PS Software-Engineering

SE-009-2001 (final) 29th January 2001

Archaeological Data System - ADS

Report

Markus Amersdorfer
mamers@cosy.sbg.ac.at

Martin Angerer
mangerer@cosy.sbg.ac.at

Dominik Engel
dengel@cosy.sbg.ac.at

Sandor Herramhof
sherram@cosy.sbg.ac.at

Christoph Obermair
cober@cosy.sbg.ac.at

Academic Supervisor Mag. Dipl.-Ing. Schwaiger Roland

Department of Computer Science
University of Salzburg

Correspondence to:

Universität Salzburg
Institut für Computerwissenschaften und Systemanalyse
Jakob–Haringer–Straße 2
A–5020 Salzburg
Austria

Archaeological Data System - ADS

M. Amersdorfer, M. Angerer, D. Engel, S. Herramhof, C. Obermair
Department of Computer Science
University of Salzburg
A–5020 Salzburg, Austria
{mamers}, {mangerer}, {dengel}, {sherram}, {cober}@cosy.sbg.ac.at

Abstract

This document deals with the development of a software-system with the working title "Archaeological Data System - ADS" for entering, administrating and presenting archaeological data via the internet.

Contents

1 Requirements Definition **4**

1.1 Initial Situation and Objective Target 4
1.1.1 Introduction 5
1.1.2 General Overview 5
1.1.3 Database 6
1.1.4 Entering Data 7
1.1.5 Virtual Museum 8
1.1.6 Additional Notes 8
1.2 System Environment 9
1.2.1 Hardware 9
1.2.2 Software 9
1.2.3 Data 9
1.2.4 Users 9
1.3 User Interfaces 9
1.4 Functional Requests 9
1.4.1 Database 10
1.4.2 Web-Frontend 12
1.4.3 Virtual Museum 13
1.5 Non-Functional Requests 14
1.6 Error Handling 15
1.6.1 Database 15
1.6.2 Web-Frontend 15
1.6.3 Virtual Museum 16
1.7 Documentation Requirements 16
1.8 Acceptance Criteria 16
1.9 Risk Analysis 16
1.9.1 Database 17
1.9.2 Web-Frontend 17
1.9.3 Virtual Museum 17
1.9.4 Stability 17
1.9.5 Personnel 17
1.9.6 Hardware 18
1.9.7 Miscellaneous 18
1.10 Glossary and Index 18

2 Use Case Documentation **19**
2.1 Introduction 19
2.2 System Description 19
2.2.1 System-Level Use Case Diagram 19
2.2.2 Actors 19
2.2.3 Primary Use Cases 21
2.3 Detailed Use Case Description: Primary Use Cases 22
2.3.1 Use Case: Authenticate 22
2.3.2 Use Case: Enter Data 26
2.3.3 Use Case: Enter Scientific Data 28
2.3.4 Use Case: ViewData 34
2.3.5 Use Case: ViewMuseum 37
2.3.6 Use Case: Edit Museum 41
2.3.7 Use Case: SearchData 47
2.3.8 Use Case: CreatePublications 51
2.3.9 Use Case: Administrate 54

3 Project Proposal **57**
3.1 Weighting Actors 57
3.2 Weighting Use Cases 58
3.3 Weighting Technical Factors 59
3.4 Weighting Environmental Factors 60
3.5 Conclusion 61
3.5.1 Use Case Points 61
3.5.2 Project Estimate 61
3.5.3 Costs 61

4 Architecture and Project Plan **62**
4.1 Architecture 62
4.2 Project Plan 63
4.2.1 Iteration 1 64
4.2.2 Iteration 2 64
4.2.3 Iteration 3 65
4.2.4 Iteration 4 65

5 Prototypes **66**
5.1 The first prototype 66
5.1.1 General 66
5.1.2 Facts 66

6 Iteration Plans **68**
6.1 Iteration 1 68
6.1.1 Database Diagram 68
6.1.2 Class Diagram 68
6.1.3 Sequence Diagrams 69
6.1.4 Collaboration Diagrams 69
6.1.5 StateDiagram 70

A Protocols 79
A.1 11th October 2000 79
A.2 18th October 2000 79
A.3 31st October 2000 80
A.4 8th November 2000 80
A.5 16th November 2000 80
A.6 7th December 2000 81
A.7 20th December 2000 81
A.8 11th January 2001 81
A.9 25th January 2001 82

B Points to clarify (November 8, 2000) 83
B.1 Use-Case 83
B.2 Webfrontend 84
B.3 Database 84
B.4 Management of Rights 84
B.5 Museum 84
B.6 Publishing of CD-Rom and Catalogue 85

Bibliography 86

Chapter 1

Requirements Definition

Under "requirements definition" we understand the development of contract between our anonymous customer and us, the SE AG. This contract defines exactly, what the software system should be able to do and which premises are valid for the realization.
Since there is no exact definition for "requirements definition" we follow the ANSI/IEEE Guide to Software Requirements Specification [ANSI 830 - 1984] and use the following structure for the requirements definition:

1. Initial Situation and Objective Target
2. System Environment
3. User Interfaces
4. Functional Requests
5. Non-Functional Requests
6. Error Handling
7. Documentation Requirements
8. Acceptance Criteria
9. Risk Analysy
10. Glossary and Index

Before the contract between our customer and us can be established, the requirements definition has to be proved of completeness and correctness and the customer's demands have to be verified whether they can be realized technically and economically. Therefore it is necessary to perform a risk analysis (see Section 1.9 on page 16).

1.1 Initial Situation and Objective Target

This section aims at clarifying topics such as our customer's wishes and requirements concerning the software, it shall give restrictions what the software is not to do as well as point out questions which are to be discussed with our customer.

