University of Salzburg
Institute of Computer Science

PS Software Engineering Part I

InfoWarrior

Online Information System for News Agencies

Thorsten Engl

Date: December 18, 2001
Version: 1.0

Contents

1 Introduction **5**
1.1 Rough System Requirements . . . 5

2 The CRC session **6**
2.1 Fundamentals of the CRC Session . . . 6
2.2 Planning the CRC Session . . . 6
2.3 Scenarios . . . 6
2.4 Identified Classes . . . 6
2.5 Goals . . . 7
2.6 Open Questions . . . 7
2.7 Session Results . . . 7

3 The Second CRC Session **8**
3.1 Session Goals . . . 8
3.2 Use Case Modeling . . . 8
3.2.1 Introduction . . . 8
3.2.2 Essential Use Case Models . . . 8
3.2.3 System Use Case Models . . . 9
3.3 Conclusion . . . 9
3.4 Essential Use Cases for InfoWarrior . . . 9
3.5 Essential User Interface Prototyping . . . 9
3.5.1 Introduction . . . 9
3.6 User Interface Prototyping . . . 10
3.6.1 Introduction . . . 10
3.6.2 Cycle . . . 10
3.7 Session Results . . . 10

4 The Identified Use Case Diagrams **12**
4.1 Competence Management . . . 12
4.2 Information Delivery . . . 13
4.3 Information Presentation . . . 13
4.4 Information Gathering . . . 14

5 Workflow System **15**
5.1 Activity Diagram - Competence Management Workflow . . . 15
5.2 Activity Diagram - Story Workflow . . . 16

6 Essential User Interface Prototypes **17**

7 Analysis Class Diagrams **18**
7.1 WorkflowSubSystem . . . 18
7.1.1 Description . . . 18
7.1.2 Used Patterns . . . 18
7.1.3 Class Diagram . . . 18
7.2 WorkflowTasks . . . 20
7.2.1 Description . . . 20
7.2.2 Class Diagrams . . . 20
7.3 RequestHandling . . . 20
7.3.1 Description . . . 20
7.3.2 Used Patterns . . . 21
7.3.3 Class Diagrams . . . 21
7.4 Actions . . . 21
7.4.1 Description . . . 21
7.4.2 Class Diagrams . . . 22
7.5 System users . . . 22
7.5.1 Description . . . 22
7.5.2 Class Diagrams . . . 23

8 Sequence Diagrams **24**
8.1 Introduction . . . 24
8.2 RequestHandling . . . 24
8.3 WorkflowSubSystem . . . 25
8.3.1 Check Workflow Process . . . 25
8.3.2 Query Stories . . . 25
8.3.3 Enter Vacations . . . 25

9 Activity Diagrams **26**
9.1 Introduction . . . 26
9.2 Login action . . . 26
9.3 Save Story action . . . 27

10 User Interface Prototypes **28**
10.1 User Interface Flow Diagram . . . 28
10.2 HTML User Interface . . . 29
10.3 WML User Interface . . . 30
10.3.1 Default Home Page . . . 30
10.3.2 Login Screen . . . 30
10.3.3 New Registration . . . 31
10.3.4 Registered User . . . 31
10.3.5 Reporter or Editor . . . 32
10.3.6 Adding a new Story . . . 32

List of Figures

3.1 Iterative steps of prototyping 11

4.1 Use Case Diagram: Competence Management 12
4.2 Use Case Diagram: Information Delivery 13
4.3 Use Case Diagram: Information Presentation 13
4.4 Use Case Diagram: Information Gathering 14

5.1 Activity Diagram: Competence Management Workflow 15
5.2 Activity Diagram: Story Workflow 16

7.1 Analysis Class Diagram: WorkflowSubSystem 19
7.2 Analysis Class Diagram: WorkflowTasks 20
7.3 Analysis Class Diagram: RequestHandling 21
7.4 Analysis Class Diagram: Patterns used in RequestHandling 22
7.5 Analysis Class Diagram: Actions 23
7.6 Analysis Class Diagram: System Users overview 23

9.1 Activity Diagram: Login Action 26
9.2 Activity Diagram: Save Story Action 27

10.1 User Interface Flow Diagram 28
10.2 WML user interface: Default home page 30
10.3 WML user interface: login screen 30
10.4 WML user interface: request new registration 31
10.5 WML user interface: registered user logged in 31
10.6 WML user interface: Reporter or Editor logged in 32
10.7 WML user interface: enter a new story 32

List of Tables

Chapter 1

Introduction

This chapter gives an overview about motivation for an online information system for news agencies.

1.1 Rough System Requirements

A News Agency (called "e-News") wants to redesign their information grabbing and information presentation processes with involvement of IT-Systems.
Potential users of the system are persons that collect news and other that retrieve that information. Beside them, also third party systems should be able to gather information out of InfoWarrior.
Therefore we can identify two possible scenarios:

- InfoWarrior will deliver the gathered information to other or proprietary systems by using XML-Technologies.
- Discrete persons will retrieve the gathered information from InfoWarrior. As a prerequisite for this task the information has to be prepared in an ergonomic manner.

Another special requirement is easy maintenance of InfoWarrior, which presumes that there will be no local system installations.
InfoWarrior should also be able to prepare information for the need of individual persons that are accessing the data.

Chapter 2

The CRC session

This chapter describes the fundamentals till the finished CRC session.

2.1 Fundamentals of the CRC Session

In effort to get the whole information from a problem the crc card system is a very good invention because information from a customer is mostly incomplete and not very precise.
The CRC session is in general like a "role game". A "player" (developer) represents a class. By stepping through predefined scenarios correlations between the different classes are quite simple found.
A customer joining this session sees the process of the program and is able to request some modifications.

2.2 Planning the CRC Session

The first task in a crc session is to prepare scenarios, which cover all possible steps of the problem.
By stepping through the scenarios possible classes are identified.

