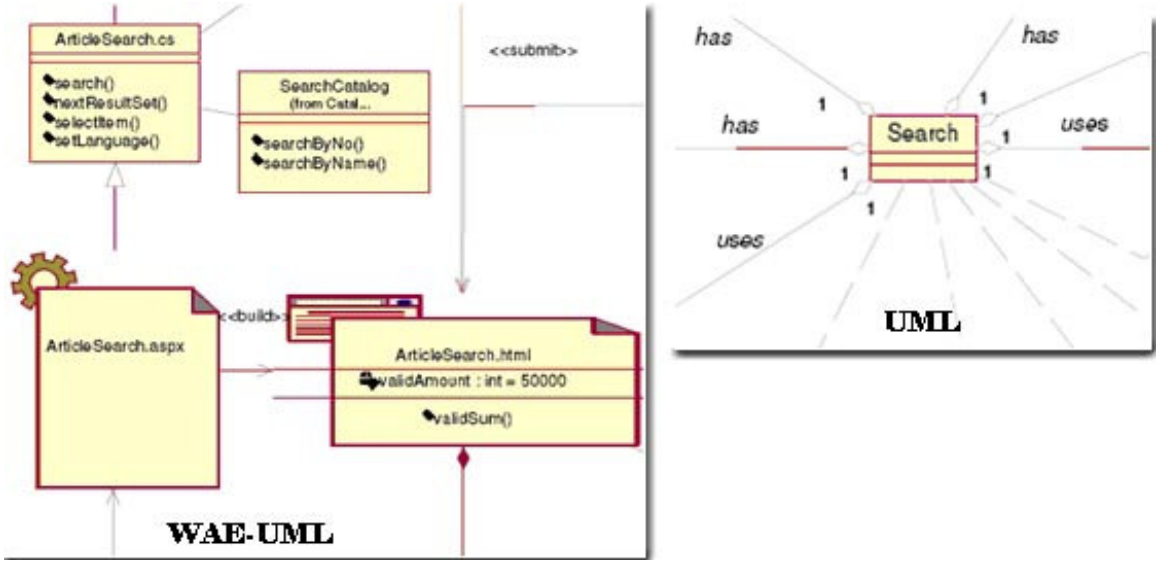


Figure 2: Article search Web page with UML and WAE-UML

Figure two shows extractions from the UML design and the WAE-UML design. The WAE-UML extraction shows that the Web page is composed of a client page, a server page and a code behind class. In the UML extraction the Web page is

modelled with a single object supposed to include the client page, server page and the code behind class.

Screens in the navigational map resembles reality in the way that they are the physical objects that the user sees and provides him with input fields and dynamic behaviour (read navigation).

Communication

The communication aspects we are looking at means that the design should prove to provide unambiguous artefacts, be easy to understand and communicate the real message in compliance with the use-cases and requirements.

We can look at communication from two viewpoints. The reader of the diagrams does not have any knowledge of software development and UML notation, such as sometimes in the case with customers. The reader of the document is a programmer or designer with medium to high knowledge of software development and UML notation.

If we suppose that a stakeholder has the required knowledge of the software development process, UML and WAE-UML stereotypes it is fairly easy to interpret and understand the design artefacts depending on the design diagrams level of detail. The design artefacts produced with WAE-UML tends to show a lot of detail in the design, especially the sequence diagrams (see appendix three, chapter 11.2.5-11.2.9). This can be both positive and negative from a communicational aspect and from a maintainability point of view. The good part is that a high level of detail certainly is to resemble reality better than an abstract design. In the diagrams in appendix three, chapter 11.2.5 – 11.2.9, we can clearly see what is happening behind the scene. On the bad side it has limitations on the communication and maintainability aspect because it will be hard for the stakeholders to interpret and grasp the complete picture of the system. For the same reasons, as with the condition of the artefacts, since the architecture is revealed in the design we can state that the artefacts are less unambiguous and communicate the real message of the application (See example in figure two).

For a stakeholder, e. g. a customer or a person with less knowledge of UML, the navigational map can present a good way to mediate a comprehensive view of the system from the users' perspective. Using mock-ups and the navigational

map, the message of the application design can be communicated to a customer. The navigational map mediates the basic flow of information and navigation between screens (Web page views) if the user submits a form or navigates to a new screen. A large Web application can contain many Web pages and a complex navigation structure. The navigation structure is also haphazard in the context of the users' navigation habits. Therefore, all legal flows in an application cannot be modelled. Conallen agrees on this viewpoint and provides guidelines to overcome this problem (2002, s. 210). Even though the example navigational map developed during the redesign of the e-commerce system contains a very small amount of screens it is still complex. The navigational map provides links and user input on a less technical level of understanding and can be mediated regardless of the receiver's technical understanding. Once a stakeholder grasps the concept of a screen the navigational map is easy to understand.

Maintainability

There are two types of maintenance on a piece of software and its blueprints. When the application is under development maintenance of the documents can be done by programmers or by the design team. When the application is delivered, bugs must be fixed. Most applications are extended or redesigned in the future. Both aspects require maintenance of the design diagrams. These two types are equally important, but to be able to maintain the design during post-delivery maintenance they should be updated during the implementation phase.

Opposed to previous design efforts with the UML, UX modelling with WAE-UML provides a good starting point for the whole design process. Directly starting with the use-case analysis phase can be an obtrusively hard task when designing a Web application. This quick start modelling should improve the commitment in the development team to maintain and follow the documents during the whole process.

Most of the UX design artefacts are to our opinion easy to change and update. Because hypermedia and information design can be complex it is easy for both the UX diagrams and the design diagrams to become confusing and messy if the application under analysis is large. It restrains maintenance and prohibits communication. To some extent this can be limited by the use of adornments. Adornments describe additional properties of screens. In figure three we have limited the relations of the article search screen by the adornments $-sign and +-

sign. The $-sign tells us that the ArticleSearch screen can be navigated to from every other screen in they application. In addition the +-sign describes that the result of the search is presented to the user using several screens. In this way we can reduce the complexity of modelling this with relations between screens and delimit the search mechanism to one screen. A more in-depth description of adornments and their use is provided by Conallen (2002, p. 210).

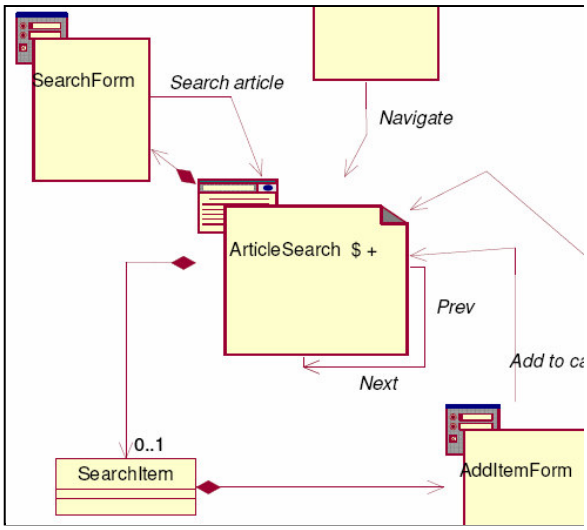


Figure 3: Article search navigation screens – WAE-UML

For the same reason of complexity the design diagrams can also become confusing and messy if the application is large. Modelling a large application the documents can become too extensive in size which leads to that they are hard to handle and less perspicuous. It can also be hard to demarcate the documents into smaller pieces, e. g. the sequence diagrams, class diagrams and storyboards.

5.2 Result and analyze of the interviews

5.2.1 Condition

We asked the interviewees to see if they could trace and verify that the requirements for the article search use-case was fulfilled in the new and old design diagrams. In the old design, requirement one and two could be traced and verified by three of the interviewees although some guessing was required. Some aspects in the design were unclear to them. However in the new design all interviewees proved to be able to trace and verify the requirements one and two. Requirement number three was not fulfilled according to three interviewees. One interviewee could indeed verify the requirement but he was a part of the old

design project team. Seventy-five percent of the interviewees could trace and verify this requirement in the new design. The fourth could verify it when pointed out that the two sequence diagrams were linked together in an ad-hoc way. Requirement number four could not be verified by any of the interviewees in the old design but fifty percent could verify it in the new design. The result of the requirements verification is summarized in diagram two where we present the percentage of the interviewees' ability to trace and verify requirements one to four.

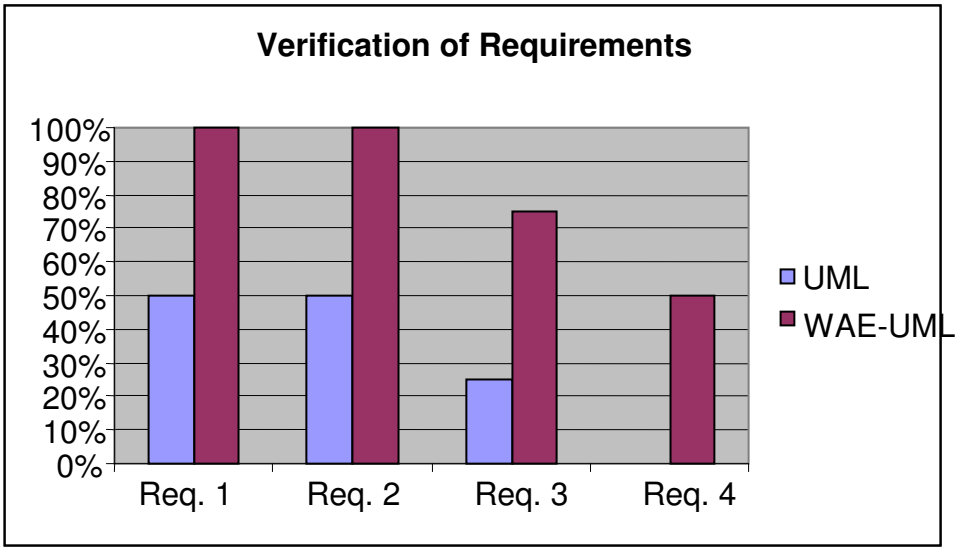


Diagram 2: Interview requirement verification of WAE-UML and the UML design.

We also asked the interviewees to try to trace and verify that the use-case was fulfilled by the UML and WAE-UML design artefacts separately. None of the interviewees could trace the use-case and verify it in the UML design documents completely. The WAE-UML design showed to fulfil the use-case basic flow to all the interviewees. However, one of them thought that there were too much detail in the WAE-UML design and it was hard to connect it to the use-case specification.

5.2.2 Communication

To get a feel for the communication properties that the old and new design holds we asked the interviewees to describe what they thought the system did looking first at the old and then the new design. The two interviewees that had been involved in the old e-commerce design proved to grasp the concept of that it actually was a search mechanism described in the old diagrams. The other two

showed more willing to find such information as that there was a database included, that it was possible for a user to login and that the user seemed to be able to place orders of some kind. Looking at the new design the two interviewees that were not part of the old e-commerce design team seemed to grasp the concept of that the new design indeed was a Web application and that it could be used for some kind of search mechanism. The interviewees that participated in the implementation of the old design found additional properties in the new design documents such as the possibility to search for articles and then add them to an order up to a certain amount.

When the interviewees where asked if they would be able to implement the article search function given the UML diagrams, two of them were not sure if they could do that given only the design diagrams. It would help them to use the use-case specification and requirements. Two of the interviewees were quite certain that they would be able to implement it; however they were both familiar with the UML diagrams since they were part of the old design team. All the interviewees were positive that they could implement the article search function with the help of the WAE-UML diagrams. It was commented that the WAE-UML design code could easily be auto-generated with a CASE-tool. As one of the interviewees was more familiar with PHP instead of .NET he would have chosen to use that instead.

We asked the interviewees to explain what happened if an article was added by the user to a non-existing order. In this way we wanted to see if the design documents were clear enough for what happened if an order did not exist in the diagrams. It seemed that none of the interviewees could trace in any of the diagrams, not UML diagrams or WAE-UML diagrams, what would happen.

5.2.3 Maintainability

The interviewees were asked to change a small functionality in the WAE-UML design documents. In that way we wanted to see how hard it would be to change a small function in the article search functionality. Three of the interviewees claimed they would be able to redesign it although they could not point out how to change the diagrams. One interviewee showed the correct way of doing this in the diagrams with some help provided by the interview leader.

5.2.4 Performance

From a performance perspective in the UML design and WAE-UML design separately, we asked the interviewees how they thought the implemented article search would perform. We tried to percept the feel and understanding they had for the design variants by asking this question.

One interviewee argued that the performance of the UML variant wouldn't be too slow because it showed high cohesion. The rest thought that this variant would be very complicated and lumpy in performance. The WAE-UML variant proved to be more popular by the interviewees from a performance perspective. However, one interviewee thought that it would be worse due to the fact that he could now see that the system used a lot of redirection but showed to improve the cohesion issue from the UML design.

5.2.5 Interviewee opinion

The interviewees were asked to become a customer and then asked from this aspect which design they would choose. Three interviewees would have chosen the WAE-UML design, however, from a customer perspective the design diagrams showed too much details. One of the interviewees would have required only the use-case specification as a customer. From a programmer perspective they all would have chosen the WAE-UML variant because they were more perspicuous according to them.

The interviewees thought that the WAE-UML variant could indeed help them in future designs although the people working in the industry thought it would require too much effort to develop these design artefacts and too much time would be required to learn the technique.

6 Discussion

The data analysis and practical work in our qualitative research have proved to be slightly affected by our own stance towards WAE-UML. As we found this way of modelling appealing and easy it might have affected our critical stance towards the other methods and the objectiveness towards modelling with WAE-UML.

In the introduction part of the thesis we stated a few problem areas regarding the modelling of Web applications with the UML in previous projects according to

our own experience. The problem description circulated in three major areas, namely the condition of the design artefacts, ability for communication between stakeholders and design diagram maintenance. We defined condition in regard to verification and validation of requirements and use-case. The design artefacts should resemble reality and not be misleading. Communication between stakeholders defines the provision of unambiguous design artefacts that are easy to understand and that communicate the real message in compliance with the use-cases. The last important quality attribute of the design artefacts were maintainability. This attribute defined the ability and ease of maintaining and update of the design artefacts. The following discussion of the result does circulate around these three quality attributes.

6.1 Condition

When we traced and validated the requirements and use-cases towards the designs we found astonishing results. Fifty percent of the requirements and none of the use-cases were completely incorporated in the old UML design. The requirement verification and validation done by the interviewees provides similar result as in our own verification and validation. The use-case verification and validation also provided exactly the same result as in our own evaluation. Even though we are unsure that the e-commerce UML design is representative in an industry perspective, this result must be regarded as poor from a software engineering point of view. The result of this is that the programmers have no value using the design artefacts, but have to use the requirements specification and use-cases, not to forget their imagination, when implementing the system. Two interviewees mentioned that they did not use any specific modelling technique, and mainly were dependent on specifications and descriptive text documents. Those interviewees worked in the industry developing Web applications. WAE-UML showed to improve the requirement and use-case traceability radically even though not all requirements could be traced. The improvements can be connected to the use of User Experience as a part of the early stages modelling with WAE-UML; since UX incorporates, in an early stage, the use of use-cases in the modelling. The consequence is that the use-cases can be traced all the way to the finished design if careful attention is taken during later stages. However, the interviewees were reluctant to adapt a new technique of modelling Web applications because of the time aspect.

Another way of looking at the requirements is that the modelling environment can not be the same in the two circumstances in our case. Some of the aspects

that play a role are, first, the people modelling the two versions were not the same except for one person. This might influence the condition of the new design. Second, as a researcher, it is hard to take an objective standpoint when he or she knows that the work will be judged.