As R. Yeh and P. Zave describe [1], a great deal of time and money is invested in maintenance of a software-product. To reduce this in order to be more productive as you can concentrate on other projects, a very important part of software-developement is to specify exactly the system's requirements.
Yeh and Zave also mention 3 major steps which should be persued during this phase of the software-project:

1. *Problem identification:* One shall identify and describe the needs of a system for certain purposes;
2. *Problem understanding:* Collect and analyze information about the system and its environment, as well as their interaction;
3. *Problem specification:* Describe the behavior of the system.

This section is the product of these 3 steps.

1.1.1 Introduction

Our company was engaged to develop a software-system with the working title "Archaeological Data System - ADS" which first of all shall:

- Provide an interface via the world-wide-web enabling our customer to enter archaeological data such as tombs, settlements or artefacts into a central database.
- Enable third parties to query the database for information via the world-wide-web.
- Present the data also via a virtual museum.

We are responsible for the design and implementation of this system including the following points:

- Web-front-end to manipulate and display data.
- The design and implementation of the database itself.
- Maintain these services.
- Host the database on a server or at least administrate this work.

1.1.2 General Overview

As the database shall be accessible via a standard web-browser, we will not develop an independent program which could connect to the database for example via TCP/IP. Just the web-frontend for web-browsers will be designed and implemented.

Our Customer has no experience with a similar software, so we have to organize seminars to teach the customer how to operate the system and additionally write a documentation manual for the software. This is necessary because once ADS is in action the customer himself is responsible for the system. We only will act as ASP (Application Service Provider) then.

There are no limitations in the financial capital we have for this project. In January 2001 the Inception- and Elaboration-Phase of the project should be finished and maybe we can present our customer a prototype.

1.1.3 Database

Currently there are 4000 records to enter into the database. A record will consist of approximately 20 attributes and can also contain pictures, sketches, movies and sounds. The expected annual growth of data is about 400 records at all (400 pics, 400 sketches, 10 movies and several sound-samples). The multimedia-files have to be uploaded through our web-interface. These files will not be stored into the database, but rather using the filesystem. The database will only store the path to the respective files.

The attributes for the different fields can be found in the next chapter.

To search in the database, our web-frontend will provide a search-screen. There will be three kinds of queries:

Simple query: Just enter a keyword and all the attributes will be compared to it.

Extended query: One can specify, which data certain attributes should contain and connect all the search-options by logical operators.

Query in the museum: The visitor can search for special items in the museum.

The data collectors can specify certain attributes of a record, the visitor cannot see when searching in the database. He can also make whole records invisible for the visitors.

As the customer wishes the option to have several virtual museums at once, one of the records attributes will propably be, to which virtual museum a specific object belongs to.

Connection Pool

A dynamic web site generates HTML pages from information stored in a database. Each request for a page results in a database access. But connecting to a database is a time consuming activity since the database must allocate communication and memory resources as well as authenticate the user and set up the corresponding security context. The exact time varies, of course, but it is not unusual to see connection times of one or two seconds. Establishing the connection once and then use the same connection for subsequent requests can therefore dramatically improve the performance of a database driven web application. Since Servlets can keep information between requests, a database connection pool is a straight forward solution.
The database connection pool should provide methods to

- get an open connection from the pool,
- return a connection to the pool,
- release all resources and close all connections at shutdown.

It also handles connection failures, such as time-outs, communication failures, etc. and can limit the number of connections in the pool to a predefined max value. By developing a web application where all users can execute some database operations but others are restricted to authorized users, it's posible to define one pool for the general user and another pool for the restricted group using the same URL but different user names and passwords.

1.1.4 Entering Data

There are two main groups who can enter data into the database:

1. Data collectors
2. Scientific data collectors

Members of the first group will enter general data concerning the archaeological object such as the place where it was found, while members of the second group will add additional information such as interpretations and publications concerning this object.

To address all questions which have to do with "Who has the right to do what in the database?" a database-access-system has to be developed. It should enable the customer to organize the users of the software-system in groups and equip them with certain rights (read, write, modify).
So ADS will provide a user-management such as known from UNIX. There will be five groups so far:

Data collectors: They can enter and manipulate general data concerning the archaeological objects. They also can define relationships between different types of data or specific records. Data collectors also can edit the virtual museum where the objects are to be presented.

Scientific data collectors: They can also enter and manipulate data (even more specific one, such as publications concerning the archaeological objects). Again they have the option of defining relationships as mentioned before as well as editing the virtual museum. So this group is something like an extension of the data collectors.

Designers: They are responsible for the design of the museums/rooms/showcases. They do not have the right to modify data in the database.

Visitors: They will just have the option to query the database for information as well as to walk through the virtual museum.

Administrators: The administrators will be able to do anything they want within the system, so they have the ability to maintain and repair small but urgent concerns, perhaps via the internet.

In the design of ADS we will keep an eye on the ability of adding new groups or changing groups in future versions.

1.1.5 Virtual Museum

The software-system shall include a virtual museum where the visitors can walk through and zoom into descriptions of archaeological objects and artefacts. The museum shall not host all the data collected as the customer wants to perform special exhibitions on current topics.
There is no need to model the museum with 3D-graphics. Nevertheless, we have several options:

- Show photography-like pictures of the rooms, move within the museum by clicking on arrows or signs.
- Present the museum animated (via tools such as Quicktime, Flash, Shockwave, etc.)
- Just show a clickable room-map.
- Others.

At this point we want to remind the customer of the fact that the virtual museum will be accessed via the internet and as the majority of visitors will have low-speed connections, they will not be able to fully enjoy the museum if there is too much (graphical) data to be transferred.