2.3 Scenarios

Can be found in the group documentation

2.4 Identified Classes

When preparing the CRC session we identified the following classes.

- InfoWarrior Request Handler
- InfoWarrior Submit Handler
- InfoWarrior Third Party System Handler

- Database
- News
- Message
- User
- XML engine
- XSLT engine
- File

2.5 Goals

- Full problem definition
- Requirement clarification
- Delimit of requirements
- Minimize risk for misunderstandings
- Class identification
- First approaches to Use Cases

2.6 Open Questions

- Multiple selection of default XSLT types or user specific?
- User hierarchy?
- How many different categories?
- Processing requests from third party system with only a simple XML interpreter, or various formats?

2.7 Session Results

According to the information I received the first CRC session was very successful, although the group did a little misinterpretation of the assigned task. We planned the CRC session in very technical terms, but this first meeting with the ordering customer should have been for identifying the whole business problem instead of identifying classes. But as of the very good session preparation almost all business requirements had been matched.

Chapter 3

The Second CRC Session

A second CRC session is of advantage to reach a higher state of clarification for the problem. Another point for a second session is to eliminate misapprehensions between the customer and the developers of the problem.

Another way to accomplish the elimination of such disaccordings is the usage of Essential User Interface Prototyping.

3.1 Session Goals

The primary goal of the second CRC session will be the identification of essential use cases with its corresponding actors. Beside, an essential user interface prototype should be discussed.

3.2 Use Case Modeling

3.2.1 Introduction

An important goal in requirements modeling is to come to an understanding of the business problem that your system is to address, in order to understand its behavioral requirements. With respect to object-oriented development, the fundamental artifact that you should develop to model behavioral requirements is a use case model. There are two basic flavors of use case models:

- Essential use case models
- System use case models

3.2.2 Essential Use Case Models

An essential use case model, often referred to as a business or abstract use case model, models a technology-independent view of your behavioral requirements.

3.2.3 System Use Case Models

System use case models, also known as concrete use case models or detailed use case models, model your analysis of your behavioral requirements, describing in detail how users will work with your system including references to its user-interface aspects.

3.3 Conclusion

A use case is a sequence of actions that provide a measurable value to an actor. Another way to look at it is that a use case describes a way in which a real-world actor interacts with the system.

An essential use-case is a simplified, abstract, generalized use case that captures the intentions of a user in a technology- and implementation-independent manner. It is complete, meaningful, and well designed from the point of view of users in some role or roles in relation to a system and that embodies the purpose or intentions underlying the interaction.

3.4 Essential Use Cases for InfoWarrior

Results can be found in the group documenation.

3.5 Essential User Interface Prototyping

3.5.1 Introduction

The user interface (UI) is the portion of software that a user directly interacts with. An essential user interface prototype is a low-fidelity model, or prototype, of the UI for your system it represents the general ideas behind the UI but not the exact details. Essential UI prototypes represent user interface requirements in a technology independent manner, just as essential use case models do for behavioral requirements. An essential user interface prototype is effectively the initial state, the beginning point, of the user interface prototype for your system. It models user interface requirements, requirements which are evolved through analysis and design to result in the final user interface design for your system.

There are two basic differences between essential user interface prototyping and traditional UI prototyping. First, the goal is to focus on users and their usage of the system, not system features. This is one of the reasons why essential use case modeling and essential user interface prototyping should be performed in tandem: they each focus on usage. Second, prototyping tools are very simple, including white boards, flip chart paper, and sticky notes. The minute that electronic technology is introduced to prototyping efforts a design decision about the implementation technology may habe been made. If an HTML development tool is used to build a user interface prototype then immediately the design space to the functionality supported is narrowed within browsers. If Java development environment is chosen, then the design space has been narrowed to Java, the same occures if a Windows-based prototyping tool will be used. Understand the problem first, then solve it.

3.6 User Interface Prototyping

3.6.1 Introduction

User interface prototyping is an iterative analysis technique in which users are actively involved in the mocking-up of the UI for a system. UI prototyping has two purposes:

- It is an analysis technique because it enables you to explore the problem space your system addresses.
- UI prototyping enables you to explore the solution space of your system, at least from the point-of-view of its users, and provides a vehicle for you to communicate the possible UI design of your system.

3.6.2 Cycle

- Determining the Needs of Your Users
- Building the Prototype
- Evaluating the Prototype
- Determining if you're finished

3.7 Session Results

The second CRC-Session was very successful, because we clarified further system requirements and identified new use-cases and gui requirements. One of the biggest steps forward to a reasonably complete system specification was the discovery that our system should have some abilities to manage business workflow in the target companies.

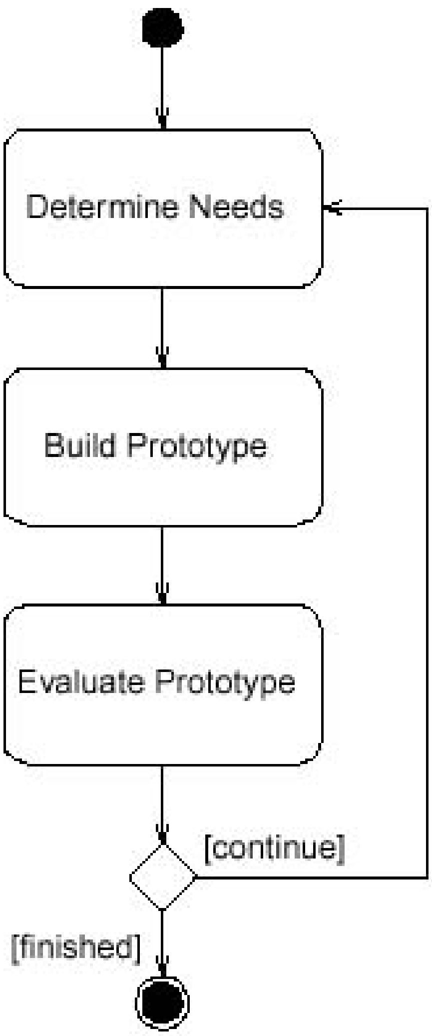

Figure 3.1: Iterative steps of prototyping

Chapter 4

The Identified Use Case Diagrams

By working through the scenarios and the essential use cases the following use cases were identified.