The interviews showed that a person is more willing to see that the architecture is a Web application in the WAE-UML design. We can conclude by this that the WAE-UML design is more likely to mediate a more truthful *reality*.

6.2 Communication

A result of using UX is that we can reduce the complexity of performing analysis and design. In the redesign work the UX modelling provided us with a good quick start to modelling, adapting a user perspective. Because the use-cases were a big part of UX we were forced to thoroughly understand them to be able to perceive and perform the UX modelling. From a designer's perspective, understanding the use-cases is vital for the communication. Conallen advocates that the UX modelling should be performed by a separate group. Opposite from Conallens view, we think that if the UX modelling is performed by the designers, or in cooperation with this separate UX group, it can improve the communication in the development team by improving the understanding of the use-cases and the system functionality. It then can mediate the real message in compliance with the use-cases and requirements to other stakeholders.

As all of the interviewees were very positive that they would be able to implement the WAE-UML design easier than the UML design we can say that the new one provides a less *unambiguous* design and are easier to understand. In the context of condition it is not likely to be *misleading*. However, since none of the interviewees were able to trace what was happening if an order was not created before adding articles in any of the design, we suspect that this question was too detailed and complicated to answer in a good way. Since one of the interviewee misunderstood the question and was referring to an order object instead of a physical order we suspect that the question had certain indistinctness. Out of this result we can not draw any conclusions, for any of the designs, in the matter of ambiguousness.

6.3 Maintainability

The maintainability of design artefacts is hard to measure. In the practical work we were sometimes forced to accomplish small redesigns due to the fact that we missed vital parts of the use-case. Looking at maintenance from this aspect we managed to accomplish the redesigns quite easily. The interviews on the other hand proved to us that a person that is not that well involved in the design and not so accustomed to the semantics of the language, and the stereotypes, is obtrusively much less probable to be able to accomplish a redesign. In this aspect it is important that the maintainers and programmers, if they are to update the documents during implementation, are well accustomed to the design and possesses adequate knowledge of the semantics used. The level of detail in the documents is also an important aspect of maintainability.

The interview question about maintainability was too detailed to be answered with the UML design. We asked this question to get opinions at to what extent the WAE-UML provided maintainability. Even though we thought the change in the WAE-UML design should have been very easy to do none of the interviewees seemed to see to be able to show us how to perform it during the interview. We suspect that it would have required a longer design study and better preparation of the interviewees to perform this redesign.

Because with WAE-UML we can divide a web page object into its real components, server page, html page and forms, among other, in the design the cohesion of individual objects are affected by the modellers' choice of responsibility distribution. This can in turn affect maintainability because the responsibility of objects can become too related to one another that enormous complexity is inherent. Then it will be very hard to break them and redistribute the responsibility for these objects. This division of web pages also has influence on the ability to demarcate the system design and compose several smaller design documents instead of a large.

6.4 Test of hypothesis

Since we use a qualitative method for examine the relationships between a UML design and a WAE-UML design we can not test the hypothesis in a statistical way. The attempt to test the hypothesis is based on experience-based assessment and the interpreted qualitative data from the interviews.

During the practical work we showed how the UML could be used to model a Web application by the use of an extension method, WAE-UML. According to our objective appraisal we can try the hypothesis in the e-commerce redesign particular case only. By no means can we state that WAE-UML provides improvements in all Web application designs.

To reason about the hypothesis we investigate the quality attributes that constitutes it; we need to look separately at the condition, communication and maintenance. The redesigned e-commerce system improves many aspects when it comes to condition. All use-cases and most of the requirements are fulfilled. Depending on the level of detail as chosen by the designer it does resemble reality better and is less misleading for the programmers. Although we cannot prove that this is true in the general case we see an improvement of these aspects in the redesign.

Again, depending on the level of detail used by the designers the WAE-UML version provides us with unambiguous artefacts, that are easy to understand, and communicates the real message in compliance with the use-cases and requirements.

The maintainability aspect is improved in they way that the programmers are most likely to find the design artefacts useful and commit to updates and maintenance during the implementation. As the interviews showed, the designs can contain too much detail. This means that maintenance personnel other than the original implementers have it harder to understand the design and perform updates of it. If these people do have a common knowledge of the WAE-UML it might be a bit easier.

Even if the redesigned e-commerce application artefacts provide improvements in all aspects of the hypothesis we can not hold for certain that it is true for all Web applications. There might be a lot of different technologies and aspects involved that can have influence on the quality attributes. The level of detail in the design artefacts plays a large role in this influence and it is up to the designers experience to model the Web application at a sufficient level. If this level of detail is attained, WAE-UML provides an interesting way of modelling and can improve the design artefacts according to the discussed aspects.

6.5 Goals

Provided with an extension mechanism, the UML is certainly suitable for the modelling of Web applications. In the practical work we incorporated the UML with the extension to UML, WAE-UML. We showed how pre-existing knowledge of the UML can indeed, indirect by using the extension method WAE-UML, be used for modelling Web applications. Since WAE-UML is developed from UML with extensions we have found a good way of introducing UML modelling into the Web application industry. Even though the industry people were a bit reluctant to adapt such a process into their work, due to the time aspect, we think that it can ease the maintenance and increase implementation speed. According to our previous experience, both with Web application and traditional desktop programming, it is much harder to program using a descriptive specification than by a visual design. A formal and correct design can also be used to generate stub code by a CASE tool which would decrease the time spent on implementation. It is also important to note that a correct design that proves to trace and validate the requirements and use-cases decreases inconsistencies in the program code. More requirements will be implemented in the product from the start. This might decrease the time spent on testing and error fixing. That is why it is important to model also Web applications. As most software development projects have budget and time limits, modelling should improve both aspects.

From our experience of the work with this thesis we have developed a method and instinct into modelling Web applications with WAE-UML. Since this method showed to provide many improvements of the design we will in the future be able to create better design artefacts for Web applications. Depending on the security aspects of the application under development we would choose to develop Web applications instead of the more traditional client/server applications.

6.6 Discussion of methods

The interviewees can have been influenced during the interviews in a couple of ways. The UML diagrams were printed in black and white while the WAE-UML diagrams were printed in colour. This was pointed out to us by one if the interviewee. The interaction modelling of the use-case was described by a collaboration diagram in the UML variant and as a sequence diagram in the WAE-UML. The amount of influence these aspects had on the interviewee's opinions is uncertain. Both these diagrams, collaboration diagrams and sequence

diagrams are compatible and describe the same things. We choose to do the WAE-UML design with sequence diagram because they provide a better structure if the diagrams contain many details. Collaboration diagrams with a lot of detail easily get complicated and grow in all directions. Sequence diagrams grow mostly vertically and can be presented and printed easier. Both diagram types can be constructed with the UML and WAE-UML.

Robson (2002, p. 273) argue that an interview under thirty minutes does not give too much value. An interview above sixty minutes is inappropriate due to the time consuming aspects for the interviewee. It can also decrease the willingness of the subjects to participate (Robson, 2002, p. 273). Robson (2002, p. 273) also believe that this may lead to biases in the outcome of the interview. Due to the fact that the interview is a long process and that we required the interviewees to perform a design study in advance, we were forced to decrease the amount of work load on them. A selection of one use-case for the interview was necessary. The selected use-case in turn was selected as to be as small and uncomplicated as possible. We think, however, that this use-case reflects the system as a whole and can provide a measure for the whole e-commerce system.

Two of the interviewees had certain previous knowledge about the system in matter. This could have affected the questions about the functionality of the system.

We noticed a different level of preparation among the participants. Those who did not do the design study in advance or not enough preparation in the design study were eager to do more guessing than the others and it took longer time for them to reason and answer the questions. This does not necessary mean a negative impact on the result on all aspects. It might have had a negative impact on the ability to trace and verify requirements and use-cases. On the other hand, less preparation provided a more honest explanation of the functionality.

The final phase in the case-study, the redesign of the e-commerce system, could have been done differently if more time had been available for us. An alternative could have been to compare all four ways of modelling, UML, WAE-UML, W2000 and WebML. Since designing a software system takes a long time, and we did not have sufficient time to redesign the e-commerce system three times, we choose one alternative only. The reason is also that making three designs of the same system would have improved our understanding of the design during the second and last design. These designs would have been better then regarding requirements and use-case traceability and verification. We think that by

evaluating the different methods by examine the documentation about them we have chosen the most appropriate method for improving the UML design.

Working in the form of a case-study provides ways for thoroughly examine the problem area and discover methods for solving it. In this way we could probe the field of Web application modelling. We found the appropriate way of ending the case-study flow by presenting the redesigned version of the e-commerce system in the form of the resulting WAE-UML diagrams.

The evaluation could have been strictly reduced to the interviews and verification and validation of requirements and use-cases. However, to perform a sort of experience-based evaluation provided us with additional information. This information was in many aspects confirmed by the informants in the interviews.

6.7 Summary of the discussion

With WAE-UML we can improve the quality attributes of the designs when developing Web applications. In the discussion of the studied case result we have repeatedly returned to two consistent aspects. The result is dependent on two variables, namely the level of detail in the diagrams as chosen by the designer and the readers understanding and knowledge of the semantics. We think that these two aspects are a crucial part of people involved in any application design process and implementation. Designing Web applications with WAE-UML can improve the overall design in the aspects of condition, communication and maintainability depending on these two important variables.

7 Conclusion

WAE-UML is by no means completely error free in its semantics. We have found some semantic inconsistencies in the specification that can be improved and provide for future research. One of the interviewees also agreed on this and came up with suggestions of improving WAE-UML. We can for example introduce several more stereotypes that can improve the understanding of the design. This would make it easier to model any type of Web application regardless of the chosen technology. There is also need for CASE tools that can generate code depending on the programming method used. Rational Rose provides us with the tools for designing applications using the WAE-UML but the version (Rational Rose Enterprise Edition, Release Version 2002.05.20) that we used is not

capable of generating any code. Later versions might incorporate this functionality. During the practical work we also found improvements needed in the semantics in regard to the .NET architecture; opposite from how Conallen (2002) uses it. Correct and improved semantics is an important aspect to look at for the development of CASE tools.

The methods we investigated, W2000, WebML and WAE-UML, all approach Web application modelling and we take no stance at to what extent they are suitable. They provide good and less good tools and can be used for modelling a variety of Web applications. We could incorporate a method such as WebML, which provides means for personalization and content management, into the User Experience modelling with WAE-UML. In this way we can develop a more complete methodology and a total concept for Web application modelling. What WAE-UML lacks is the possibility for personalization and a better way of content management modelling. A complete method for Web application modelling, according to us, incorporates improved WAE-UML semantics and an extended User Experience modelling, perhaps with the help of WebML. In addition, a CASE tool must be developed for generating code automatically.

In future research it would also be interesting, and important, to measure at to what extent modelling with WAE-UML, and modelling Web applications at all, provides the project team with decreased development costs. The most interesting aspects are to measure the number of requirements and use-cases that are verified in the finished product before system testing. Consequently, it is needed to measure the amount of time spent correcting inconsistencies and errors in the final implementation during system testing. This in opposite to an application developed with specifications and text documents.

A third aspect is to measure how the knowledge of semantics among the people involved in a Web application project influences the finished product. The level of detail in the design documents is also important here as the ability to provide detail in a design is directly dependent on the semantic knowledge. This reflects both the understanding of the design artefacts and the ability to perform updates to them.

8 References

Baresi, Luciano, Garzotto, Franca & Paolini, Paolo (2001). *Extending UML for Modeling Web Applications*. Journal: Proceedings of the 34th Annual Hawaii International Conference on year: 2001. Pages: 1285-1294. Published by IEEE.

Bengtsson, Lars (2004). *Att arbeta med case*. Malmö: Liber Ekonomi. 91-47-04573-6.

Biddle, Robert, Noble, James & Tempero, Ewan (2002). *Towards Evaluation of Object-Oriented Designs.* [www]. Accessed from School of Mathematical and Computing Sciences, Victoria University, http://www.mcs.vuw.ac.nz/comp/Publications/archive/CS-TR-02/CS-TR-02-8.pdf. Accessed March 14, 2005.

Bosch, Jan (2000). *Design and use of Software Architectures – Adopting and evolving a product line approach*. Harlow: Addison-Wesley. ISBN: 0-201-67494-7.

Ceri, Stefano, Fraternelli, Piero & Bongio, Aldo (2000). *Web Modeling Language (WebML): a modeling language for designing Web sites*. Journal: Computer Networks, Vol: 33, issue: 1-6, pages: 137-157. Published by Proquest.

Conallen, Jim (1998). *Modeling Web Application Design with UML.* [www]. Accessed from http://www.itmweb.com/essay546.htm. Accessed February 6, 2005.

Conallen, Jim (1999). *UML Extension for Web Applications 0.91.* [www]. Accessed from http://www.conallen.com/technologyCorner/webextension/WebExtension091.htm. Accessed February 6, 2005.

Conallen, Jim (2002). *Building Web Applications with UML Second Edition*. Boston: Addison-Wesley, 2nd edition. 0-201-73038-3.

Ely, Margot, et. al. (1993). *Kvalitativ forskningsmetodik i praktiken – cirklar inom cirklar*. Lund: Studentlitteratur. 91-44-37111-X.

Freedman, Daniel & Weinberg, Gerald (1990). *Handbook of Walkthroughs, Inspections and Technical Reviews – Evaluating Programs, Projects and Products*. Boston: Dorset House. 3rd edition. 0-932633-19-6.

IEEE Computer Society (1998). *IEEE Standard for Software Verification and Validation*. Journal: IEEE Std., 1012-1998. Published by IEEE.

Jacobson, Ivar, Booch, Grady & Rumbaugh, James (1998). *The Unified Modeling Language Reference Manual*. Reading: Addison-Wesley. 1st edition. 0-201-30998-X.

Larman, Craig (2001). *Applying UML and Patterns; An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Upper Saddle River: Prentice Hall. 0-13-092569-1.

Martella, Ronald, Nelson, Ronald & Marchand-Martella, Nancy (2003). *Research Methods – Learning to become a critical research consumer*. Needham Heights: Allyn & Bacon. 0-205-27125-1.

OMG, Object Management Group (2001). *OMG Unified Modeling Language Specification.* [www]. Accessed from http://www.omg.org/technology/documents/formal/uml.htm. Accessed at February 8, 2005.

Robson, Colin (2002). *Real world research: a resource for social scientists and practitioner-researchers*. Oxford: Blackwell publishing. 2nd edition. 0-631-21305-8.

Sommerville, Ian (2001). *Software Engineering*. Essex: Addison Wesley. 6th edition. 0-201-39815-X.

Staroń, Miroslaw (2003). *Customizing UML with Stereotypes*. Karlskrona: Blekinge Institute of Technology. 91-7295-028-5. Master thesis.

Strand, Lotta (2003). *UML & RUP – Att lyckas med OO-projekt*. Sundbyberg: Docendo. 91-7882-587-3.

Swedish Research Council (2002). *Forskningsetiska principer inom humanistisk-samhällsvetenskaplig forskning.* [www]. Accessed from http://www.vr.se/publikationer/sida.jsp?resourceId=12. Accessed April 11, 2005.

Trung Thanh Dinh Trong (2003). *A Systematic Procedure for Testing UML Designs.* [www]. Accessed from http://www.cs.colostate.edu/~ghosh/papers/issre2003trung.pdf. Accessed March 17, 2005.