A museum will initially have three floors, each of which consist of about ten rooms. In every room there are several showcases which can contain various archeological artefacts each. The designers (as well as the (scientific) data collectors) have the ability to design the museums as they. And of course the administrator can also manipulate the museums. For designing museums, it will be best to either offer a given set of different showcases, frames and boards to choose from or to give a list, where one can choose which entries of the data-record will be shown.

The museums are usually used for special exhibitions, so there will not be all the items from the database in the museum. The visitors have the ability to search here for archeoligical artefacts. As a result they will get a descritpion how to get to the desired item in the museum. If one selects an item in a showcase he is given the available data from the database.

1.1.6 Additional Notes

The customer wants to produce represantative CDs as well as catalogues of the virtual museum and its exhibitions. These CDs will only contain the museum itself (accesible through a web-browser) and the corresponding extract from the database. The customer himself is responsible for the production of this CD. So we will add a feature which automatically copys the museum-data in a certain directory, so that he can burn the contents of this directory on a CD-R. An equivalent feature will exist to create the catalogue.

Our management demands the use of XML in order to be as platform-independent as possible.

1.2 System Environment

1.2.1 Hardware

As mentioned above, ADS will be hosted by our company. We will be responsible for readiness for use and maintenance of the system and also accessibility via the internet. Therefore we need a web-server, on which we run ADS. Further we need a database-server, to implement the database for ADS. We shall use the latest hardware from a well-known and reliable vendor.
All the ADS-user needs, is a computer with connection to the internet (e.g. via modem).

1.2.2 Software

For our servers (web and database) we need a reliable and stable operating system (such as Solaris, HP-UX, Linux) and web- and database-server-software to run under this OS.
The demand of software for the ADS-users is very small. Only an OS and a web-browser is required.

1.2.3 Data

The database will initially contain about 4000 records (each has about 20 attributes) of archeological data. Every year there will be about 400 new records to enter the database. Since the customer himself is responsible to fill the database we don't need to know anymore.

1.2.4 Users

The system distinguishes between five user-groups. Four active groups (data collectors, scientific data collectors, designers and administrators) who can change the database and one passive group (visitors) who can only query the database. The visitors will be the ones accessing ADS most of the time (about 100 hits at a time), while (scientific) data collectors will use it approximately twice or three times per day (400 new entries per year plus manipulations of existing data). The administrator is expected to work very few with the system.

1.3 User Interfaces

See storyboards starting on page 22 as a first evaluation.

1.4 Functional Requests

The functional requests in this section are formulated using the formal language which is systematically described through the Backus-Naur-Form as the following:

- <Request> ::= <Request Name>: <Request Text>. Request Type: <Request Type>. Priority: <Priority>. Reason: <Reason>

- <Request Name> ::= <Text>
- <Request Type> ::= HAVE TO|SHOULD|MUST NOT
- <Priority> ::= 1|2|3|4|5|6|7|8|9|10
- <Reason> ::= <Text>
- <Text> ::= <Character><Text>|<Character>
- <Character> ::= a|...|z|A|...|Z|0|...|9|"|.|...|;

Priorities are ranked 1 for highest and 10 for lowest priority.

1.4.1 Database

- DB1: The database has to be able to handle the initial 4000 records plus 400 new records per year over several decades (every record consists of about 20 attributes, containing text and maybe multimedia-files)
- Priority: 1
- Reason: This is a crucial part of the software-system.

- DB2: Individual attributes of the data-records have to be lockable and whole data-records have to be hideable.
- Priority: 2
- Reason: Some attributes as well as whole records must not be search-/viewable by the visitors.

- DB3: The collected data has to be editable.
- Priority: 1
- Reason: One should have the possibility to edit already collected data in order to add or correct information.

- DB4: The System has to distinguish between different user-groups.
- Priority: 1
- Reason: This is necessary to deal with the rights the different users have.

- DB5: There has to be a group for each of the following: administrators, data collectors, scientific data collectors, designers and visitors.
- Priority: 1

- Reason: The administrators are necessary to maintain and repair urgent concerns. The other four groups are demanded by the customer.

- DB6: The group-management has to be extendable.
- Priority: 4
- Reason: For future releases or updates this can be very useful and time-saving.

- DB7: An artefact record will contain the following attributes
  - inventory-number
  - name
  - category
  - scientific text
  - size
  - material
  - the name of the person who found the artefact
  - picture
  - video
  - sound
- Priority: 1
- Reason: These are the essential attributes demanded by the customer.

- DB8: A grave-record will contain the following attributes
  - gps-data
  - gis-data
  - scientific text
  - the name of the person who found the grave
  - picture
  - video
  - sound
- Priority: 1
- Reason: These are the essential attributes demanded by the customer.

- DB8: A settlement-record will contain the following attributes
  - gps-data

  - gis-data
  - scientific text
  - the name of the person who found the settlement
  - picture
  - video
  - sound

- Priority: 1
- Reason: These are the essential attributes demanded by the customer.

- DB9: There will be a 1..n relationship between graves and artefacts. Thus, n artefacts can be associated with a grave.
- Priority: 1
- Reason: It is essential to link graves to artefacts.

- DB9: There will be a 1..n relationship between settlements and artefacts. Thus, n artefacts can be associated with a settlement.
- Priority: 1
- Reason: It is essential to link settlements to artefacts.

1.4.2 Web-Frontend

- WF1: There has to be a seperate section for the (scientific) data collectors which has to be protected by a password system.
- Priority: 1
- Reason: The section where the (scientific) data collectors can edit the database has to be seperated and protected from the visitors.