Due to the new software requirement that some workflow system functions should be included, we had to add a new method for handling such cases. Therefore the old message driven quasi-workflow machine will be replaced by a template driven full enhanced workflow engine.

4.1 Competence Management

Competence management enables a secretary to manage illness or vacation of employees. This is a very important part of the new workflow system because it enables the software to automatically decide the responsibility of persons for open workflow tasks, even when the original responsible person is not available.

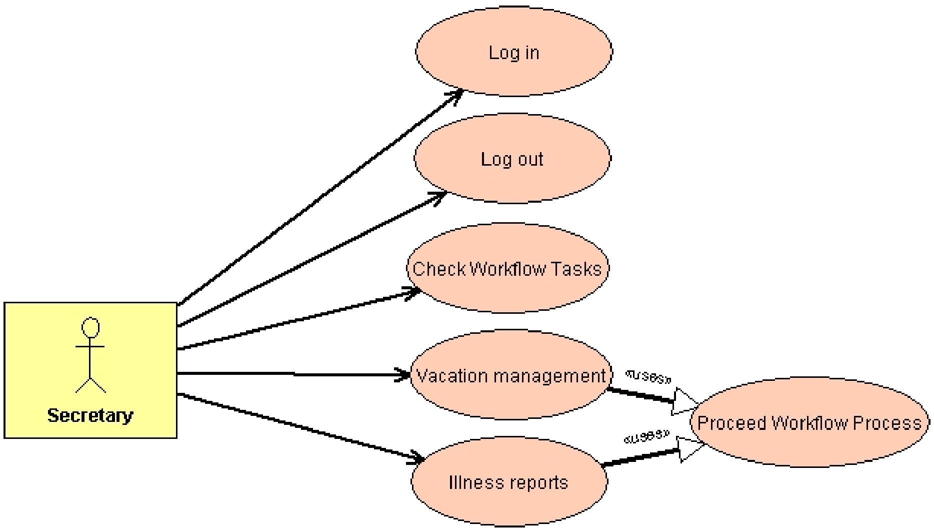

Figure 4.1: Use Case Diagram: Competence Management

4.2 Information Delivery

Third party systems are able to subscribe to InfoWarrior, to get the latest news on demand.

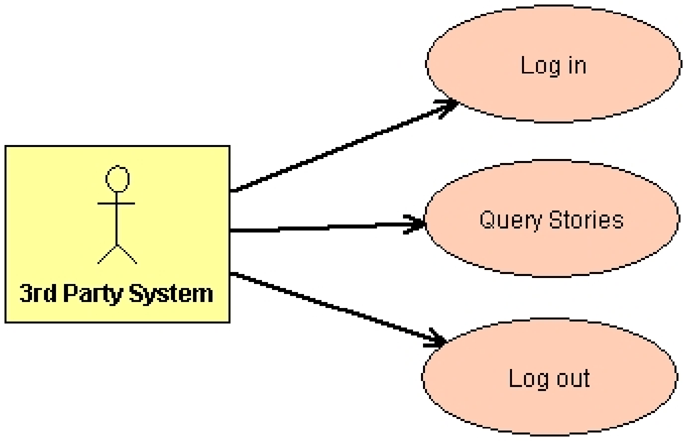

Figure 4.2: Use Case Diagram: Information Delivery

4.3 Information Presentation

Why collect stories if nobody can read them? Presentation of stories can be found here.

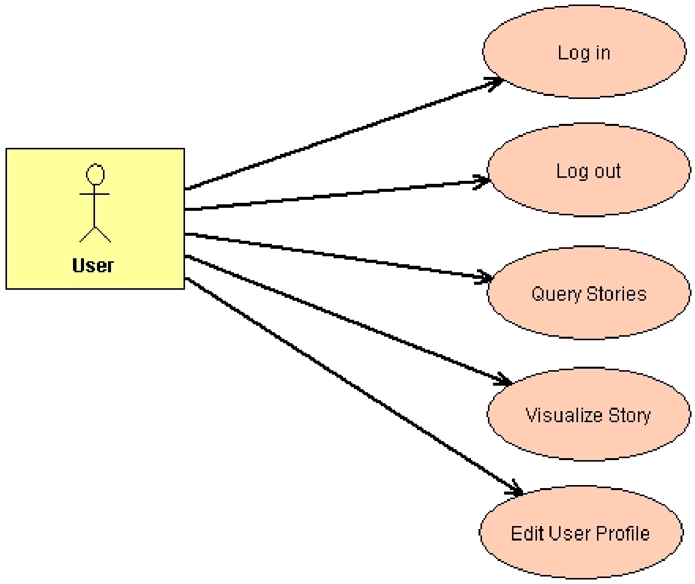

Figure 4.3: Use Case Diagram: Information Presentation

4.4 Information Gathering

The most complex use case diagram, can be found here. It shows the creation of a story until its release or cancellation.