9 Appendix one – Use-cases

9.1 Place order

Actors: Customer
Goal: To place an order using the e-commerce system.
Name: Place order

9.1.1 Flow of Events

Basic Flow

1. The customer clicks on the ”Place order” menu item.
2. The customer enters the article number.
3. The system displays the name and price of the article.
4. The customer enters the quantity.
5. The system displays the sum of the articles.
6. The customer chooses a delivery date for the article.
7. The customer presses the “Ok” button.
8. The system adds the order row to the order table below the input fields.
9. The customer repeats the steps 2-8 for all the articles he wants.
10. The customer clicks on the “Confirm order” menu item.
11. The system displays the order table to the customer.
12. The customer enters his password and presses the “Confirm” button.
13. The system updates the database tables with the new order.
14. The system sends the order to XAL.
15. The system displays to the customer that the order is sent.

Alternative Flows

9.1 The customer clicks on the “Edit” link that corresponds to the article he added in the order table.
9.2 The system deletes the order row from the order table and displays the values of it in the input fields for a new order row.
9.3 The customer enters new values for the article number, quantity or delivery date.
9.4 The customer clicks on the “Ok” button.

10.1 The customer enters the wrong password.
10.2 The customer clicks on the “Confirm” button.
10.3 The system displays a dialog box saying that the password was incorrect.

9.1.2 Pre-conditions

The customer must be logged in to the e-commerce website.

9.1.3 Post-conditions

An order that contains the articles chosen by the customer is sent to the customer administration. The order table is cleared and if the customer clicks on the "Place order" menu item a new order is started.

9.2 Article search

Actors: Customer
Goal: To find an article and add it to an order.
Name: Article search

9.2.1 Flow of Events

Basic Flow

1. The customer clicks on the "Article search" menu item.
2. The system displays the Article search view.
3. The customer chooses to search for an article number.
4. The customer enters a part of or the whole article number in the search field.
5. The customer clicks on the "Search" button.
6. The system searches the database for articles that match the article number entered by the customer.
7. The system displays the search result below the search field.
8. The customer enters the amount of an item that is displayed by the search.
9. The customer clicks on the "+" button next to the article found.
10. The system adds the article to the order.
11. The customer clicks on the "Place order" menu item.
12. The article the customer searched for is displayed in the order table.

Alternative Flows

3.1 The customer chooses to search for an article name.
3.2 The customer enters the name of the article or part of the article name.
3.3 The system searches the database for articles that match or starts with the name entered by the customer.
3.4 Normal flow 7.

9.2.2 Pre-conditions

The customer must be logged in to the e-commerce website.

9.2.3 Post-conditions

The searched for article is displayed in the order table and is present in the database.

10 Appendix two - Requirements

10.1 Place order

10.1.1 Order rows

The customer should be able to add and delete order rows to the order.

10.1.2 Credit limit

The customer should be notified if he exceeds his credit limit when adding articles to the order. A message should suggest the customer to contact the customer service.

10.1.3 Order row content

The order row should contain article number field, quantity, status, delivery date, designation, price, and row sum.

10.1.4 Order row content filled in by the customer

The article field and quantity field is filled in by the customer. The delivery date should be the default delivery date from the order head view. The customer can change the date for every article. The remaining fields are generated from the database and should not be editable by the customer.

10.1.5 Change order rows

For each row in the order table there should be a hyperlink where upon the user clicks he can edit the specific order row.

10.1.6 Attach a message to the order row

The customer must be able to attach a message to each order row in the order table.

10.1.7 Navigating in the place order view

The customer should be able to navigate the different fields in an order row by pressing the tab-key and he should confirm the order row by pressing the enter key.

10.1.8 Deleting order rows

It must be possible for the customer to delete order rows that has been added to an order.

10.2 Confirm order

10.2.1 Send note along with the order

The customer should be able to attach a message for the customer service when sending an order.

10.2.2 Confirming sent order

The e-commerce system should notify the customer that the order was sent to the customer service after the customer has sent the order. He should come to a new view on the e-commerce site that says the order has been sent to the customer service.

10.2.3 Confirm orders with passwords

To confirm an order the customer must enter his password.

10.2.4 Sending order

The order should be sent to a map in the business system repository as a text file and handled manually by the customer service. This means that when placing an order the SQL server and e-commerce system are not directly communicating with the business system XAL.

10.2.5 Price display

On the confirm order view the VAT, price exclusive VAT and the users discount should be presented.

10.2.6 Credit limit

All customers using the e-commerce system have a credit limit and they should not be able to confirm an order that exceeds the limit.

10.3 Article search

10.3.1 Search for articles using article number

The customer must be able to search for articles using the article number. The search should be performed as front-end search. This means that if the customer enters a “123” in the search field all articles that starts with the sequence “123” should be found.

10.3.2 Search for articles using the article name

The customers of the e-commerce site must be able to search for an article using the name or part of the name for an article.

10.3.3 Order articles from the article search

The customer must be able to place the articles he searched for in order rows in the current order he has started. If he has not yet started to place an order it should not be possible to put the articles in any order.

10.3.4 Notification of articles added to the order

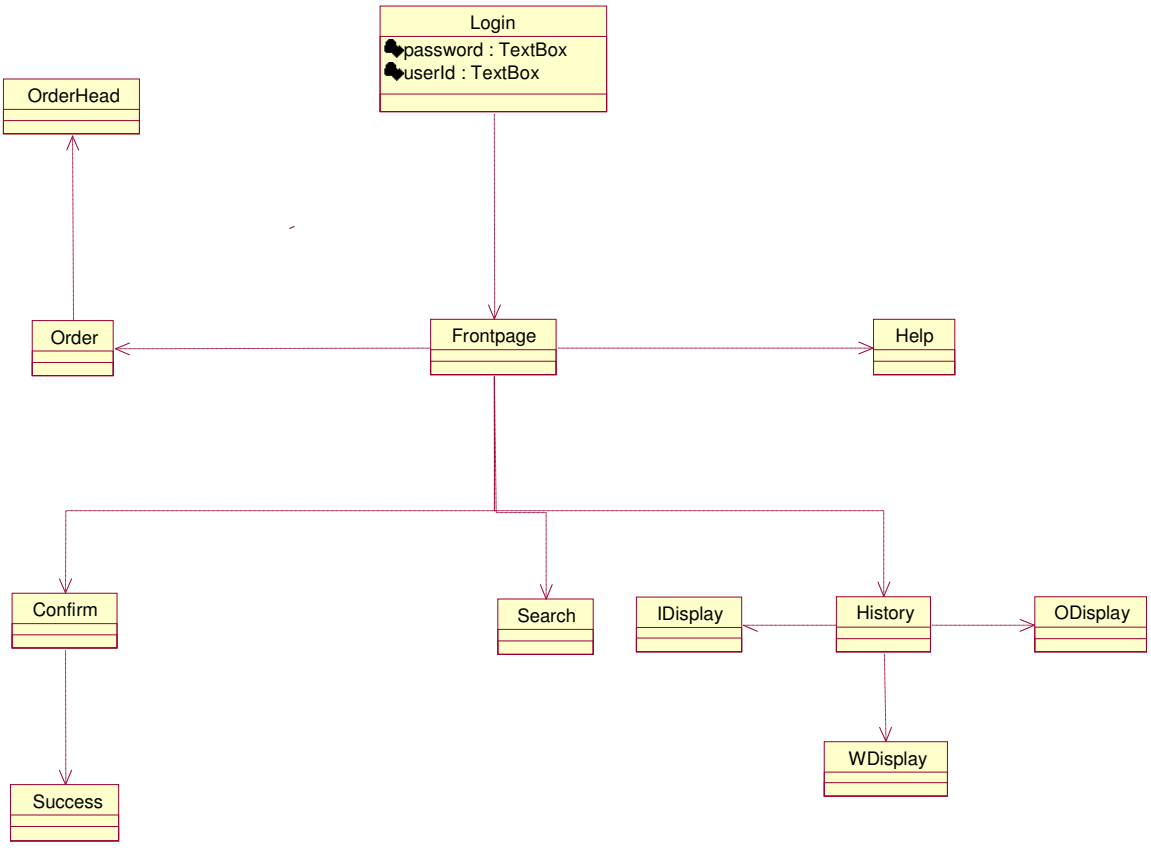
The customer should be notified with a dialog box when an article is added to the order.

11 Appendix three – UML and WAE-UML designs

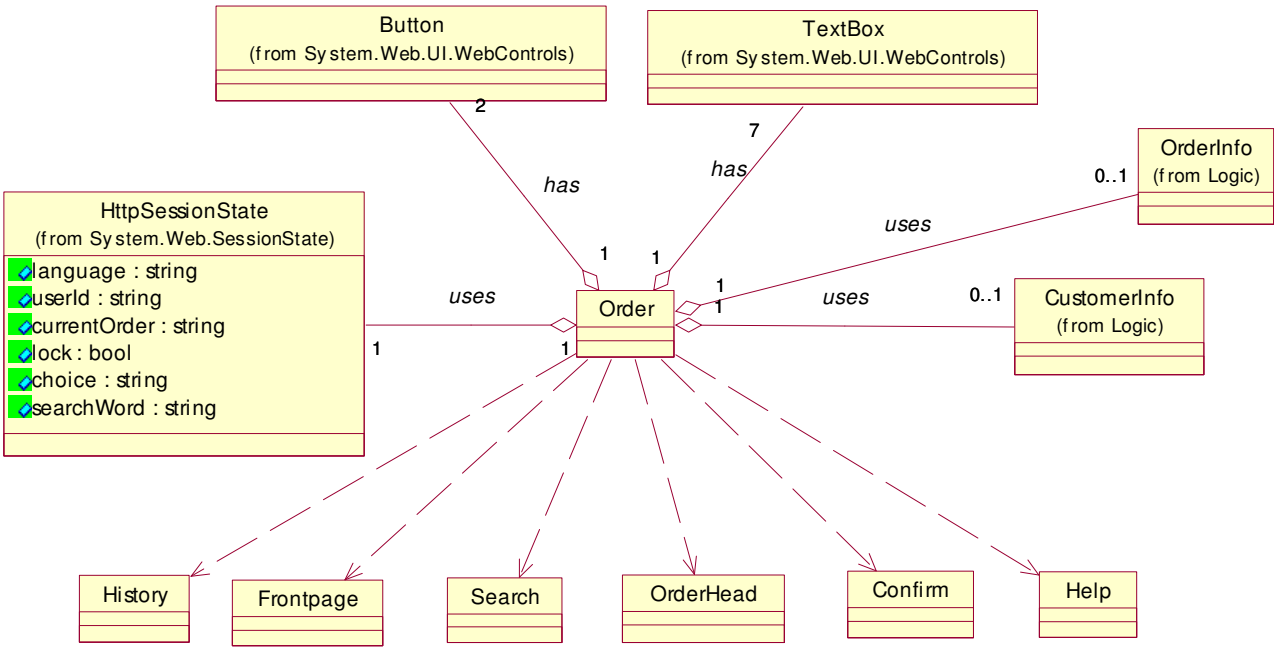
The diagrams in this appendix, both from the old design and the redesign, constitute the important diagrams only. With important we mean the final design documents, class diagrams, collaboration- and sequence diagrams, and in the case of the redesigned version a UX navigational map.