- WF2: The (scientific) data collectors have to be able to enter data in form of text into the databank, as well as upload images, audio- and videofiles.
- Priority: 1
- Reason: One of the crucial functions of the software-system is to allow the (scientific) data collectors to update the database via the web-frontend.

- WF3: There has to be a simple text-search function, which searches all the records and their attributes which contain text (except hidden records and locked attributes) for the entered search word.

- Priority: 1
- Reason: The visitor has to be able to search the database in a simple way.

- WF4: There has to be an advanced search function, which searches for specific attributes (except hidden records and locked attributes). The entries have to be connected with logical operators.
- Priority: 3
- Reason: The advanced visitor has to be able to search the database more specifically.

- WF5: There has to be a searchable knowledge tree which leads to the items.
- Priority: 5
- Reason: The visitor should be able to start his search in a general field of interest and specify his search as he moves up the knowledge tree.

- WF6: The web-frontend must not be based on additional plug-ins for web-browsers.
- Priority: 1
- Every visitor shall have the same access to our system without downloading and installing plug-ins.

1.4.3 Virtual Museum

- VM1: There has to be a Virtual Museum with initially 3 floors and 10 rooms per floor (there are several showcases in a room, where certain artefacts from the database are on display).
- Priority: 1
- Reason: This is a crucial part of the software-system. The customer has to be able to display certain artefacts.

- VM2: The visitors have to be able to browse through the museum.
- Priority: 1
- Reason: The visitors must be able to view the items that are on display in the showcases.
- VM3: The Virtual Museum has to be expandable and editable.
- Priority: 3

- Reason: The customer wants to edit the layout of the rooms and showcases. It should also be possible to add additional floors and museums.

- VM4: There has to be a search function in the virtual museum.
- Priority: 3
- Reason: There should be a way for the visitor to easily find the artefact he is interested in.

- VM5: The museum has to be able to host an exhibition concerning a special topic.
- Priority: 4
- Reason: This is a feature our customer demands.

- VM6: The museum has to be depicted by sophisticated graphics.
- Priority: 7
- Reason: Visitors should get an comfortable and eye-catching surface.

- VM7: The (scientific) data collectors have to be able to create representative CD-ROMs and printed catalogues of a special exhibition in the virtual museum.
- Priority: 2
- Reason: This is a feature our customer demands.

1.5 Non-Functional Requests

Factors such as the reaction time of our software-system depend very much on third-party's performances or other factors (e.g. internet-providers, rush-hours on the data-highway). In combination with the fact that our company uses the latest hardware and of course competent and efficient programming, it therefore makes little sense to give exact statements such as "The database gives an answer within 2.5 seconds". If the customer wants the final product to be faster than the prototype, we will try to do some improvements of performance if possible.

The stability of the system shall be reached by using high-quality hardware and a reliable operating system. Of course stability and usability also depends on the quality of our product.

The use of XML combined with the fact that no special plug-ins but just a plain web-browser are going to be used grant high portability. This is necessary in order to offer features such as producing CDs or catalogues.
XML will be an intermediary layer in the system, enabling us to publish content in HTML, PDF, or what ever format wished for by the customer.

1.6 Error Handling

1.6.1 Database

- DB-ERR1: The database doesn't react to queries from the web-frontend.
- Cause: The DB-server-daemon failed or isn't loaded.
- Solution: Generate an error message and inform the technical administrator.

- DB-ERR2: A query doesn't return any results although the desired information is definitely stored in the database.
- Cause: This might be due to malformed DB-queries.
- Solution: Our company has to go through these routines and fix the error.

- DB-ERR3: The database gives too much information (e.g. shows records or attributes which are actually locked).
- Cause: This might be due to malformed DB-queries.
- Solution: Our company has to go through these routines and fix the error.

- DB-ERR4: The (scientific) data collectors can't add or edit any information.
- Cause: The user is assigned to a wrong group or the rights of that group are not set correctly. The file-system gives a write-permission-error. The system resources are too low (e.g. not enough free disk-space).
- Solutions: The administrator has to assign the user to the right group or set the group rights correctly. The administrator has to check and correct the environment of the system.

1.6.2 Web-Frontend

- WF-ERR1: The web-frontend does not load.
- Cause: The hardware failed. The web-server-daemon failed. Third-party failure (such as the user's provider can't connect to our system).
- Solution: Contact the technical administrator.

- WF-ERR2: The web-frontend does not load properly. Parts of the page are missing.
- Cause: Files are missing or not set up correctly.
- Solution: Contact the webmaster.

1.6.3 Virtual Museum

- VM-ERR1: The Virtual Museum (or single parts of it) does not load.
- Cause: The server is down. Several files are missing.
- Solution: Contact the technical administrator.

1.7 Documentation Requirements

As our customer has no previous experience with similar software-systems and wishes to be independent after finishing the project, the product manual has to contain detailed instructions dealing with the following issues:

- Modification of the data-access-system. Explains how to
  - change the rights of certain groups
  - add users and groups
  - change the group of a user
- Modification of the Virtual Museum. Explains how to
  - add floors, rooms, showcases and artefacts to the museum
  - equip rooms and showcases with topics
  - change the layout of rooms and showcases
- Update the database. Explains how to
  - enter data-records into the database
  - upload files into the database

Because ADS isn't a commercial software-product and the customer didn't demand it, there's no use to make a printed handbook. The documentation will be only available as a Software-Document (such as Postscript, Adobe Acrobat, HTML, ...) from which one can make a hardcopy.

1.8 Acceptance Criteria

The most important criteria which have to be fulfilled so that our customer accepts our product are listed in the sections above. If we can deliver the product in time while implementing the reqired features, this should lead to a positive acceptance.