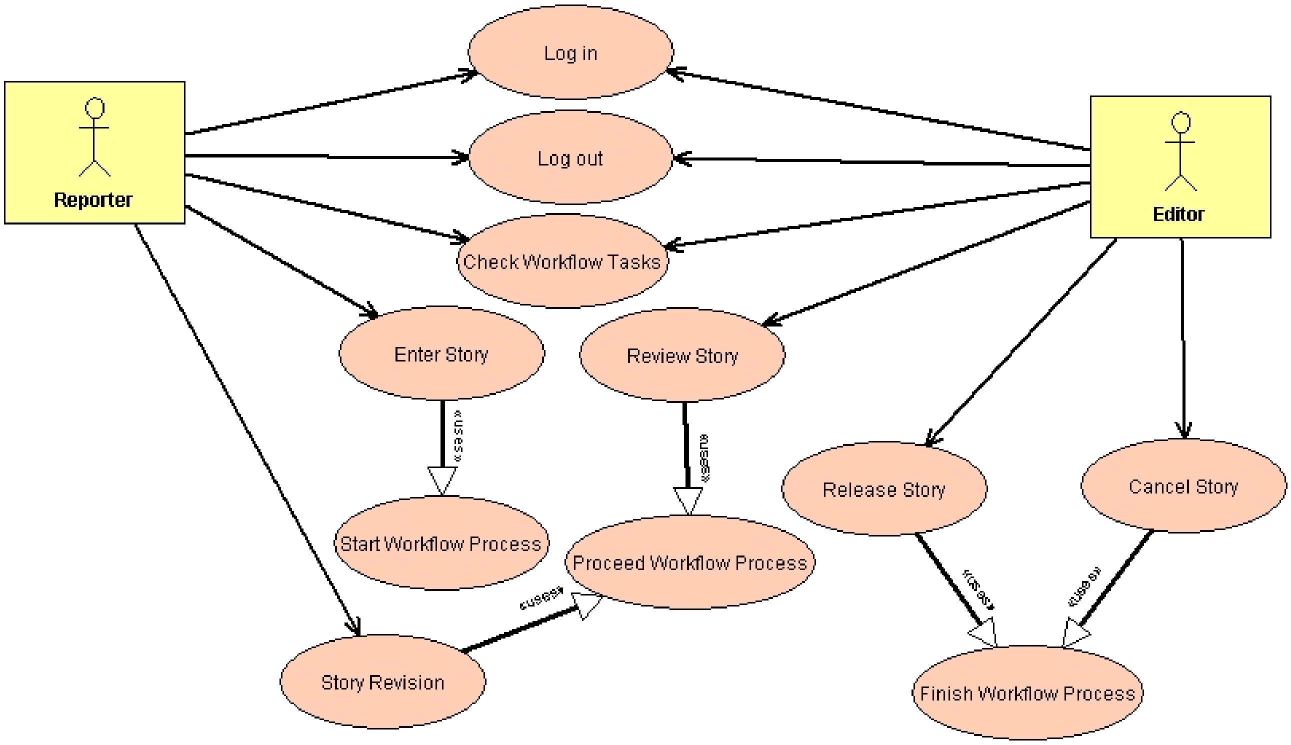

Figure 4.4: Use Case Diagram: Information Gathering

Chapter 5

Workflow System

The workflow system of InfoWarrior is a template driven system that is capable of managing different workflow processes at the same time, independent of the initiating user. The change of the responsibility of a task is also a very common used function.
The different workflow processes we have identified are modeled in UML by using activity diagrams, as show below:

5.1 Activity Diagram - Competence Management Workflow

The handling of illness reports and vacation planing of epmloyees is show in this activity diagram.

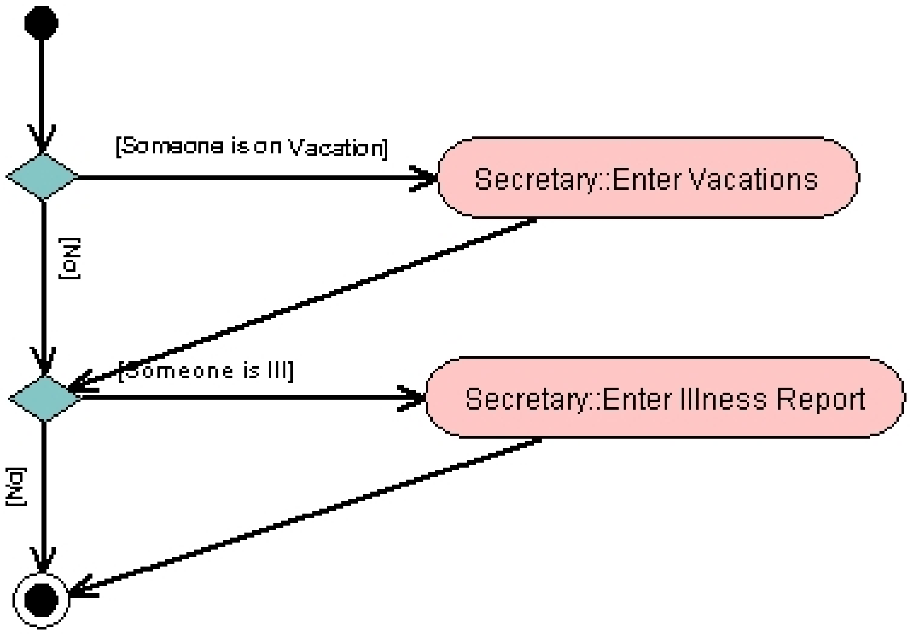

Figure 5.1: Activity Diagram: Competence Management Workflow

5.2 Activity Diagram - Story Workflow

Workflow from the input of the story in the system, until its completetation and release, or due the lack of 1/Entropy its cancellation.