11.1 Old e-commerce design - UML

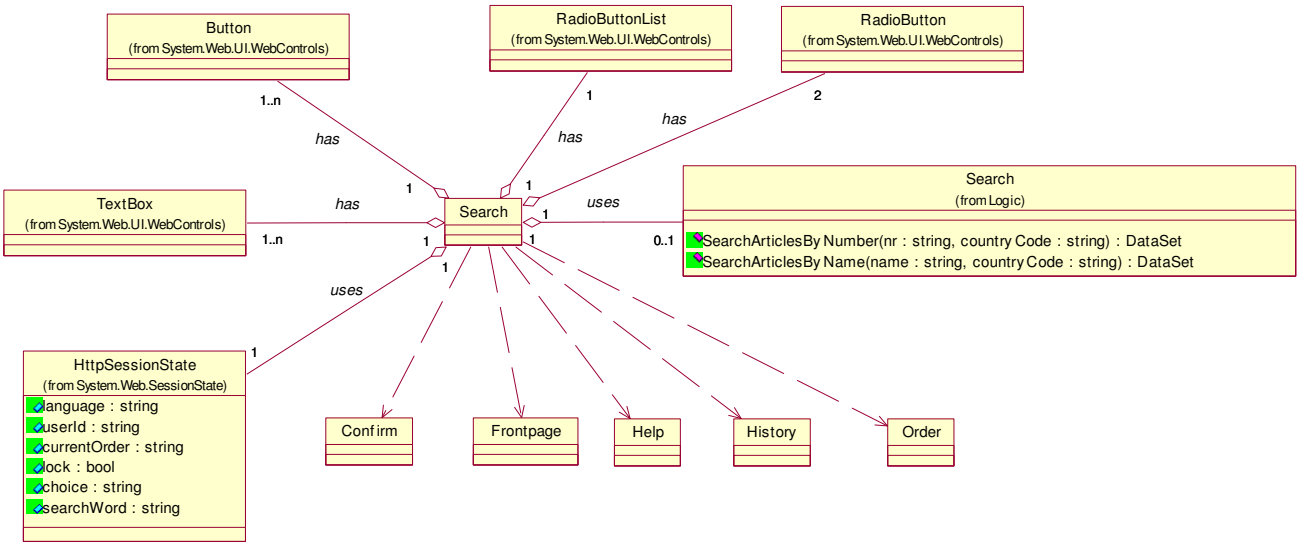
11.1.1 UML navigation map



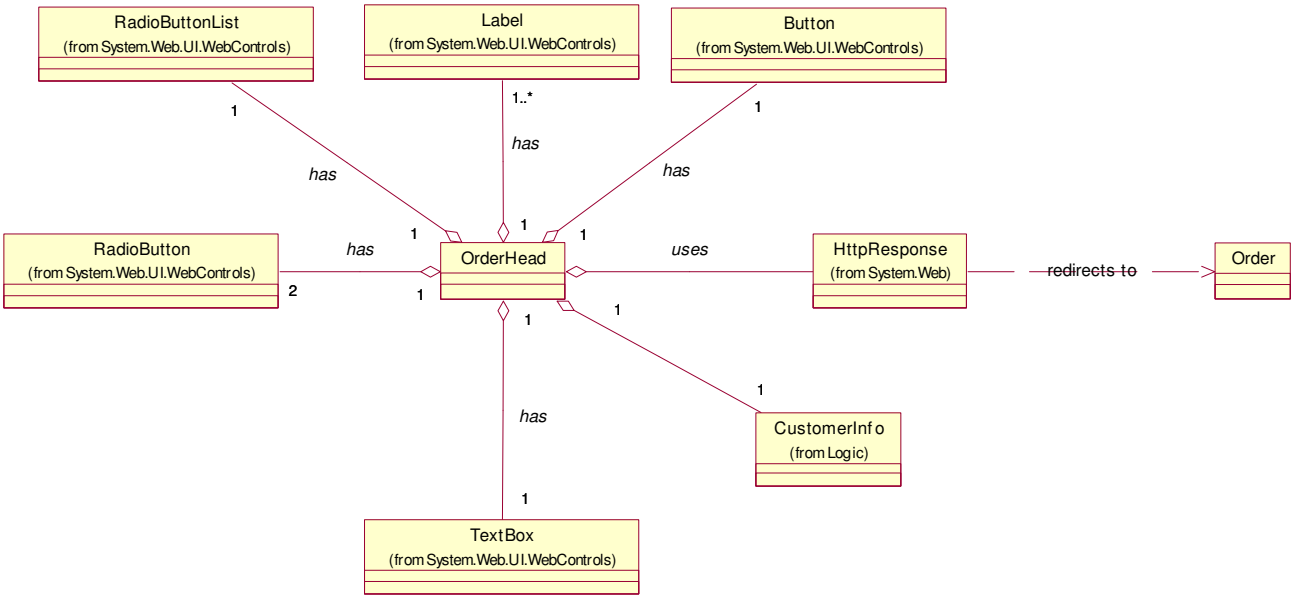
11.1.2 UML class diagram – Place Order



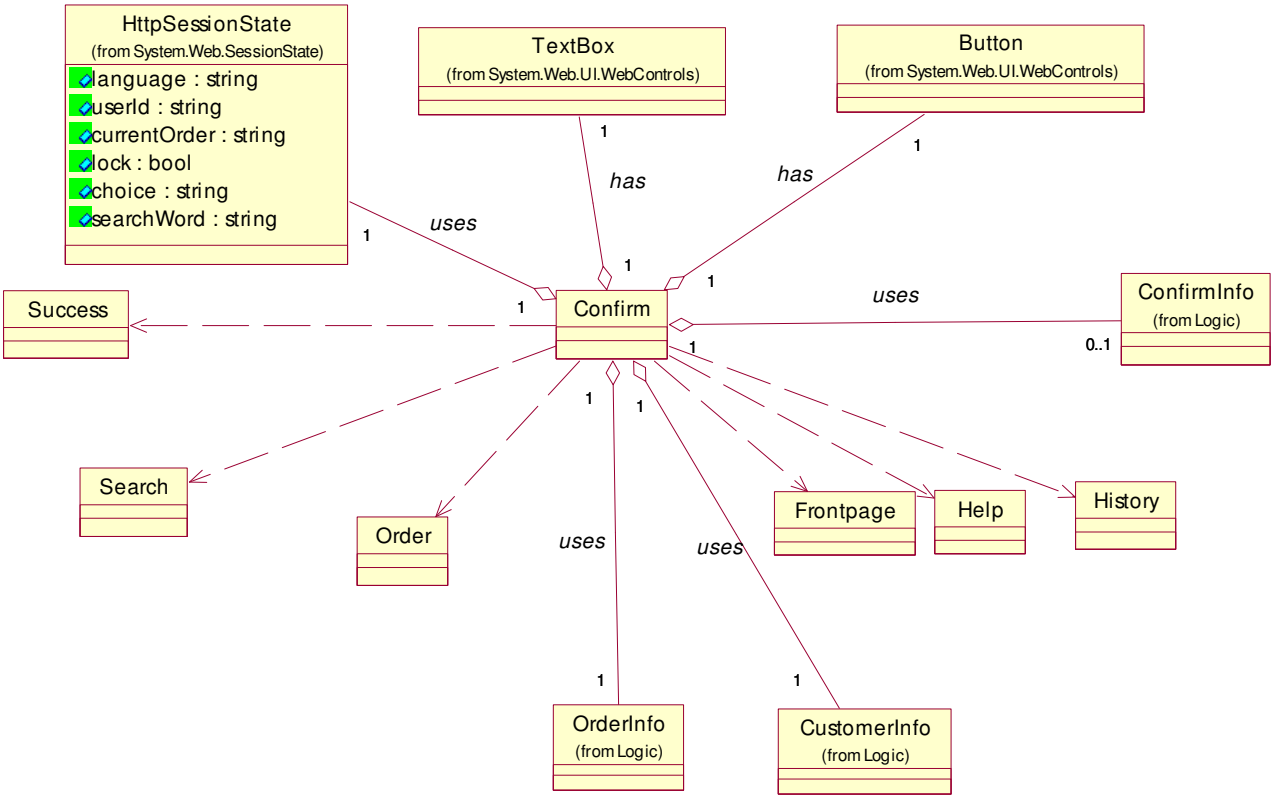
11.1.3 UML class diagram – Article search



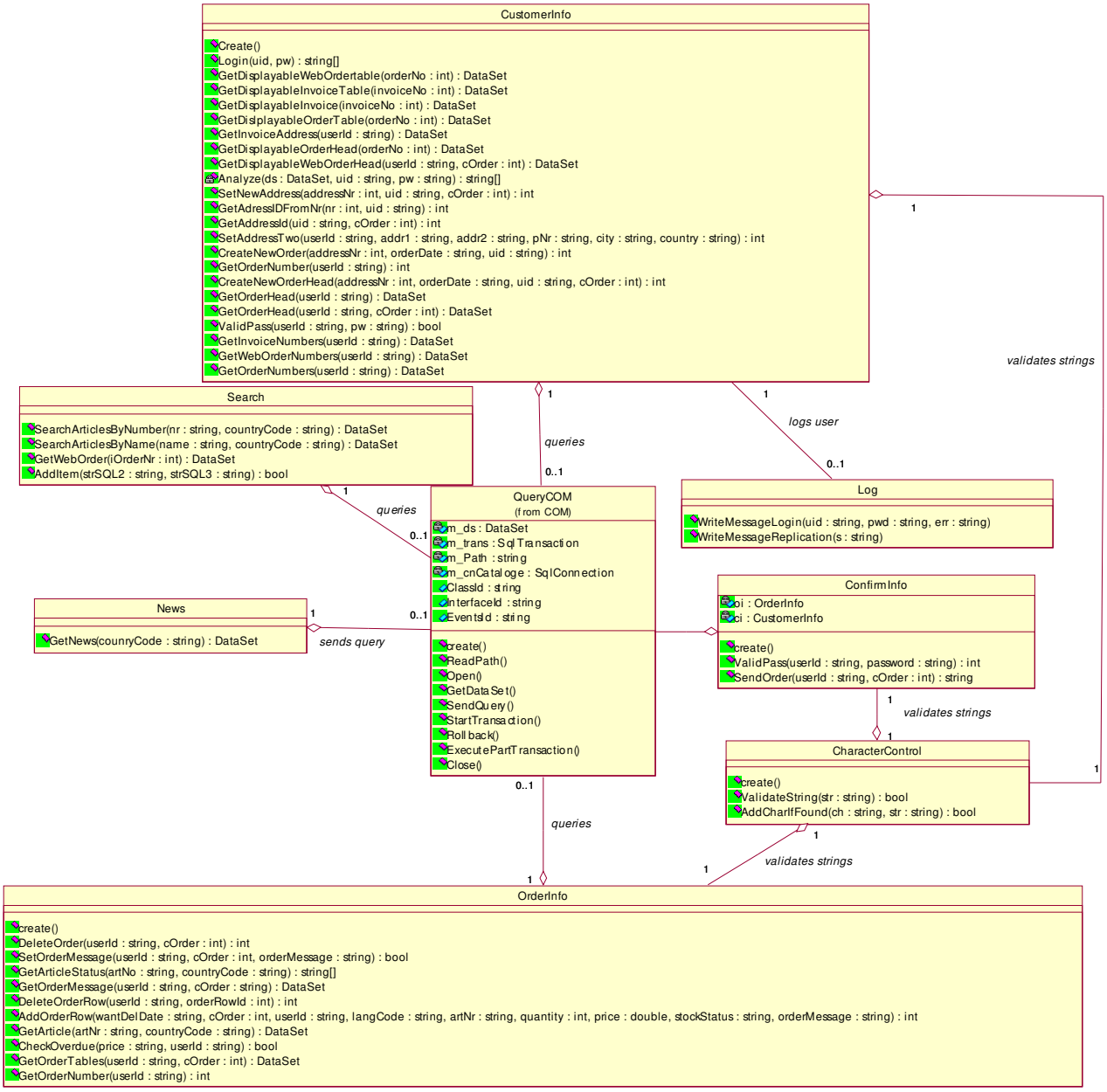
11.1.4 UML class diagram – Create order



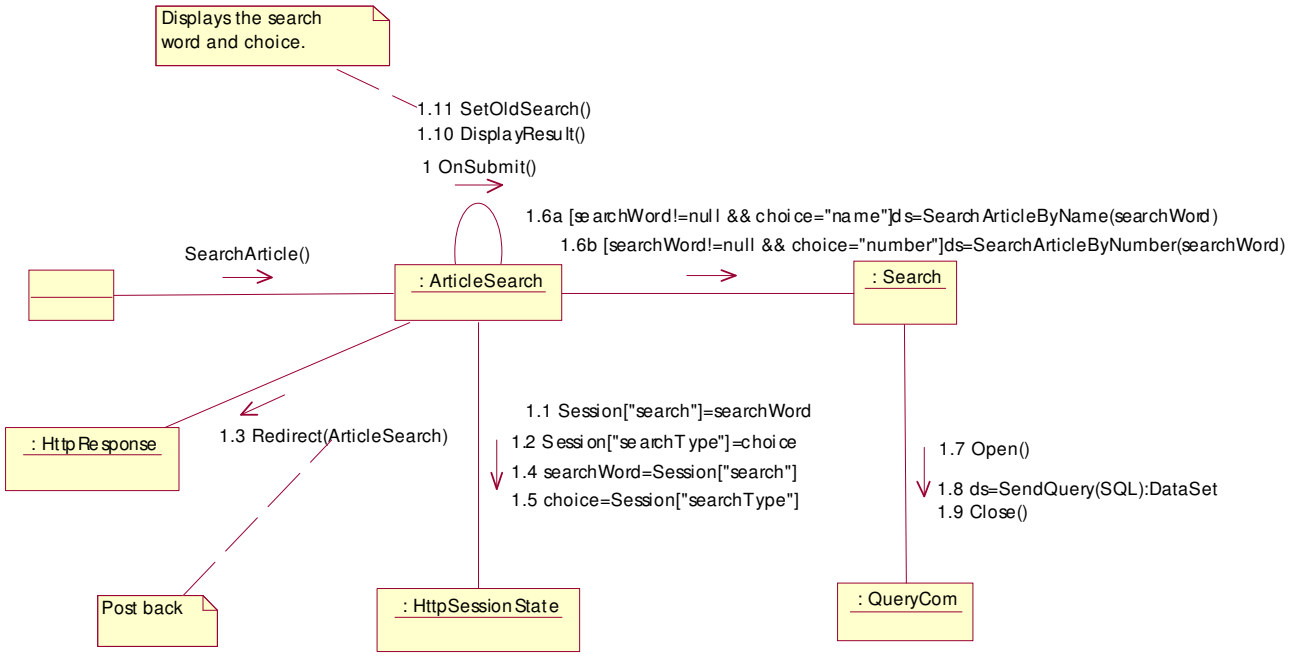
11.1.5 UML class diagram – Confirm order



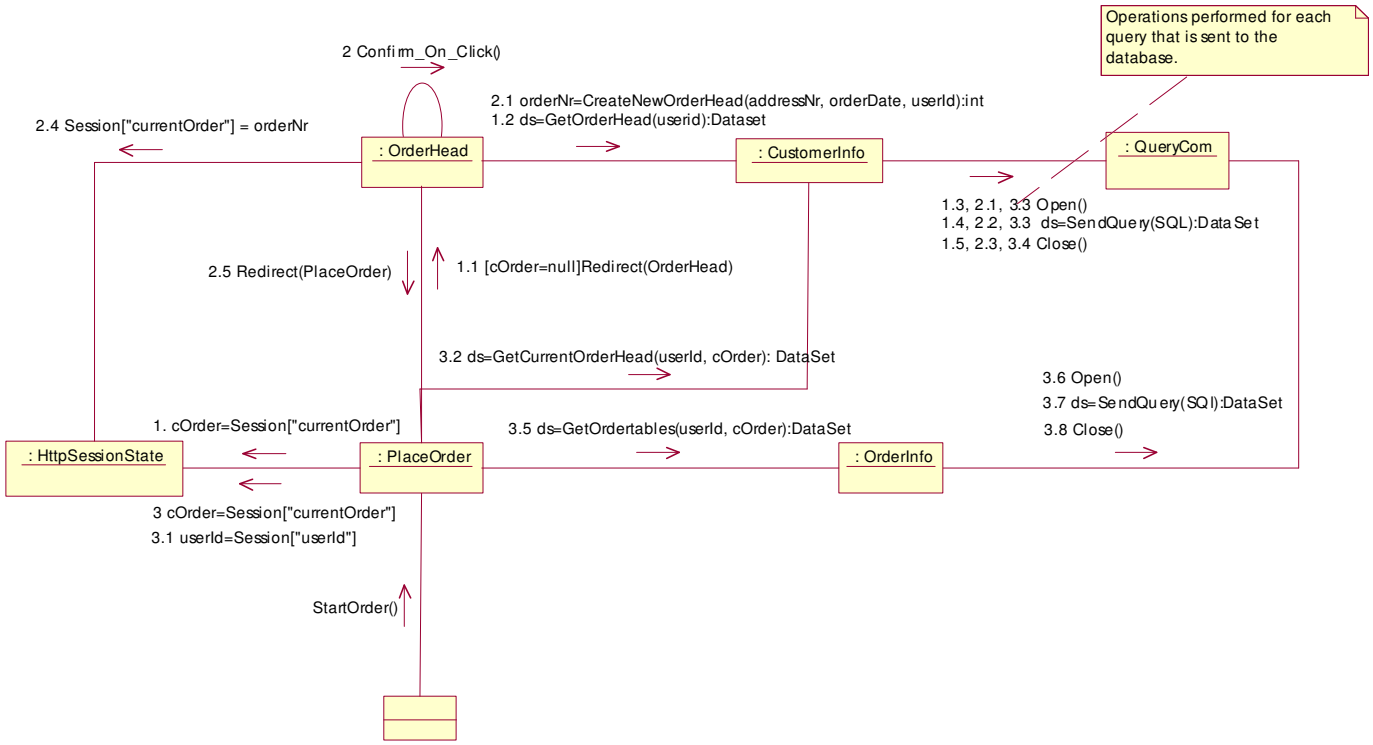
11.1.6 UML class diagram – Logic package