1.9 Risk Analysis

As an importanat part of the cost-benefit calculations we're performing a risk analysis for ADS. According to E. Yourdon [2] risk analysis should always be done, even if nobody asked for it. This is because it can be helpful to know which things can go wrong and thereby can be the cause for a lot of problems and additional costs.

1.9.1 Database

The expected amount of data in the database is rather modest and should not be a problem for a modern database management system. If the system should run short of storage space in the future, it will be no problem to provide additional disk space. Hence, the risk of problems with that part of the software-system can be classified as low.

1.9.2 Web-Frontend

The only risk connected with the web-frontend is, that the customer might not be satisfied with our design proposals. It will be important to show him some basic layouts in an early stage, to make clear what the customer is looking for and in doing so, avoid the risk of wasting too much time and manpower on unnecessary design work. The technical specifications of the web-frontend should not be a problem for our experienced developers.

1.9.3 Virtual Museum

Concerning the virtual museum, it has to be said that the customer has very loose ideas of what it should finally look like. This might lead to problems, if the customer is not satisfied with our developer's suggestions. To avoid the necessity of last-minute adjustments of the whole concept, this risk should be minimized by clarifying if our proposals meet his demands.

Another risk lies in the fact that the museum has to be viewable without the aid of additional plug-ins. On the one hand this might result in an increased need of time and work to find solutions for otherwise easily solveable problems (e.g. when using Flash- or Quicktime-plug-ins), on the other hand it minimizes the risk of compatibility problems.

1.9.4 Stability

As the system shall always be up and we are responsible for it's stability and readiness for use, this point is quite important to ensure smooth working. We therefore should use a well-known and reliable hardware and operating system for the servers as well as make a contract with a service-company for case that the hardware fails. The use of XML and possible choice of SQL as the basis for database lead to a quite platform-independent design of the core-components.

1.9.5 Personnel

As our project team is rather small, it can have a great influence on the software-development and on the project if somebody leaves the team. If we had to recruit new personnel this could cost a lot of time and money.
On the other side the facts, that within our team the project-leader changes with every new assignment and every team member has to work out an individual solution for the current assignment, have the effect that every team-member is on the same know-how-level. So, if one team-member will be disabled for just a little time, this will not handicap the whole team. But if three or more members drop out for the same time, the project is endangered.

1.9.6 Hardware

The only limitation to the used hardware is it's stability. If it turns out that there are not enough resources, such problems can easily be solved by adding more resources. As we have no limitations concerning financial investments, the hardware shouldn't be a problem.

1.9.7 Miscellaneous

Concerning some risks mentioned by E. Yourdon [2] and S. Vollmann [3] the following can be said:

- If there is much traffic on the internet, the response time may drop down. Since we can do nothing about this, we have to accept the risk of partially slow connections.
- In case of electric power breakdown, our power generator will start up, to ensure power supply for our servers. Customers won't recognice that at all.
- It remains negotiated with our customer who shall be held responsible for possible loss of data. Nevertheless, such a risk shall be minimized by making regular backups.
- It is the responsibility of our management to evaluate our cutomer's credit-worthiness.

1.10 Glossary and Index

Data-Access-System: An administrative concept to handle the rights of groups of users, defining their rights to read/write/modify data.

Knowledge Tree: A search method which starts with a general topic. The user can then choose between several branches and in doing so specify the topic he is interested in.

Router: A device use to interconnect two or more LAN's together, each of which operates with a different medium access control method - also known as a gateway or intermediate system.

TCP/IP: The complete suite of protocols including IP, TCP and the associated application protocols.

Chapter 2

Use Case Documentation

2.1 Introduction

As [4] states, when defining primary uses cases, it is not needed to capture "every detail of the use cases". Thus, use cases are used primarily to capture the high level user-functional requirements of a system (cf. [5])
Keeping this definition in mind, we design our use-case-model not primarily as a design-concept, but rather as an auxiliary means for communicating with the customer. The primary use-cases reflect the basic functionality of the ADS. Of course, the model is abstract and these primary use-case are constituted by a lot of sub-use-cases (e.g. Administrate is a very abstract use-case, consituting of CreateBackup, ManageRights, etc).

2.2 System Description

2.2.1 System-Level Use Case Diagram

see Figure 2.1 on page 20

2.2.2 Actors

- InternetUser

  The InternetUser is interested in viewing the data and museums. He ist the one visiting the webpage of the ADS and using the ADS for display-purposes

- Designer

  A Designer creates Virtual Museums. He can arrange floors, room and vitrinas, but is not necessarily able to enter data. Furthermore, a Designer is able to create publications, such as a CD-Rom or a catalogue.

- DataCollector

  A DataCollector is able to enter data concerning artefacts, graves and settlements into the system. DataCollectors (as they are generalized by Designers) can create and modify virtual museums.