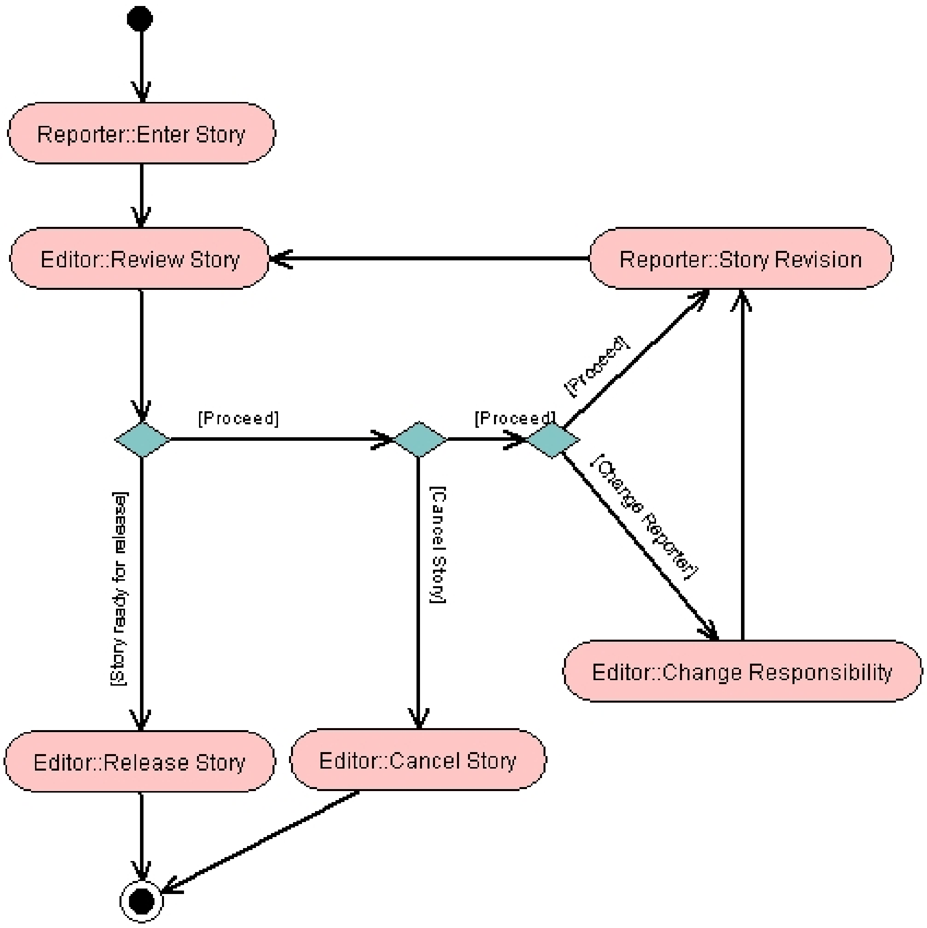

Figure 5.2: Activity Diagram: Story Workflow

Chapter 6

Essential User Interface Prototypes

I've not been personally involved with this section, but it can be found at the group documentation. But I used these essential prototypes for the first design of our user interface

Chapter 7

Analysis Class Diagrams

7.1 WorkflowSubSystem

7.1.1 Description

Each workflow consists of a process which can have an arbitrary amount of tasks. This workflow also has a protocol, consisting of each tasks state which was running during the lifetime of the workflow. This is where the Memento design pattern takes place.

Each task has specific groups of users assigned which are responsible for the further attention handling, but only one user out of this groups will actual process the current task.

Every user owns exact one calendar, which holds the information for vacation,illness and if the user is currently available . This is needed for the workflow subsystem to automatically determine which users of the groups are available for further tasks.

7.1.2 Used Patterns

Memento Design Pattern The memento pattern is used to record and export the inner state of an object, without loose of its inner encapsulation, to obtain the ability to return to this state later on. Taken from [1][Pages 317-332].

7.1.3 Class Diagram

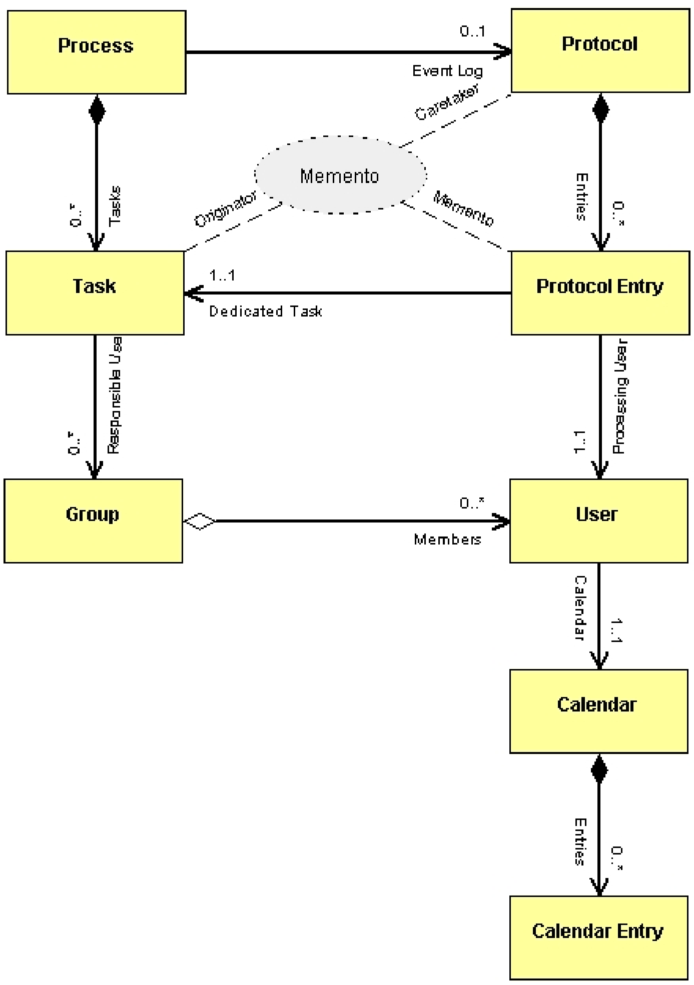

Figure 7.1: Analysis Class Diagram: WorkflowSubSystem

7.2 WorkflowTasks

7.2.1 Description

Currently three different tasks exist in our system, and each of these tasks has a different amount of successors.

WorkTask Only one successor can occure in this specific type of a task. The actual work on a story is done in this type of task.

DecisionTask Here a decision takes place what the next worktask will be. Each decision task has two possible successors.

StartStopTask For starting and stopping a workflow.