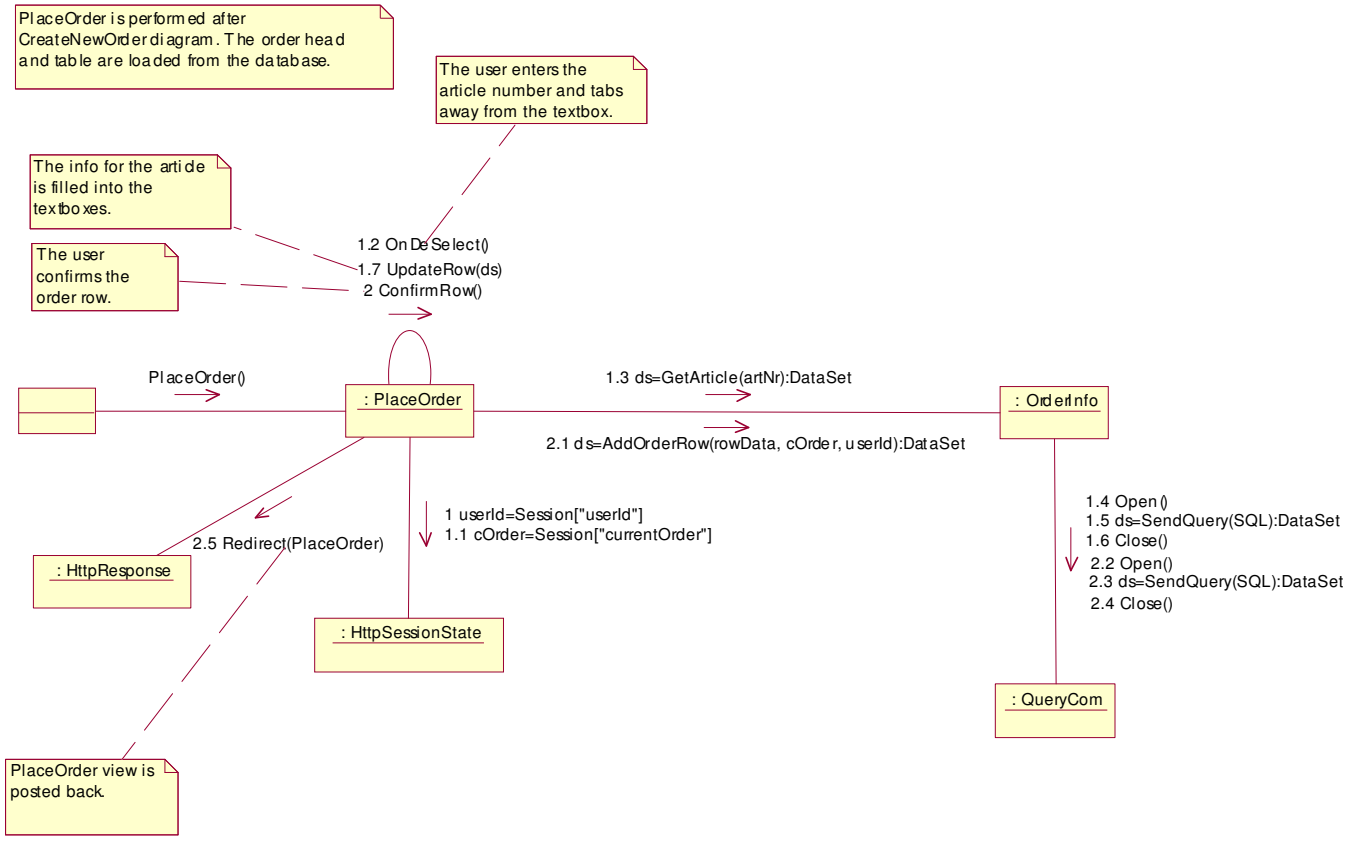
11.1.7 UML collaboration diagram – Article search



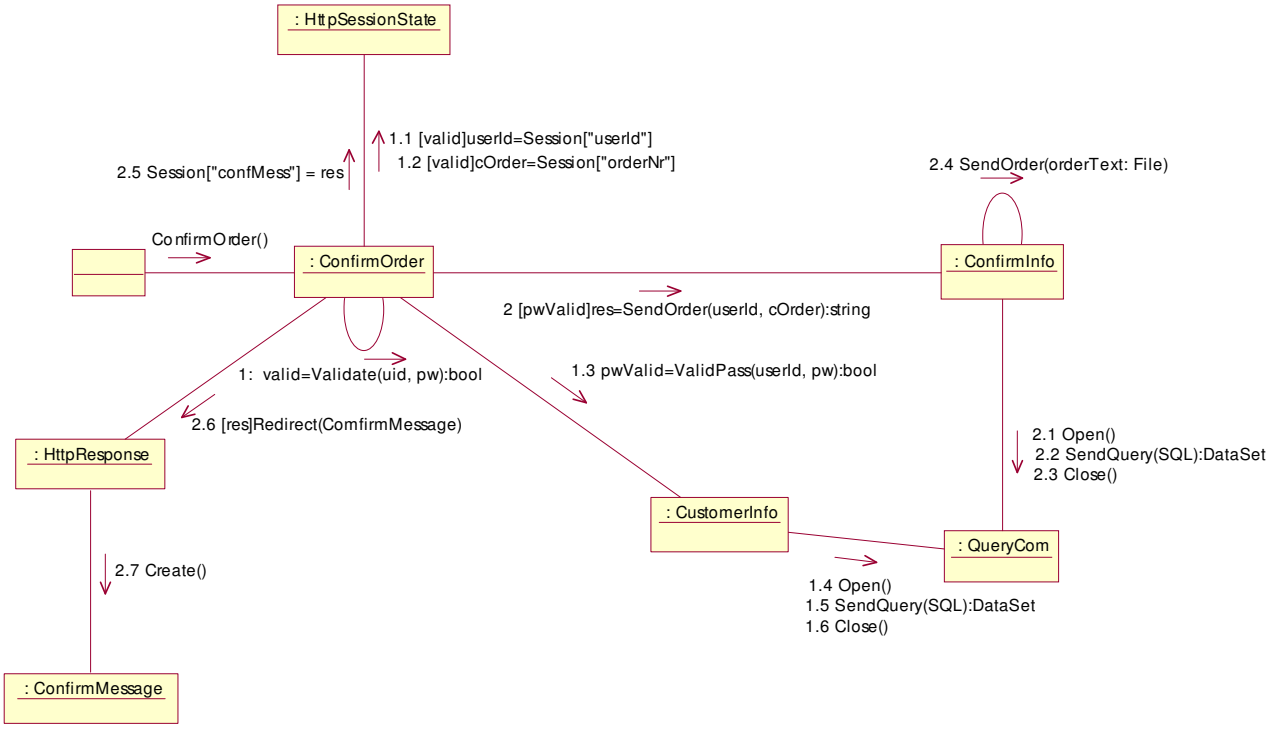
11.1.8 UML collaboration diagram – Create new order



11.1.9 UML collaboration diagram – Place order

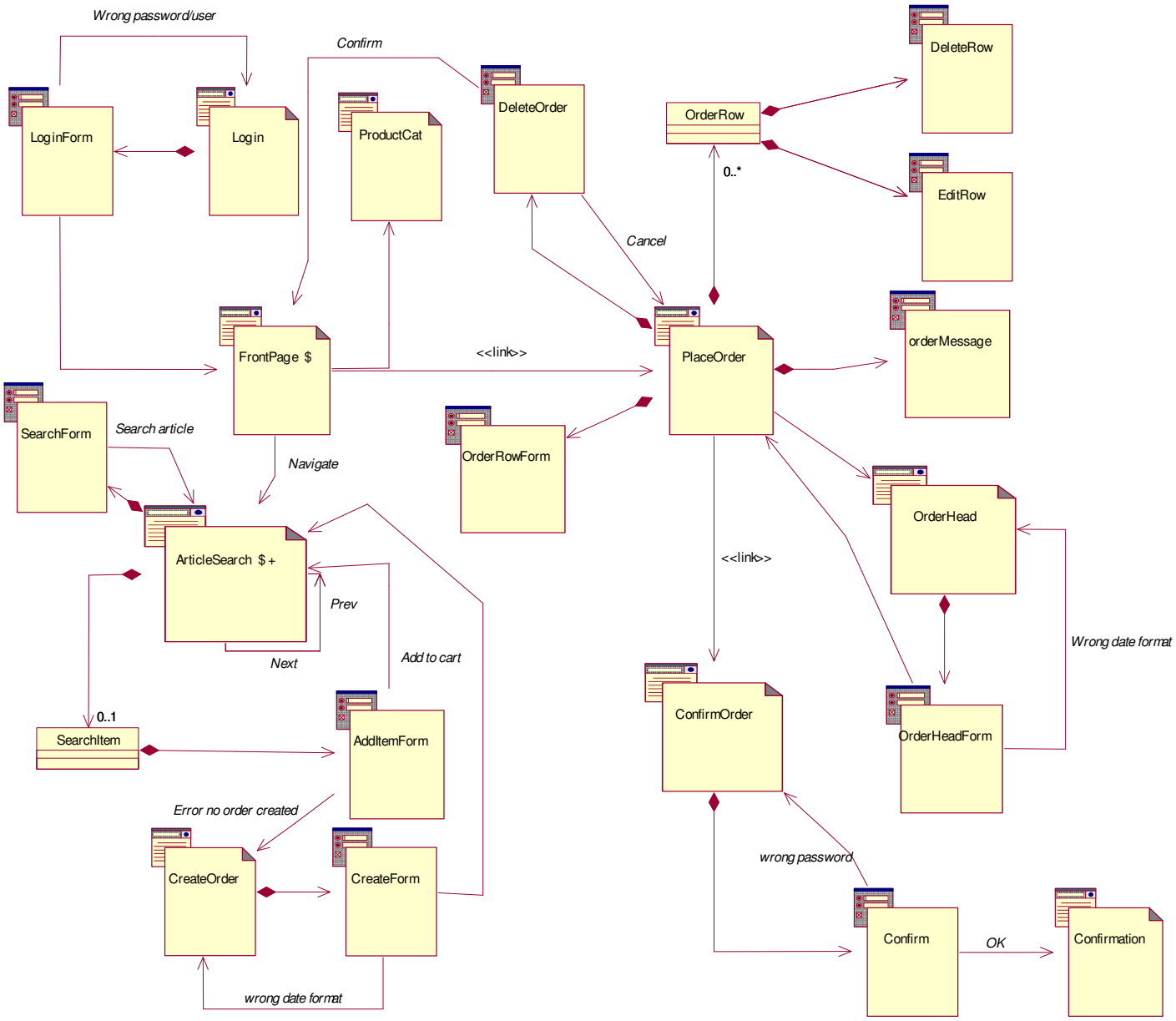


11.1.10 UML collaboration diagram – Confirm order

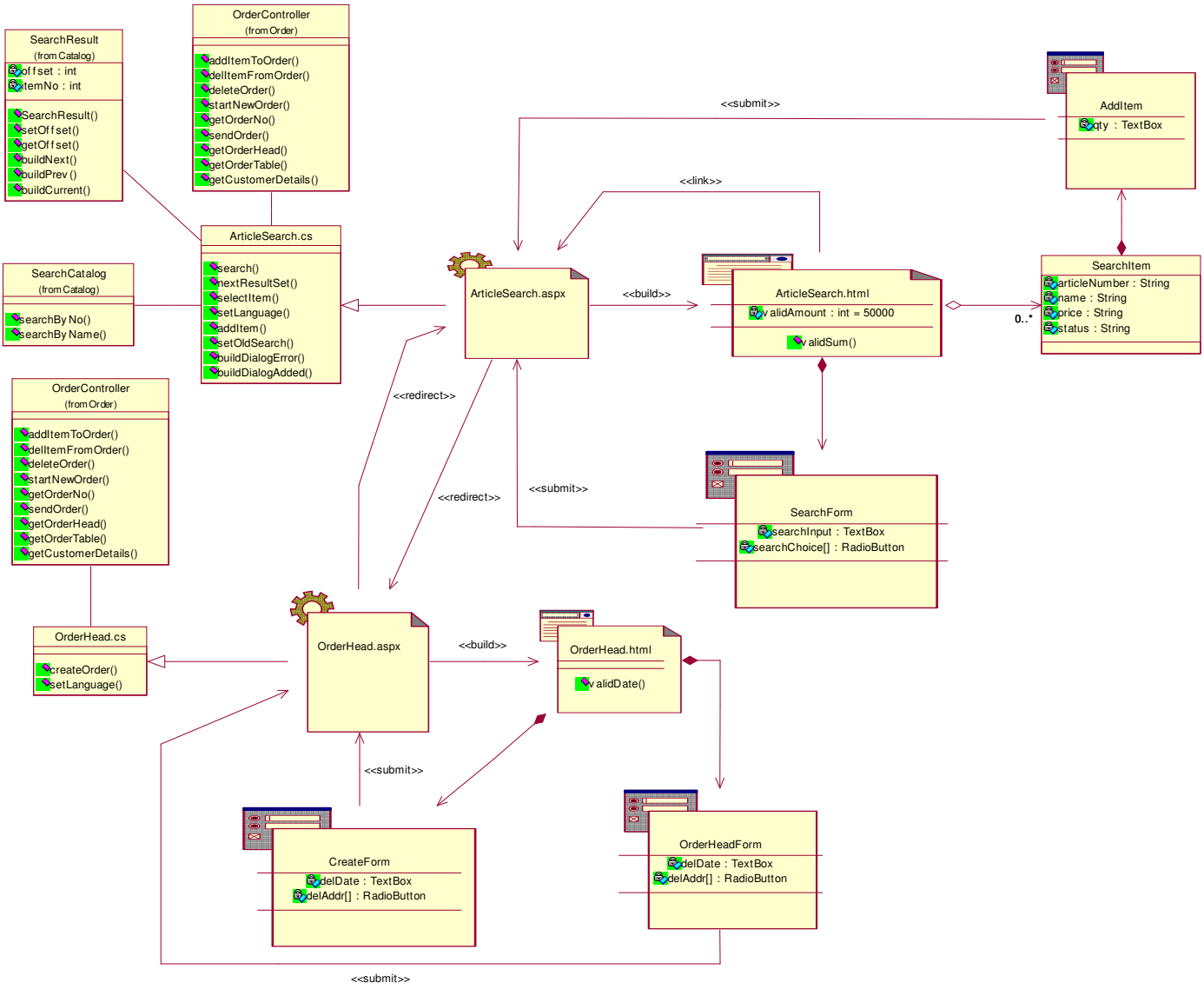


11.2 E-commerce redesign with WAE-UML

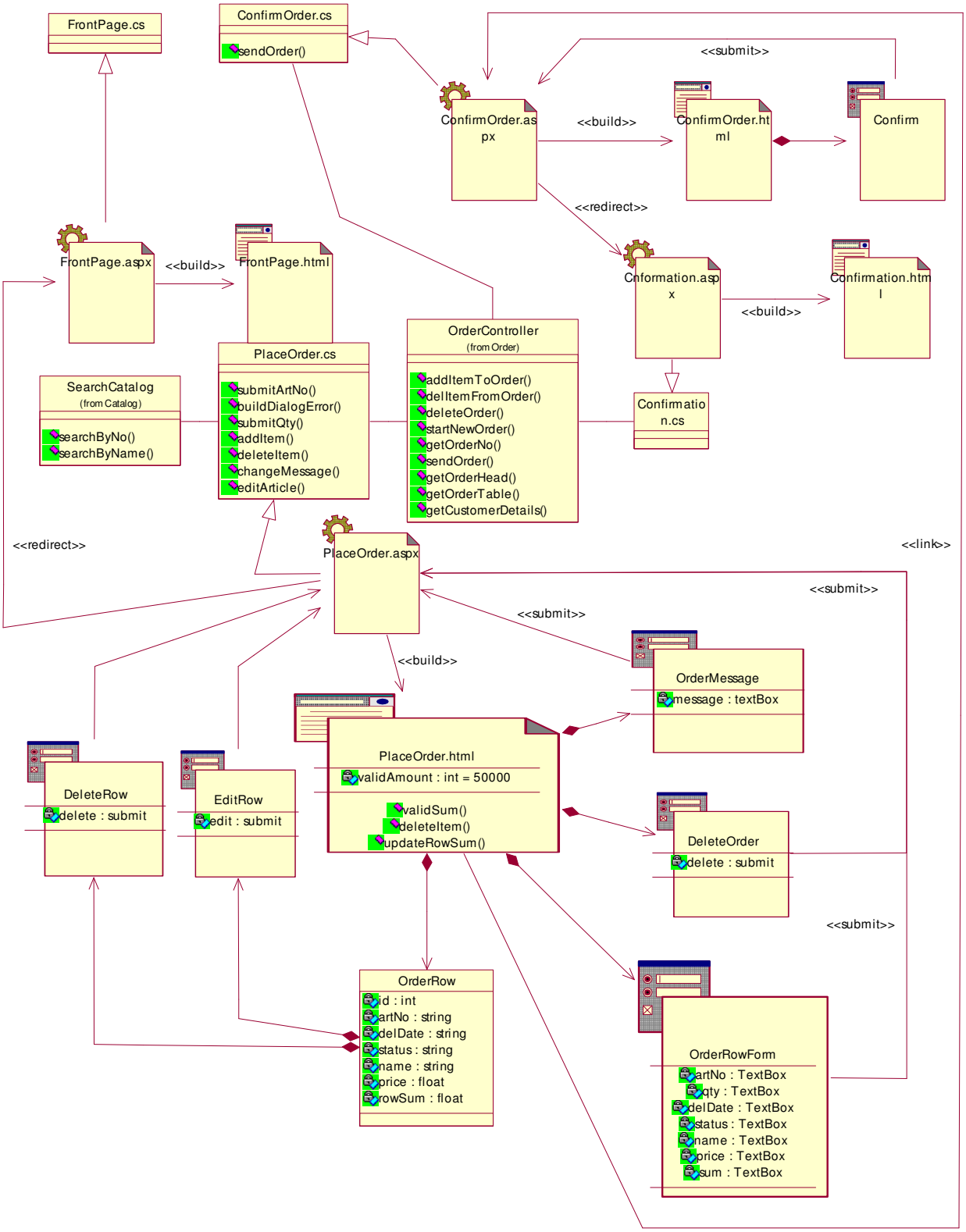
11.2.1 WAE-UML user Experience – Navigational map