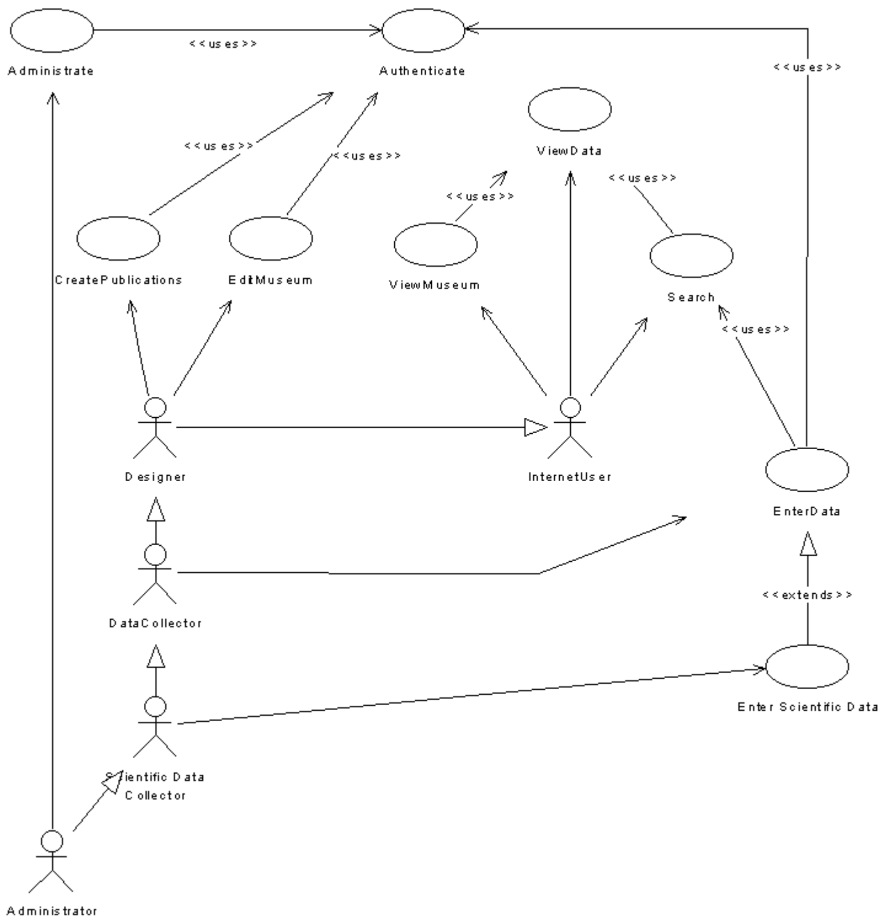


Figure 2.1: System-Level Use Case Diagram

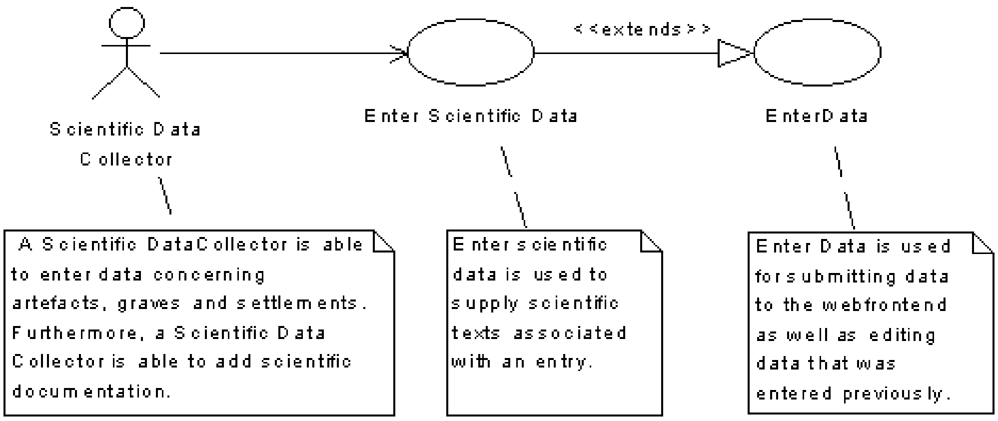


Figure 2.2: Actor: Scientific Data Collector, Use Case: Enter Scientific Data

- Scientific DataCollector

  A Scientific DataCollector is able to enter data concerning artefacts, graves and settlements. Furthermore, a Scientific DataCollector is able to add scientific documentation. Scientific Data Collectors (as they are generalized by DataCollectors) can create and modify Virtual Museums.
  For reference purposes the Use-Case: Enter Scientific Data has been added for this actor - see Figure 2.2 on page 21)

- Administrator

  An administrator has all the rights of the other groups, i.e. he is able to enter data, scientific, etc. Furthermore he has the privilege to edit rights and groups.

- Database

  Took into consideration to be an actor, because it is not part of the system but interacts closely with it.

2.2.3 Primary Use Cases

The use-cases will be explained in more detail in section 2.3.

- Authenticate
- View Data
- Enter Data
- Enter Scientific Data
- Search

- View Museum
- Edit Museum
- CreatePublications
- Administrate

2.3 Detailed Use Case Description: Primary Use Cases

2.3.1 Use Case: Authenticate

Authenticate is used to identify a user and associate him with a group with more rights than an InternetUser, i.e. Designer, DataCollector, Scientific DataCollector, Administrator.

Actors

- Designer
- DataCollector
- Scientific DataCollector
- Administrator

Priority

Priority: 3
Reason: The authentication can be implemented after other use-cases have been done.

Status

Filled

Preconditions

- User connected to web-front-end
- User clicked on login-button.

Postconditions

- The respective actor is authenticated as Designer, DataCollector, Scientific DataCollector or Administrator or authentication fails.

Extension Points

None.

“Used” Use Cases

None.

Flow of Events

Primary Scenario

1. Actor selects ”Login”
2. Actor supplies Login-name
3. Actor supplies Password
4. Login-Name and Password match
5. User-Data (and associated group) is retrieved from Data-Base
6. User is logged in and granted rights according to his group-association

Secondary Scenario

1. Actor selects ”Login”
2. Actor supplies Login-name
3. Actor supplies Password
4. Login-Name and Password do not match
5. System displays an error-message
6. User is not logged in

Activity Diagram

See figure 2.3, page 24.

User Interface

- Input-field for login-name
- Input-field for password (hidden input!)
- submit-button

See figure 2.4, page 25.