7.2.2 Class Diagrams

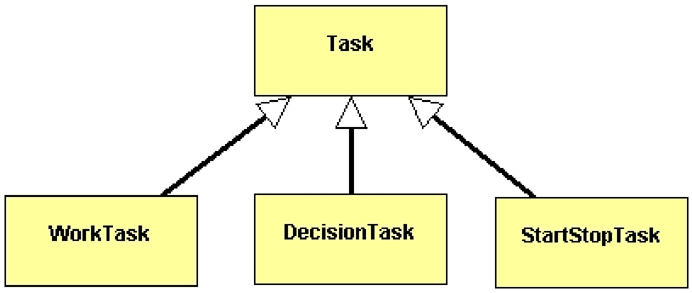

Figure 7.2: Analysis Class Diagram: WorkflowTasks

7.3 RequestHandling

7.3.1 Description

A client causes many different requests to the system, these requests have to be processed by the Front Controller to get the users proper permission and to check the requests validity. This introduces an additional security level, to keep unauthorized accesses to the system away from the inner processing of such requests.

After a request is authorized to be processed the dispatcher, in our case the Action Mapping, creates the corresponding Action. An Action is realized by implementing the Command Pattern. The client invokes the execution of such an action, undoing the last action is automatically received by using this pattern.

Triggering actions causes the creation of views respective to the specific request. Such a view could be HTML pages for browsers, or XML data for third party systems. Which will be afterwards sent back to the client.

7.3.2 Used Patterns

FrontController The request handling mechanism must control and coordinate processing of each user across multiple requests. Such control mechanism my be managed in either a centralized or decentralized manner. Taken from [2][Pages 172-185].

Command Encapsulate a command as an object. To obtain the ability, to vary the parameters of clients with different requests, to put operations into a queue, to create a log and to undo operations. Taken from [1][Pages 245-256]

7.3.3 Class Diagrams

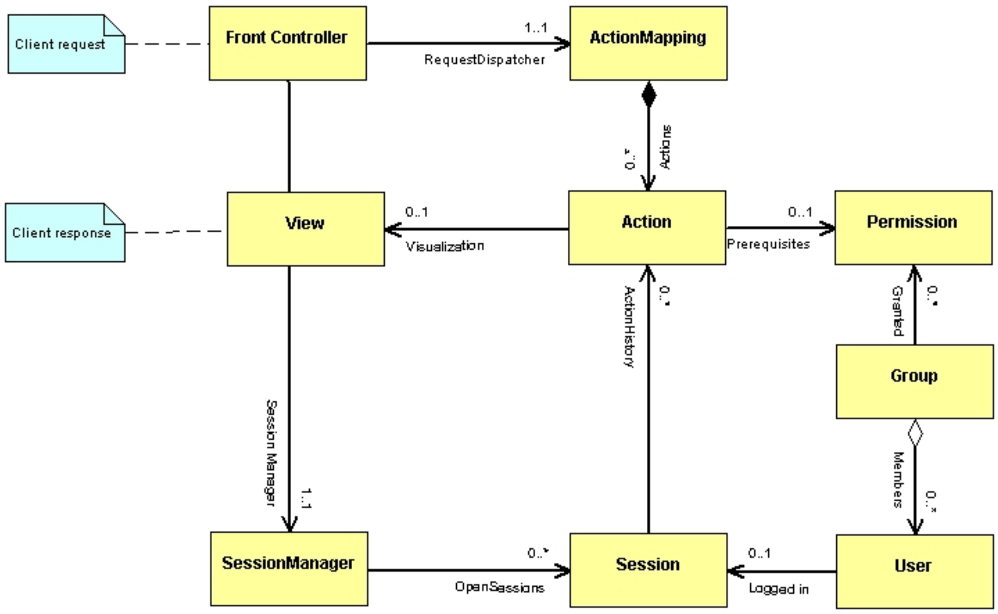

Figure 7.3: Analysis Class Diagram: RequestHandling

7.4 Actions

7.4.1 Description

All so far known needed actions are listed below.

acLogin

acLogout

acCheckWorkFlowTasks

acQueryStories

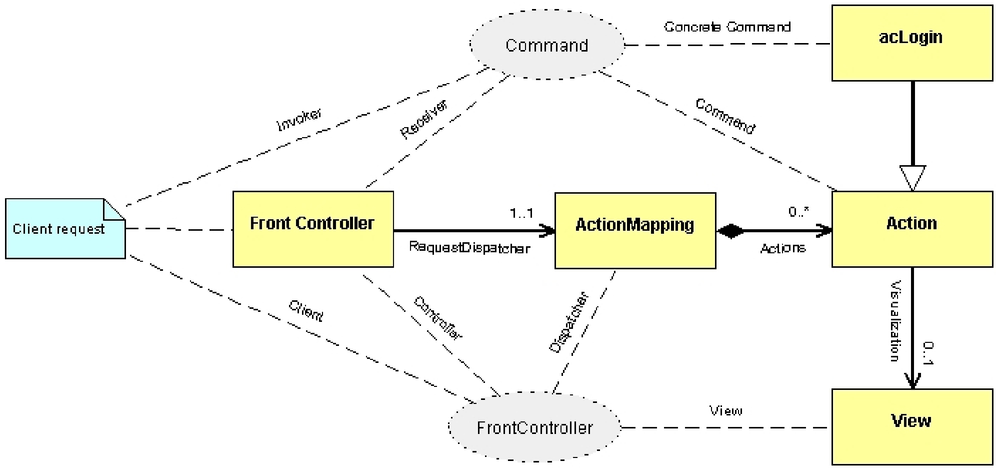

Figure 7.4: Analysis Class Diagram: Patterns used in RequestHandling

acQueryStory

acSaveStory

acReleaseStory

acCancelStory

acGetUserSettings

acSetUserSettings

acSetCalendar

acGetCalender

7.4.2 Class Diagrams

7.5 System users

7.5.1 Description

Here all so far known different type of users can be found here.