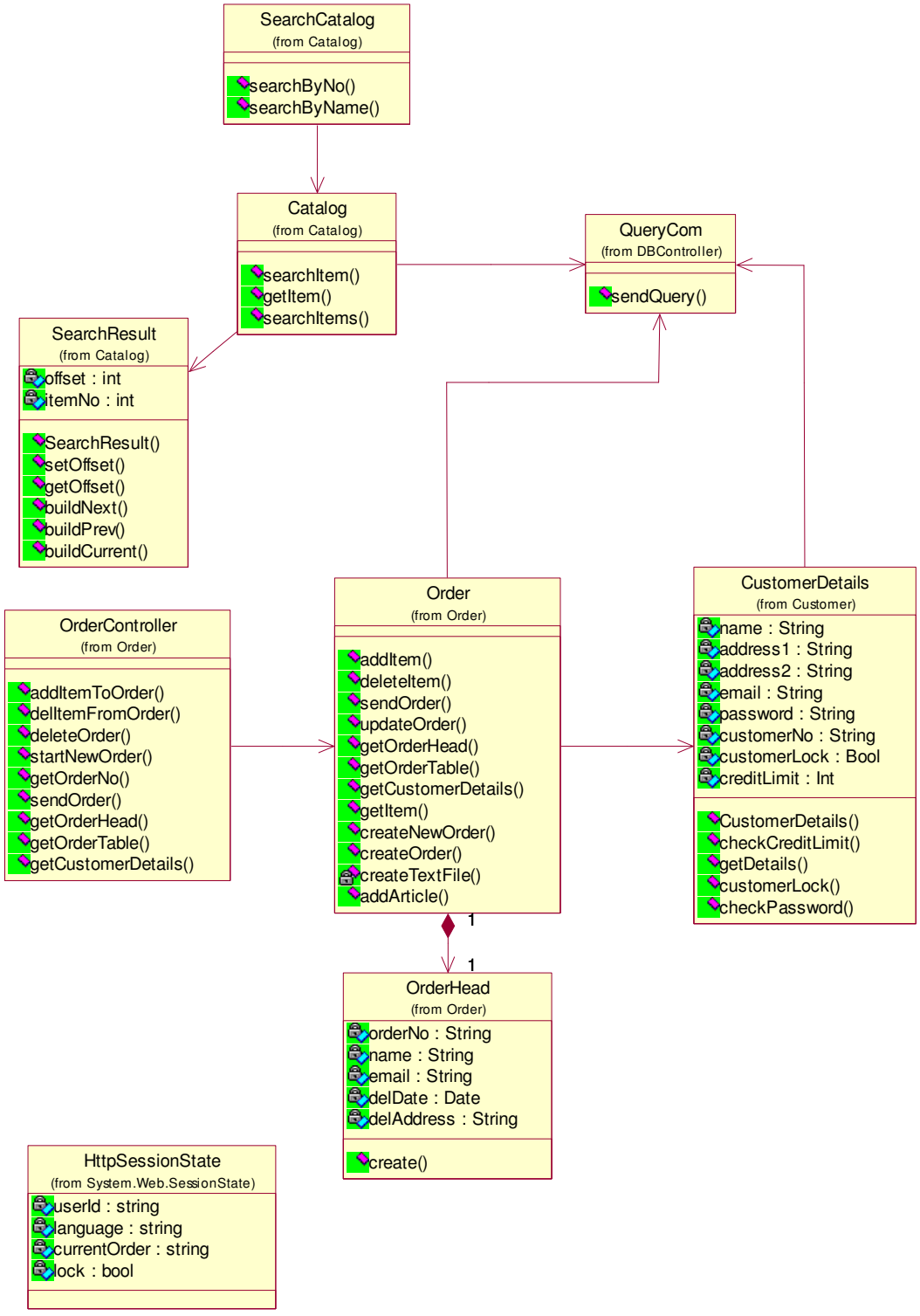
11.2.2 WAE-UML class diagram – Article search



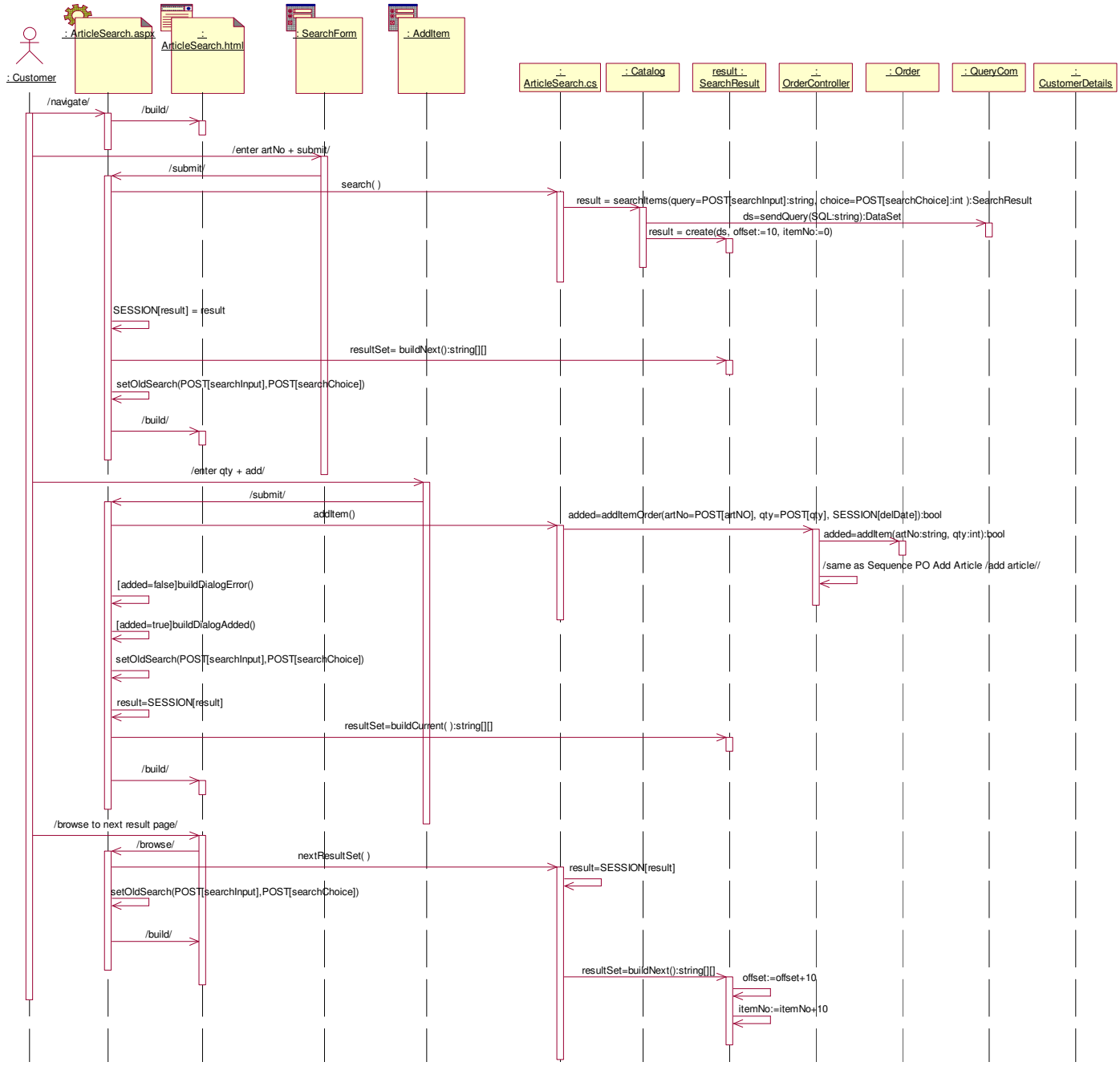
11.2.3 WAE-UML class diagram – Place order



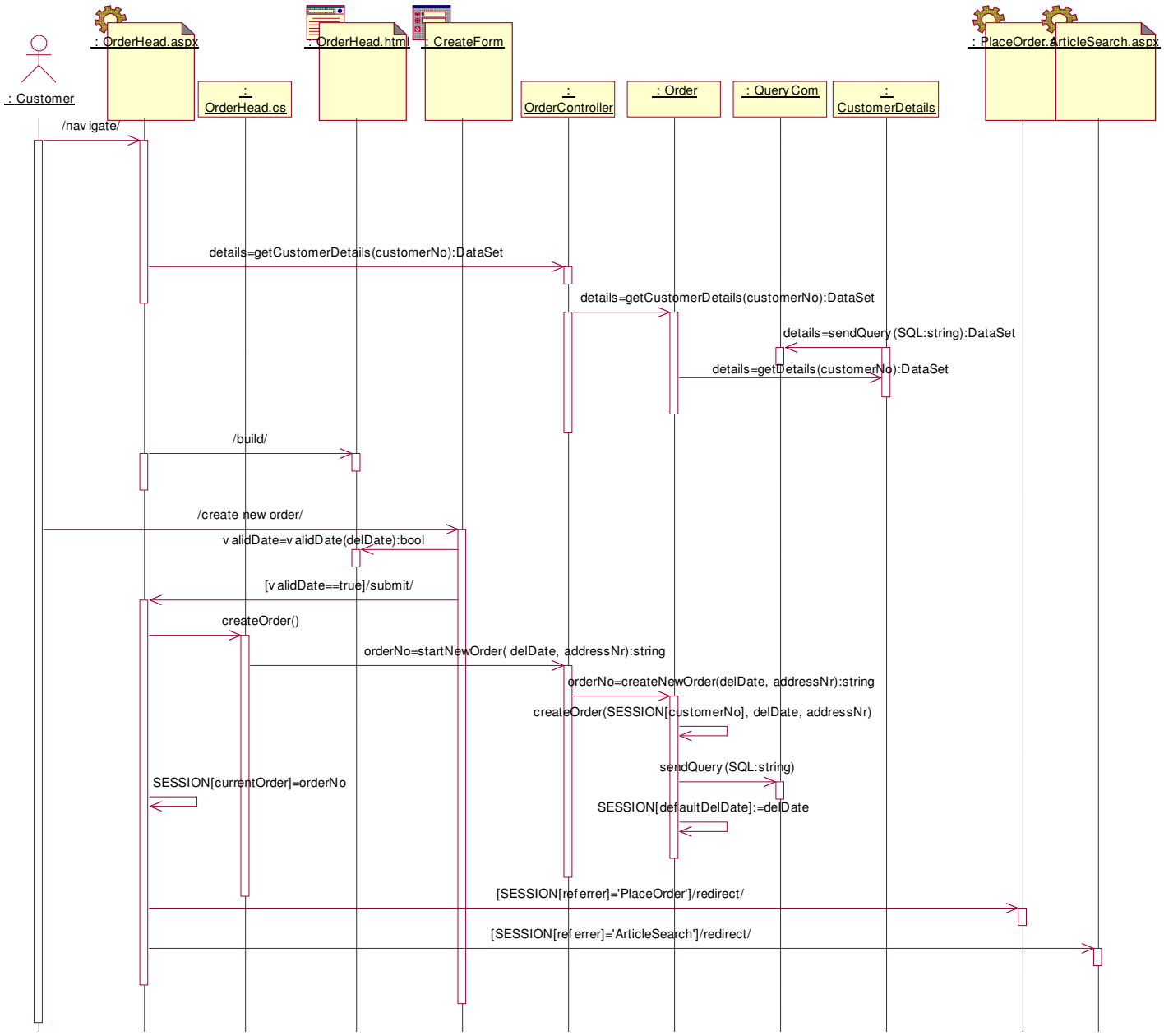
11.2.4 WAE-UML class diagram – Logic package



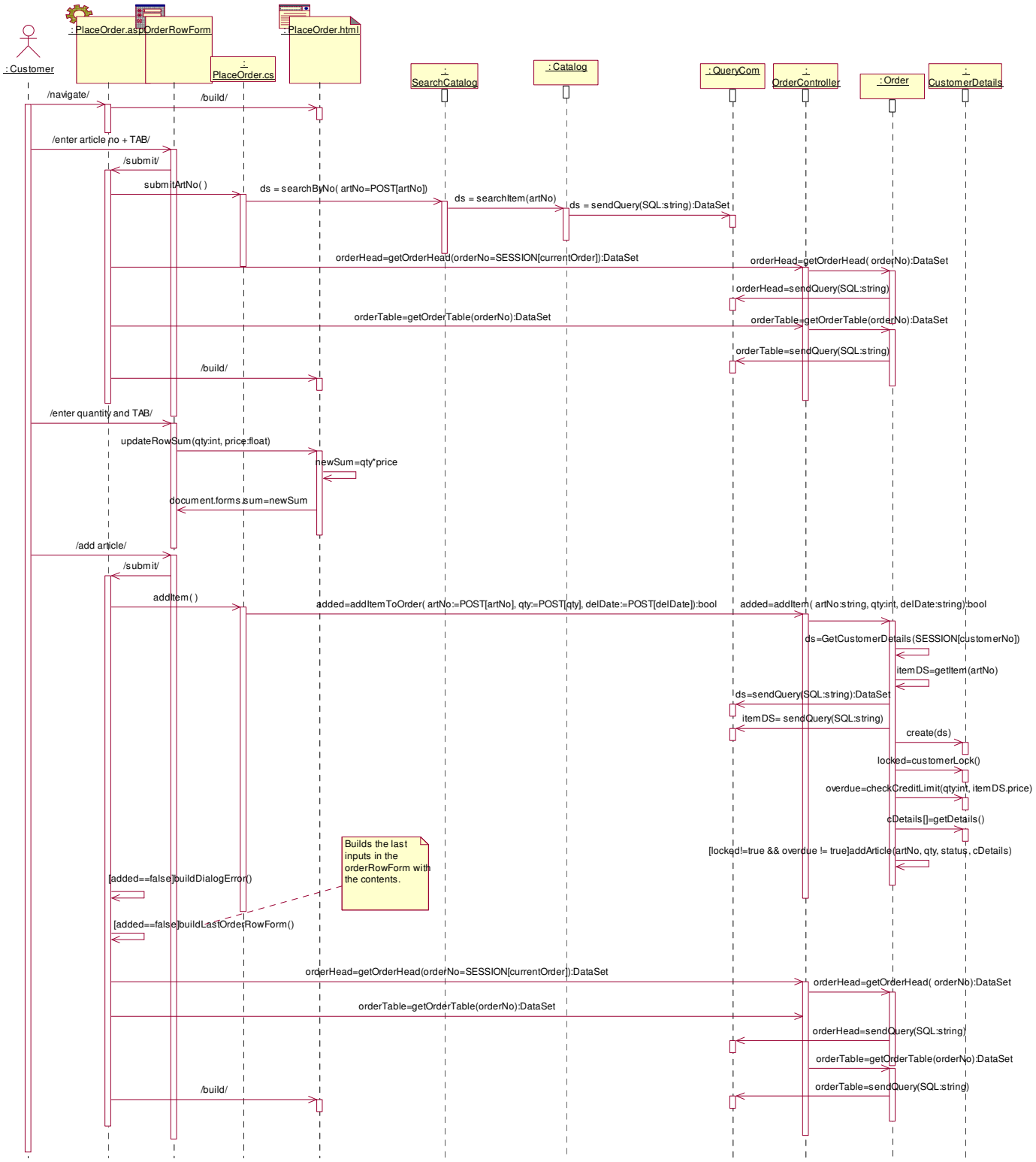
11.2.5 WAE-UML sequence diagram – Article search