Sequence Diagrams

Not relevant to this task.

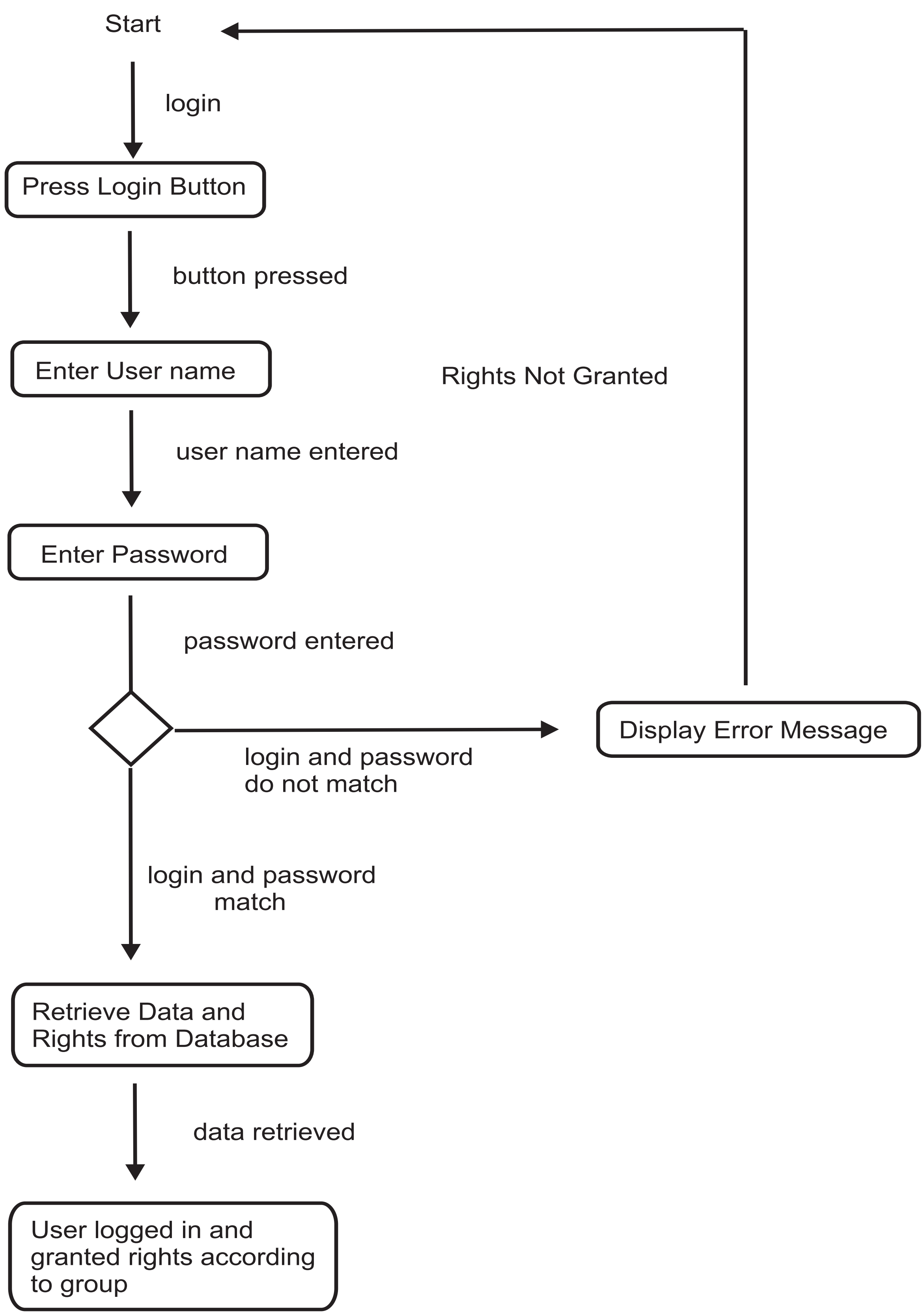


Figure 2.3: Activity diagram for use case "Authenticate"