Editor

3rd Party System

Secretary

Reporter

Viewing User

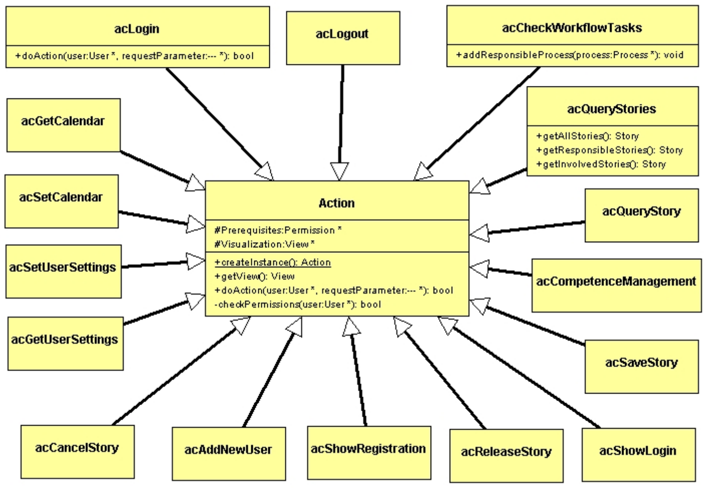

Figure 7.5: Analysis Class Diagram: Actions

7.5.2 Class Diagrams

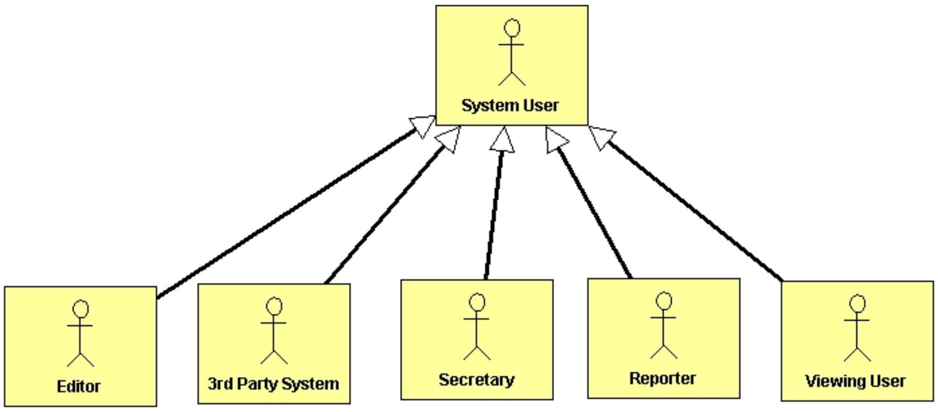

Figure 7.6: Analysis Class Diagram: System Users overview

Chapter 8

Sequence Diagrams

Detailed information can be found in the group documentation.

8.1 Introduction

A sequence diagram shows the program sequence of a method and considers the interaction with other objects. It is a form of visualizing the interaction model of an object oriented modeled target system. In doing so it shows the object relations and the chronology of method calls.

8.2 RequestHandling

This sequence diagram documents the basic technique for processing requests from any possible client.
Special design decisions:

- For every incoming request, if not yet existent, a new session will be created.
- The doAction method of the action class decides whether to delete the session or not.
- The users permission on the requested action is checked by the action itself.
- The action class has to have an internal state in order to know which view will be returned by the getView method. This method has to be called after the doAction method.

8.3 WorkflowSubSystem

8.3.1 Check Workflow Process

The sequence diagram for checking the workflow process gives an overview how the responsibility for a workflow task is determined.

8.3.2 Query Stories

This really complex diagram illustrates all possibilities of querying stories saved in the system. There are four ways for querying stories:

1. Query all stories
2. Query all stories with constraints (e.g. only sport stories)
3. Query stories the user is responsible for
4. Query stories the user is involved with

All these cases are modeled in the sequence diagram, but case one and two are almost the same, the only differ in the query parameter given to the GetStories method of the information storage.

8.3.3 Enter Vacations

This sequence diagram can be seen as a higher level model because it illustrates the interaction of a sequence of requested actions.

Chapter 9

Activity Diagrams

9.1 Introduction

The activity diagram is an alternative to a state diagram. It shows the interference of states and their relations.

Main goal of an activity diagram is the description of a methods algorithm, which in this context is called an activity.