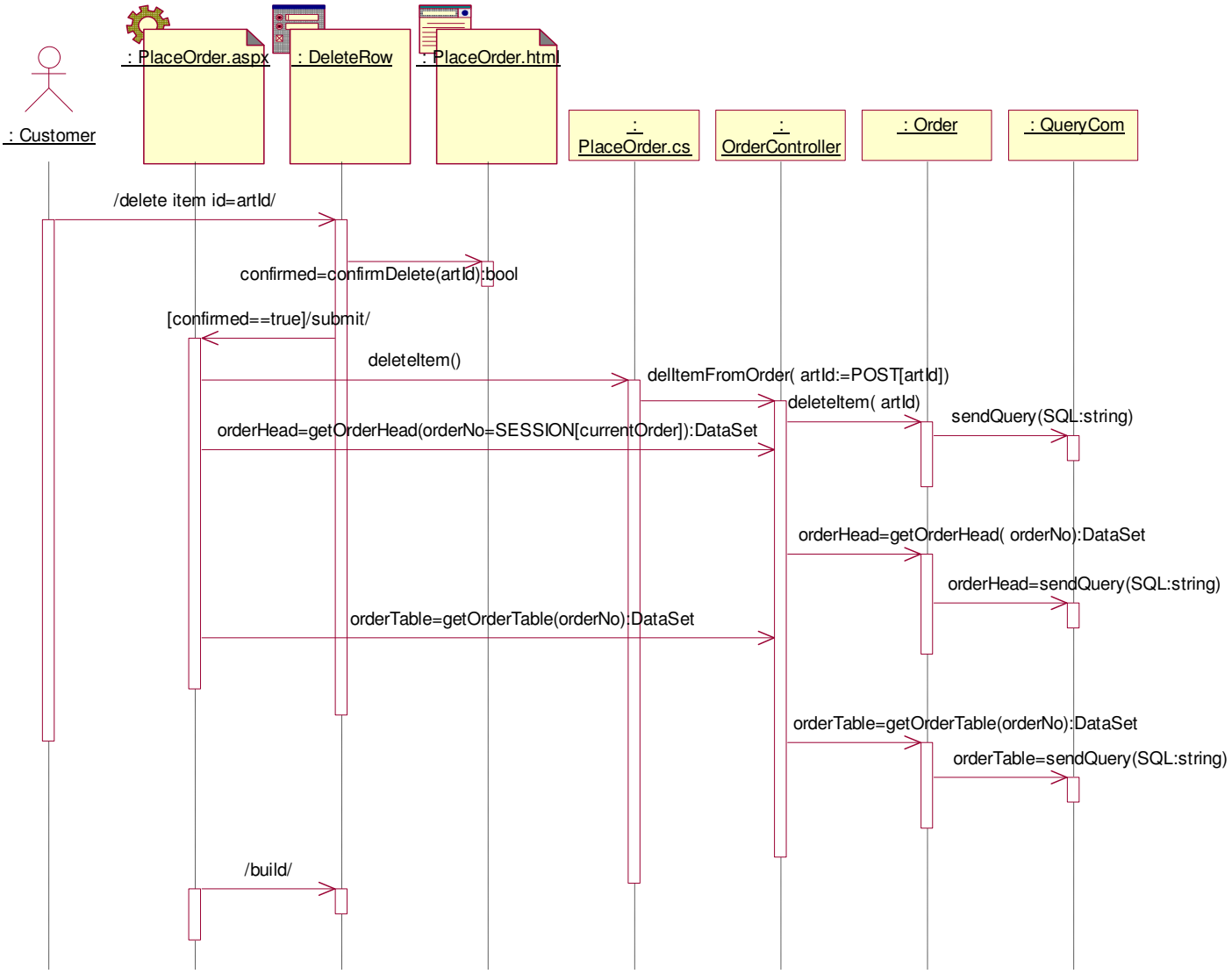
11.2.6 WAE-UML sequence diagram – Create order



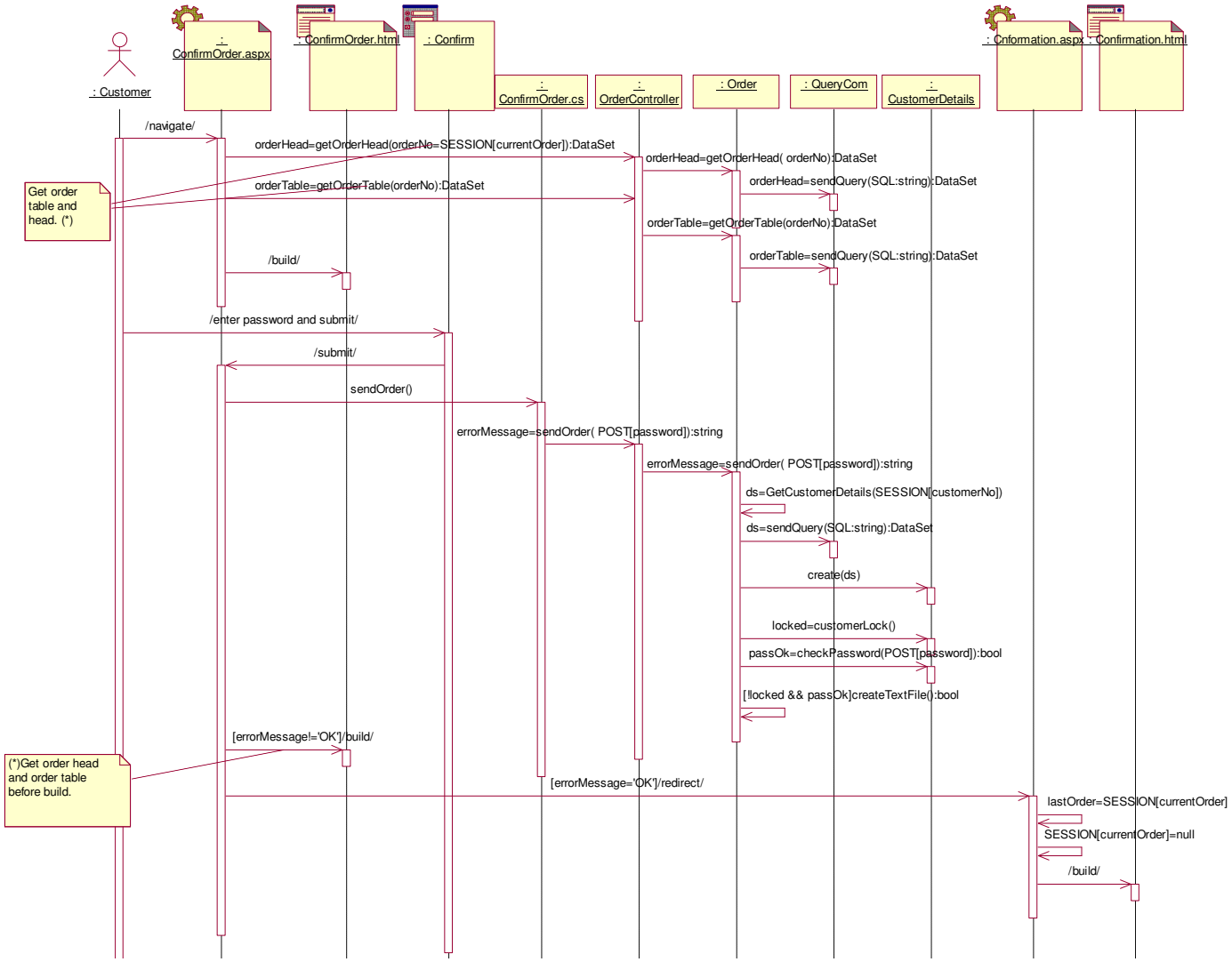
11.2.7 WAE-UML sequence diagram – Add article



11.2.8 WAE-UML sequence diagram – Delete item

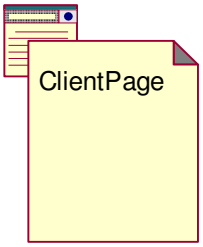


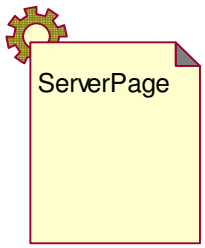
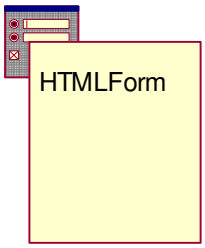
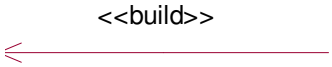
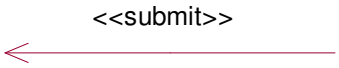
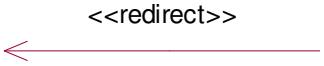
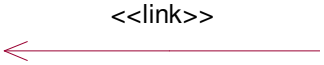
11.2.9 WAE-UML sequence diagram – Confirm order



12 Appendix four – WAE-UML notation

These stereotypes and their definitions come from Conallen (2002).

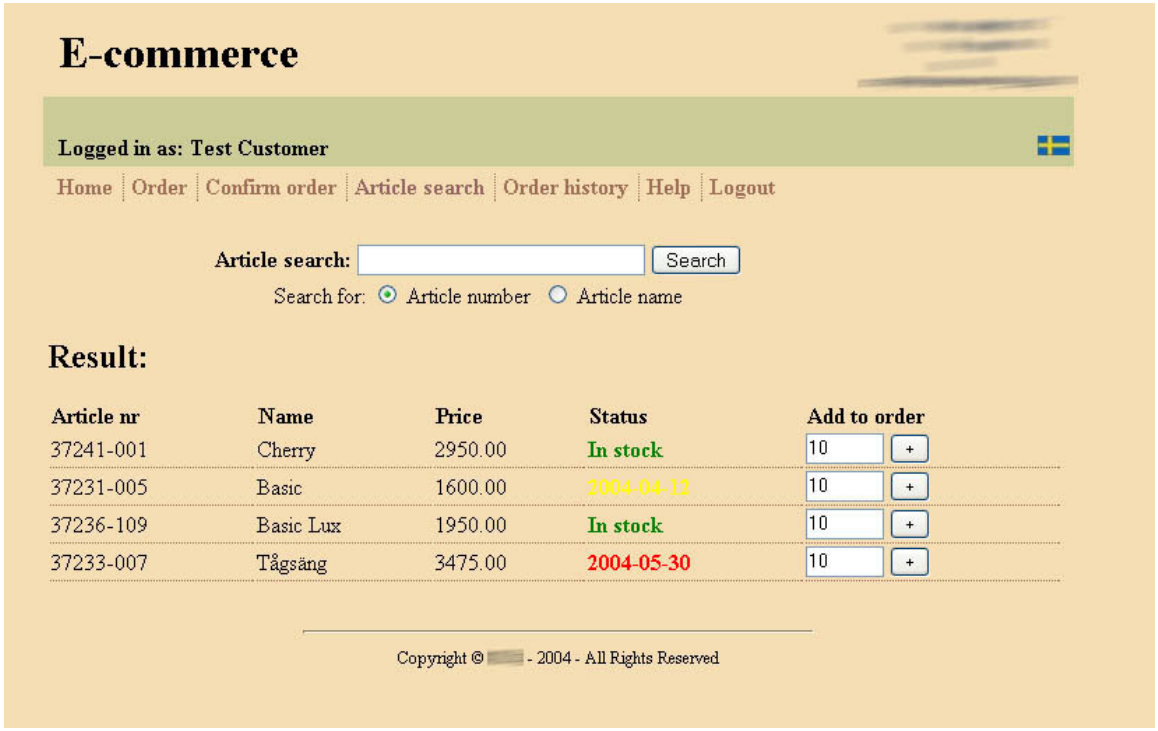
	This stereotyped class <<client page>> denotes a client page that is rendered by a web browser. Non-stereotyped attributes in this class typically defines JavaScript variables and operation defines JavaScript functions.

	The stereotyped class <<server page>> constitutes a server page that is executed on the server side. It is the logical abstraction of a Web page as seen by the server. Attributes and operations are architecture dependent.
	The class <<HTML Form>> can only exist in the context of a <<client page>> and maps directly to the <form> HTML element. Attributes map to child elements of the <form> element and cannot contain any operations.
	The stereotype <<build>> applied on an association can strictly be used in the context of a server page that streams HTML output as a client page.
	A <<submit>> stereotyped association is used by the HTML Form class allows to submit to a page class. This is typically a server page class. It can contain a list of parameters that is passed along the request.
	A <<redirect>> stereotyped association can be used by any page class. A redirect can occur both from a client page and a server page.
	A <<link>> stereotyped association points from a client page to another page. It maps directly to a HTML <a> tag.

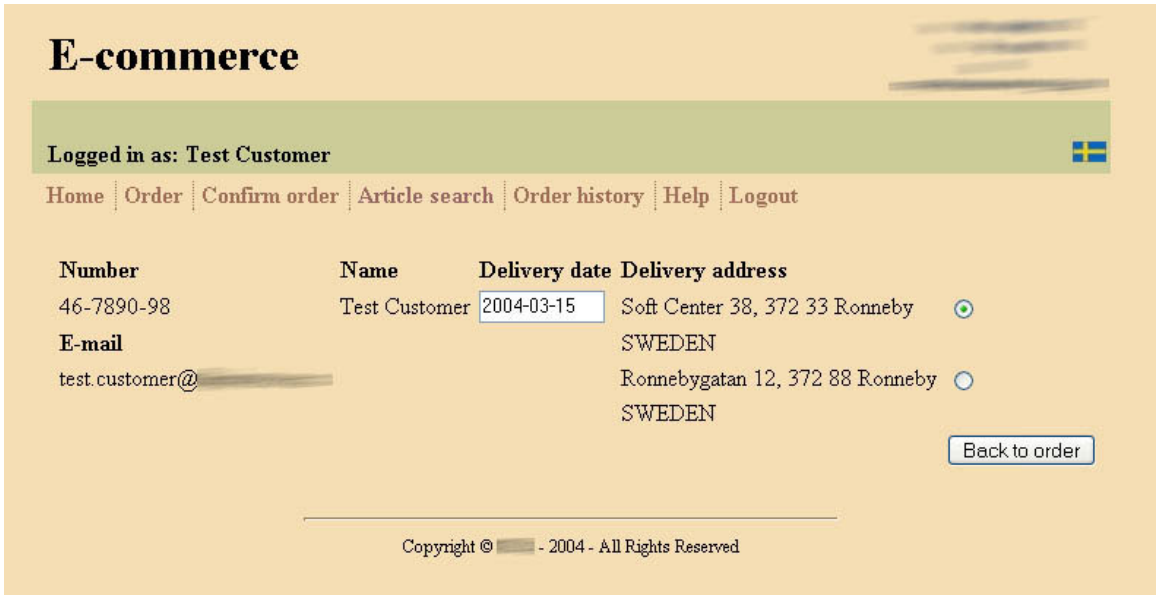
13 Appendix five – Original mock-ups

These mock-ups were developed in the initial phases of the original e-commerce Web application project.