9.2 Login action

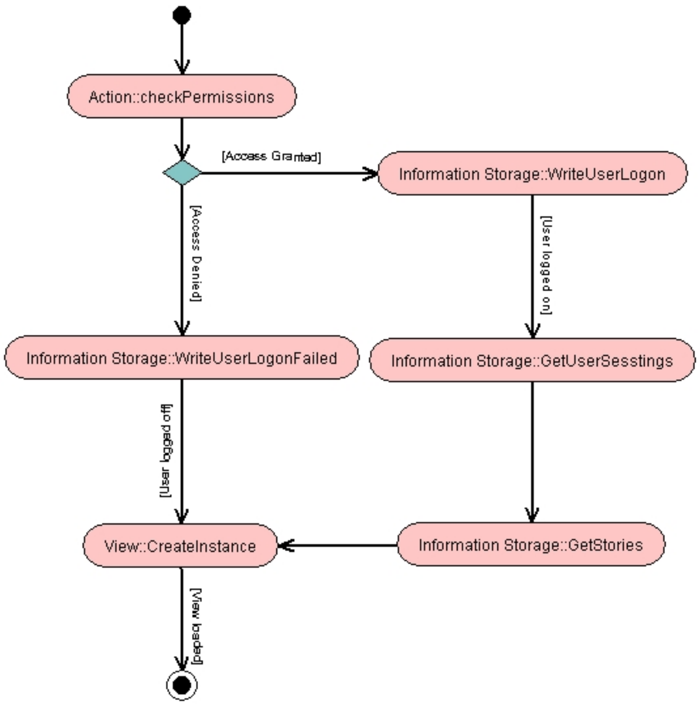

Figure 9.1: Activity Diagram: Login Action

9.3 Save Story action

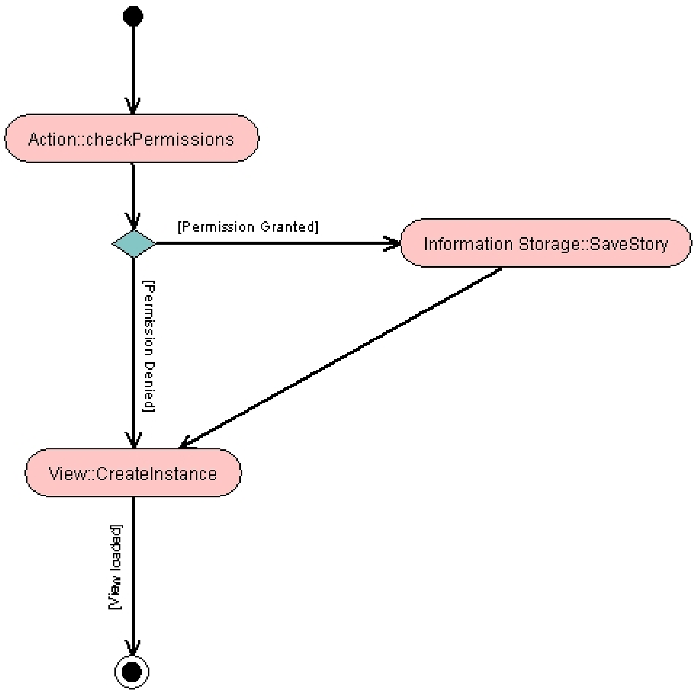

Figure 9.2: Activity Diagram: Save Story Action

Chapter 10

User Interface Prototypes

10.1 User Interface Flow Diagram

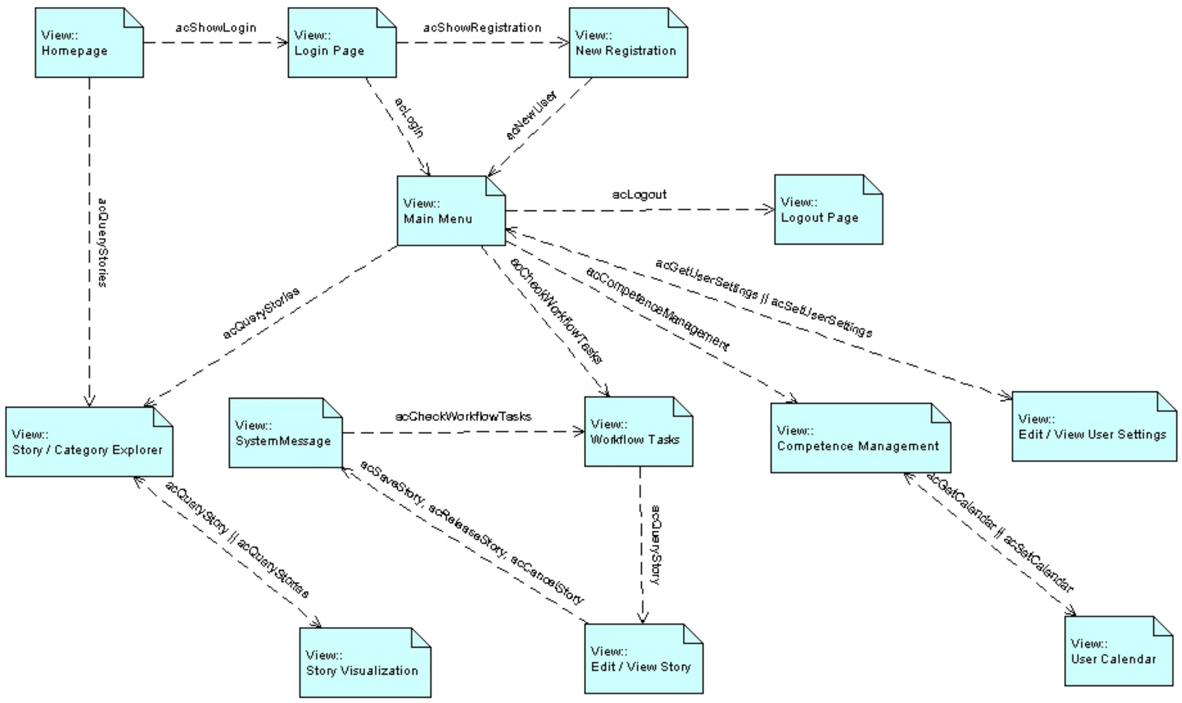

Figure 10.1: User Interface Flow Diagram

10.2 HTML User Interface

Can be found in the group documentation.

10.3 WML User Interface

At first a simple WML design was used, which worked properly in an WinWap browser, but its real functionality could only be tested on a real portable.

For this purpose I used the Nokia 7110 Simulator, at the first attempt the whole system crashed :-).

After adding an XML-Header to the file, splitting it up in more smaller files, and using references to get to other cards instead of buttons it finally worked.

10.3.1 Default Home Page

Figure 10.2: WML user interface: Default home page

10.3.2 Login Screen

Figure 10.3: WML user interface: login screen

10.3.3 New Registration

Figure 10.4: WML user interface: request new registration

10.3.4 Registered User

Figure 10.5: WML user interface: registered user logged in

10.3.5 Reporter or Editor

Figure 10.6: WML user interface: Reporter or Editor logged in

10.3.6 Adding a new Story

Figure 10.7: WML user interface: enter a new story

Bibliography

[1] Erich Gamma; Richard Helm; Ralph Johnson; John Vlissides. *Entwurfsmuster - Elemente wiederverwendbarer objektorientierter Software.* Addison-Wesley, 1st edition, 1996. ISBN 3-89319-950-0.

[2] Deepak Alur; John Crupi; Dan Malks. *Core J2EE Patterns - Best Practices and Design Strategies.* Sun Microsystems, Inc., 1st edition, 2001. ISBN 0-13-064884-1.