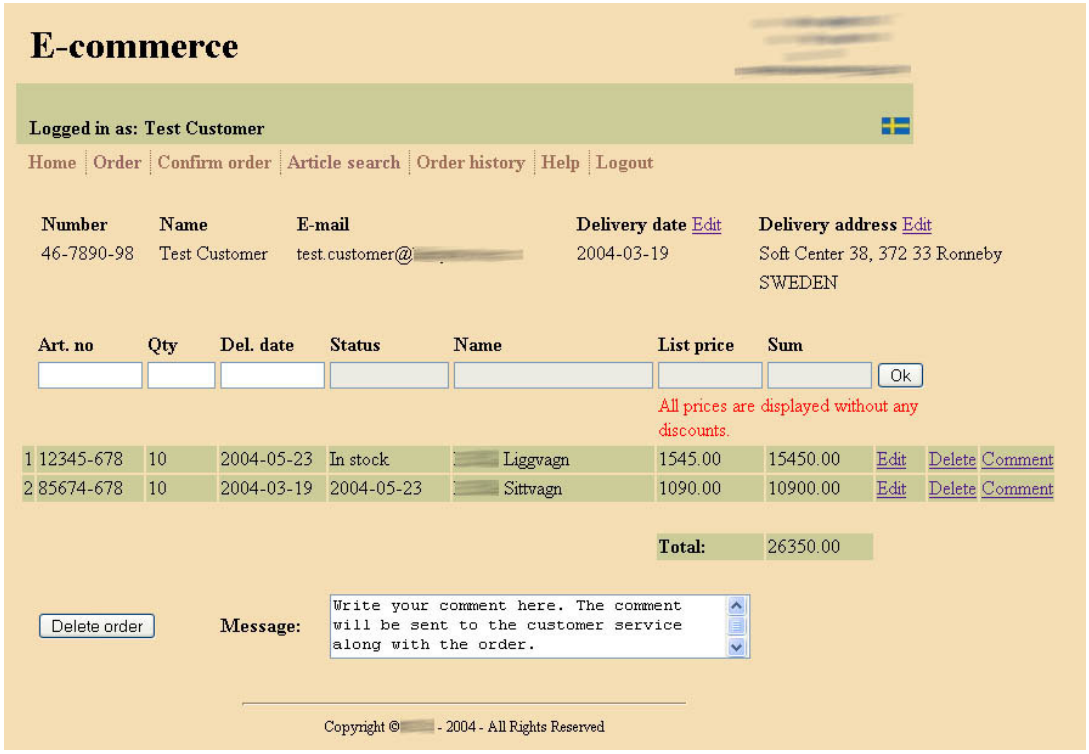
13.1 Mock-up - Article search



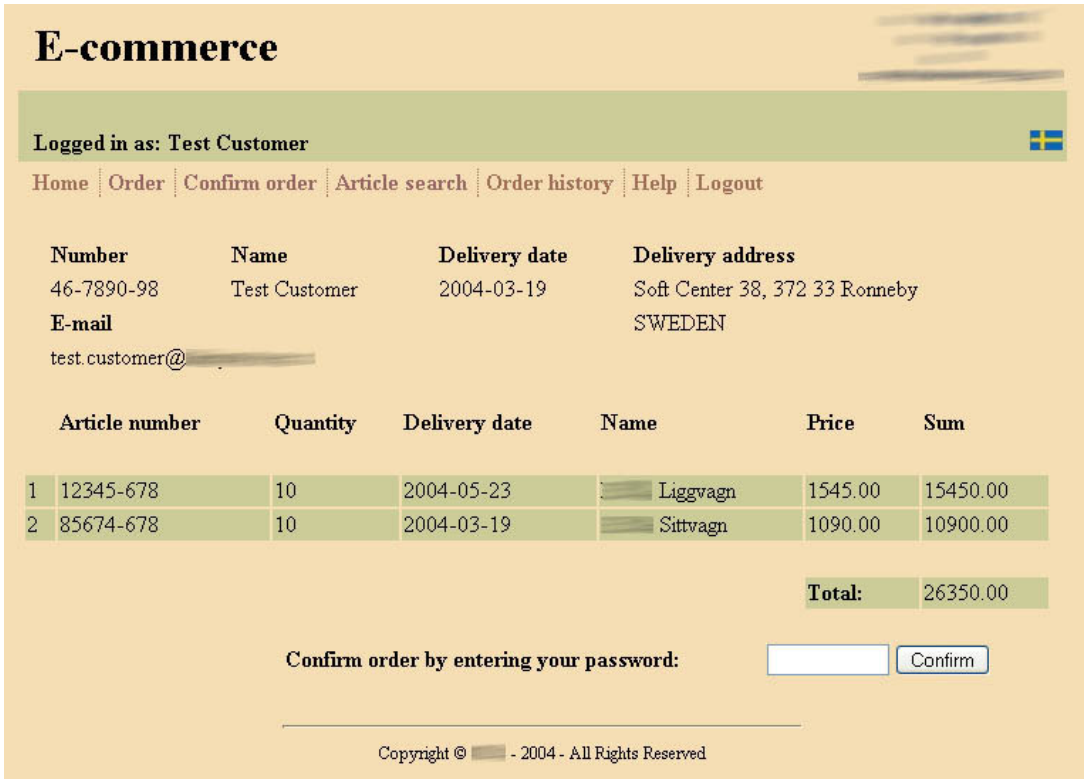
13.2 Mock-up – Create order



13.3 Mock-up – Place order



13.4 Mock-up – Confirm order



14 Appendix six – Interview answers

14.1 Answers

14.1.1 General Questions

1. Have you any experience (worked with) in developing web applications?

I1: Yes
I2: Certain experience looking at design diagrams of web applications. No programming.
I3: Yes. With .NET.
I4: Yes, develops web applications as profession.

2. What methods have you been using to modelling web applications?

I1: Text, specified requirements. Defining own web page designs from the specification. Creating own prototypes.
I2: UML, lines and boxes.
I3: WAE plus UX
I4: Specifications, use-cases, requirements.

3. Have you been programming using the .NET platform?

I1: Yes. Using VisualBasic och C#.
I2: No.
I3: Yes.
I4: Yes, but like mostly PHP.

4. Have you been modelling applications with UML?

I1: Yes.
I2: Yes.
I3: Yes.
I4: Yes.

5. Are you familiar with stereotypes?

I1: Yes, but not worked with it.
I2: Yes.

I3: Yes.
I4: Yes, but can't really explain it.

6. Have you heard the term WAE (Web Application Extension)?

I1: Yes, heard the term, not seen any examples.
I2: No.
I3: Yes, used before.
I4: No, not before interview.

14.1.2 Design questions

UML design

1. What do you think that the system do and what is its functionality? (What requirements do they find of the real requirements? Article search, search by number and name, add articles and fill in quantity).

I1: Article search, choose type with radio buttons, search, database search, showing the result of the search. Search on number or name. A session variable stores the search word and result.
I2: Article search of some kind.
I3: Database, writing in Microsoft, logins. If it is a Web site it looks confusing.
I4: Can place an order. Log what the user does, log in. Ordering system of some kind.

2. Do you think that the requirements are fulfilled by the old design?

Req. 1-2:	**I1**:	Fulfilled.
	I2:	Fulfilled with small modifications.
	I3:	No, article number missing.
	I4:	Have to look very hard and guess a few things. Not clear if.
Req. 3:	**I1**:	Fulfilled.
	I2:	Not fulfilled.
	I3:	Not like that (described).
	I4:	Can't be traced in the design.

Req. 4: **I1**: Maybe through DisplayResult() in 8.4. (Navigates to another page) Should have been more methods that place the result.

I2: Not fulfilled.

I3: Not shown, no parameter on ex submit ();

I4: No.

3. Would you say looking at the diagrams that the use-case is fulfilled?

I1: No. Some parts in the use-case are not traceable to the design.
I2: To some extent.
I3: The diagrams are not constructed like they should be. No!
I4: More abstract in the collaboration diagram than in the use-case. More details in the use-case.

4. Considering the old design diagrams, with the help of these diagrams, would you be able to implement the article search use-case? With .NET? With a language of your own choice? Which language?

I1: Maybe not. Should maybe be able to do it considering I was programming this part in the project.
I2: Yes, even though the diagrams are not that clear. Has worked with OMT and then you have to make up a lot of things yourself.
I3: Not from only the diagrams. I would need to use the use-case specification.
I4: No, not as the customer would want it. I would have missed a lot of things.

6. What happens if you try to add an article to the order but there is no order created?

I1: Can not be traceable. (No use of v 7.1.5.).
I2: Implicitly presumed an order already is created. Gets the order from a session object (Should be there already). Dos not tell what is happening if an order is not existing.
I3: Not in the diagram.
I4: Cannot tell what happens.

7. How do you think the implemented system performs? Or what the performance would be like from this design?

I1: It depends what is happening with the session object and if it saves everything in the database. It might not be so robust. I think it will be trouble with session variables. It's easier to maintain without session variables.
I2: Lumpy.
I3: Not so complicates system. It is hard to see the connection between use case and the diagram. Not slow because it uses only one object.
I4: No answer.

WAE-UML design

1. What do you think that the system do and what is its functionality? (What requirements do they find of the real requirements? Article search, search by number and name, add articles and fill in quantity).

I1: Order head, database, get/add to database, search/add article. It is easy to understand. It takes longer time to find things in the sequence diagrams. Too large sequence diagrams with too much detail. Put an item to an order; send back confirmation to the customer. Article search class diagram, it can be hard to understand what the stereotypes means. I would need an explanation of the stereotypes. Create order in order head, change date and address in order head, search articles, add articles up to a certain amount in the order.
I2: More information in these diagrams, but not that clear to get what the user does (changed his mind).Well, it is quite clear but harder to get the full picture because there are more details.
I3: Active web-pages which does interaction with the user and builds some pages, name search of something. Now you can actually see that it is a Web application.
I4: Broken down into pieces. Article search components glues together. The old design gives the background, this one the whole picture.

2. Do you think that the requirements are fulfilled by the new design?

1. Req. 1-2:

I1: Yes, can search on names, the search method is in logic.
I2: Yes (It now gave information that you can actually fill in the quantity for the line item.)
I3: Yes.
I4: Yes. Search item gives different kinds of search possibilities, status.

Connected to form.

2. Req. 3:

I1: Yes, addItem is there. Search item object. Text box.
I2: Not fulfilled. (The interviewee did not want to look at the sequence diagram or the class diagrams as they are not needed to his opinion.)
I3: Not found directly. (When pointed out that the sequence diagrams are linked he could find it.)
I4: Yes.

3. Req. 4:

I1: Yes, when BuilddialogAdded() is there, not sure where and how it is displayed. Article search sequence more clear.
I2: Yes.
I3: Maybe, no dialog box.
I4: Can not find any notification to the user (dialog box). (Commented that: "The level of detail can give different consequences depending on who it directs to.")

3. Would you say looking at the diagrams that the use-case is fulfilled?

I1: Yes.
I2: Basic flow fulfilled. I see that it constructs the page with the /build/ stereotype. Does not show what it does (the stereotype build), and what + sign for each row in the use-case is for.
I3: Detailed diagrams and hard to connect to abstract use case specification, guess yes.
I4: Same flow in the diagram and use-case. They describe the same thing.

4. Considering the new design diagrams, with the help of these diagrams, would you be able to implement the article search use-case? With .NET? With a language of your own choice? Which language?

I1: Yes I would.
I2: Would generate the code with a CASE tool and have my grandmother implement it easily.
I3: Yes, he guesses he could. Definitely.
I4: Yes, but I would have chosen PHP. I can see the adaptation to PHP instead. Would be able to do it immediately, but it is hard to see the flow of the user

navigation. Navigation flow in the Navigational map can make it easier though if removing error management and security from the navigational map.

5. We want to change the functionality so that an article added from the article search page should be handled by the placeOrder.aspx (the form is submitted here instead of back to articleSearch.aspx). Do you think that you would be able to do that? How would you do that?

I1: It is better that the article search handles this. I think I can do a redesign it but I am not sure how.
I2: Search article should post to place order. Simple. Only needs the sequence diagrams, then Rational Rose automatically will take care of the rest.
I3: Yes, some redesign required. What functionally in which? Have to dive in to the diagrams and documentation and find the background.
I4: The form in the article search posts to PlaceOrder.aspx instead.

6. What happens if you try to add an article to the order but there is no order created?

I1: It should have been a function that wouldn't allow the user to add articles in the first place; for example disabled buttons or show in the form that no order is created.
I2: Not so clear in Navigational map. (Checks sequence diagram.) Think there is an explicit order already. The order is already created.
I3: Not possible to answer. Who creates the order, assuming that the order has time line in it then it will be assumed it exists already. (He thought that we where talking about the order object when meaning the physical order.)
I4: Nothing in navigational map about this. Nothing in the class diagram either. Looking at the sequence diagram - connection to the order is visible, but not what is happening if you don't have an order created.

7. How do you think the implemented system performs? Or what the performance would be like from this design?

I1: The system quality should be better with more detailed diagrams. Clear improvements.
I2: It gives more information and can handle more things. It should function better. As for the structure; function might be equally well plus some added things. Gives more details for implementation. /build/ in the sequence diagram is annoying.

I3: Could be worse more redirection. Looks better in cohesion aspect, but could be performance problem.
I4: Can not say.

14.1.3 Follow-up questions

1. Which design would you choose if you were in a stakeholder position? Customer? Executive? Programmer?

Customer:

I1: In all cases WAE. It is a better way to design the diagrams.
I2: Would only need the use-cases if being a customer.
I3: Do not show this type to a customer. It contains too much code-behind detail and is confusing. It is difficult to follow the collaborations-diagrams in the new design. The old design does not tell all, if the system does what I want. Less entity, tells the truth but not the whole truth.
I4: WAE-UML, but if the customer doesn't know about design and UML it should be less information in the diagrams. IT-company (bigger) - WAE. An additional document for simpler stuff would be good.

Executive:

I1: In all cases WAE. A better way to design the diagrams
I2: Navigational map and sequence diagram only. It is clearer in the new design. Maybe I would choose the new one. The diagram structure is more foreseeable in the sequence diagrams (not the extensions).
I3: Same as if being a customer.
I4: Can not say.

Programmer:

I1: In all cases WAE. It is a better way to design the diagrams.
I2: The navigation map and sequence diagram is clearer in the new design. Maybe I would choose the new one. The form of the diagrams is more perspicuous for the sequence diagrams.
I3: Would choose the new design, gives me more detail.
I4: WAE design diagrams.

2. Do you think WAE could help you in future design? Is an improvement? How?

I1: The old design can be good to have as an initial template because it might go faster to completely redesign this one. New yes, might be able to reuse on future

similar designs. For example the storage layer could be used in another implementation. The screen navigational map is a bit confusing though.
I2: Did not answer because he is not developing any Web applications.
I3: Tells what is what, yes it definitely improves.
I4: It requires a lot of work to draw the design documents. Takes time to get accustomed to the diagrams and notation. Yes, would be helpful, but I would draw the design documents to fit me better. It would definitely be suitable for .NET development.