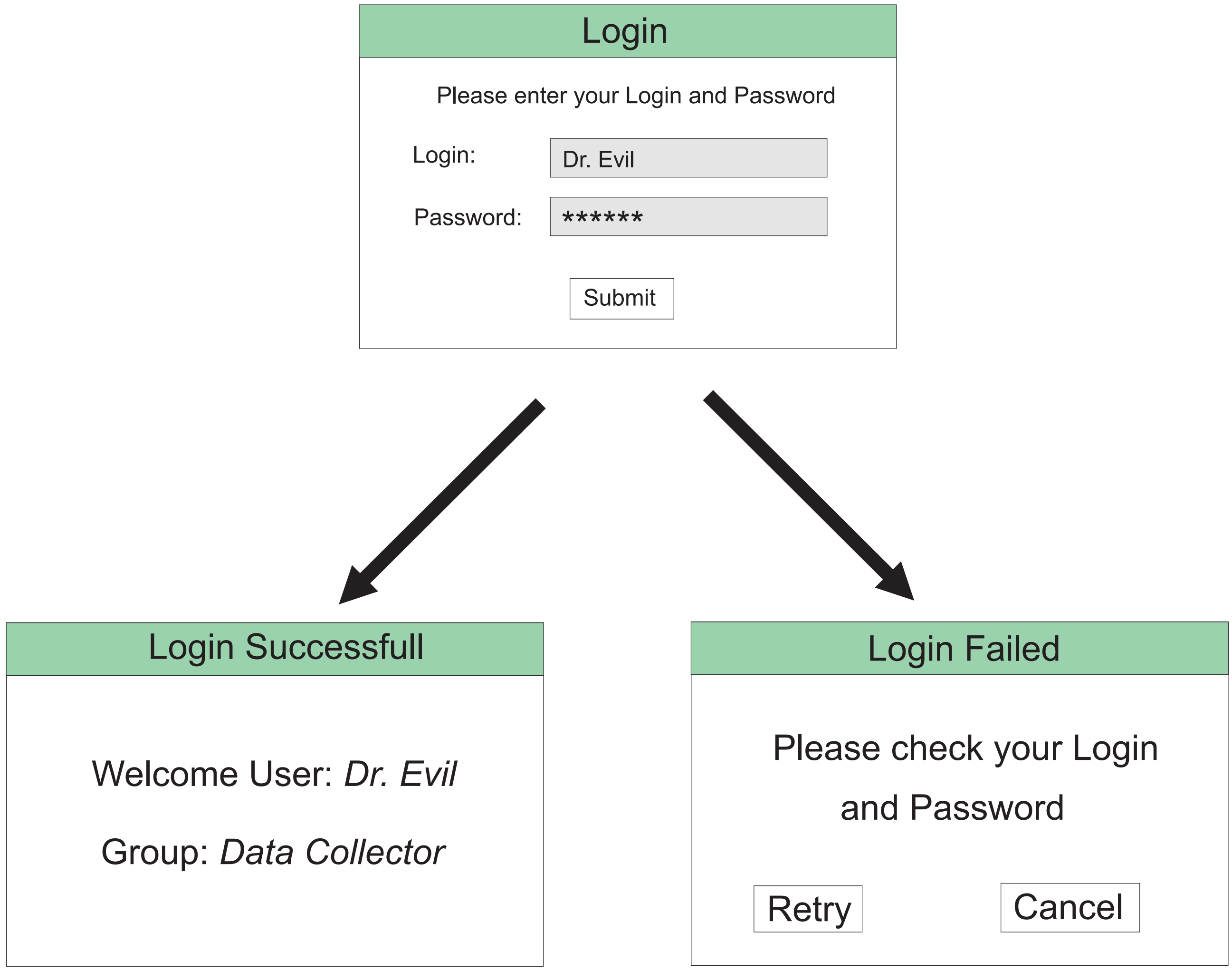


Figure 2.4: Story board for use case "Authenticate"

Subordinate Use Cases

Not relevant to this task.

View of Participating Classes

Not relevant to this task.

Other Artifacts

None.

Other Requirements

None.

2.3.2 Use Case: Enter Data

Enter Data is used for submitting data to the webfrontend as well as editing data that was entered previously.

Actors

- Data Collectors (and thus also Scientific DataCollectors as they are generalized by DataCollectors, and Administrators as they are generalized by Scientific DataCollectors) [1]

Priority

Priority: 1 Reason: Entering Data is an essential part of the system.

Status

Filled

Preconditions

- User has correctly logged on to the system. See use case "Authenticate".

Postconditions

- Data is stored in the database or use-case fails or is cancelled. In any case the actor will be presented with the normal logged-in screen.

[1] In the description of further use cases, if not stated otherwise, it will be presumed that if an actor is generalizes by another actor, the former inherits the rights of the latter. (Cf. Figure 2.1 on page 20 for "generalize"-hierarchy of actors)

Extension Points

none

“Used” Use Cases

- Use Case Authentication
- Use Case Search

Flow of Events

Primary Scenario

1. The actor selects ”Enter Data”
2. if (Actor wants to enter data based on an already existing object)
   (a) Actor searches for object using the use case ”Search” (this means by name or by inventary-number).
   (b) if (Actor accepts an object and wants to add a new one based on this existing one)
      i. Copy data to editing-/entering-mask
      ii. Give object new inventary-number
   (c) if (Actor accepts an object and wants to edit the existing data)
      i. Copy data to editing-/entering-mask
3. if (Actor wants to enter completely new data)
   (a) Actor decides if the new object is an artefact, a settlement or a grave
   (b) Empty editing-/entering-mask is opened
4. Actor enters data into and/or edits already existing data in the user-interface
5. Actor presses submit-button
6. Consistency-checks (as no inventary-number or file-names may occur twice in the system, ...)
7. if (Consistency-Error)
   (a) Give error-message and - according to the error - go back to editing-/entering-mask (inventary-number) or prompt the actor to
      - Overrule and save
      - Keep already existing data (e.g. multimedia-file) wich conflicts, but update the rest
      - Cancel upload and go back to editing-/entering-mask
8. System submits data to the database

9. Data is stored
10. Multimedia files are stored using the filesystem
11. A confirmation is displayed

Secondary Scenario
The actor can cancel the process at any time when interacting with the system using the use case "Enter Data". The use case will end at this point and the system will display the standard-screen for logged-in actors.
As the use case "Search" is integrated into "Enter Data", cancelling the process of adding data while performing a search will be dealt by the cancel-options of "Enter Data" and therefore lead to the same result as just mentioned above.

Activity Diagram

See figure 2.5, page 29.

User Interface

See figures 2.6, page 30 and 2.7, page 31.

Sequence Diagrams

Not relevant to this task.

Subordinate Use Cases

Not relevant to this task.

View of Participating Classes

Not relevant to this task.

Other Artifacts

Not relevant to this task.

Other Requirements

Not relevant to this task.

2.3.3 Use Case: Enter Scientific Data

Opposing to "Enter Data", "Enter Scientific Data" is used to supply scientific texts and additional information associated with an object.
Also, the system provides a function "Enter Scientific Data" instead of "Enter Data" as soon as an actor has logged in.

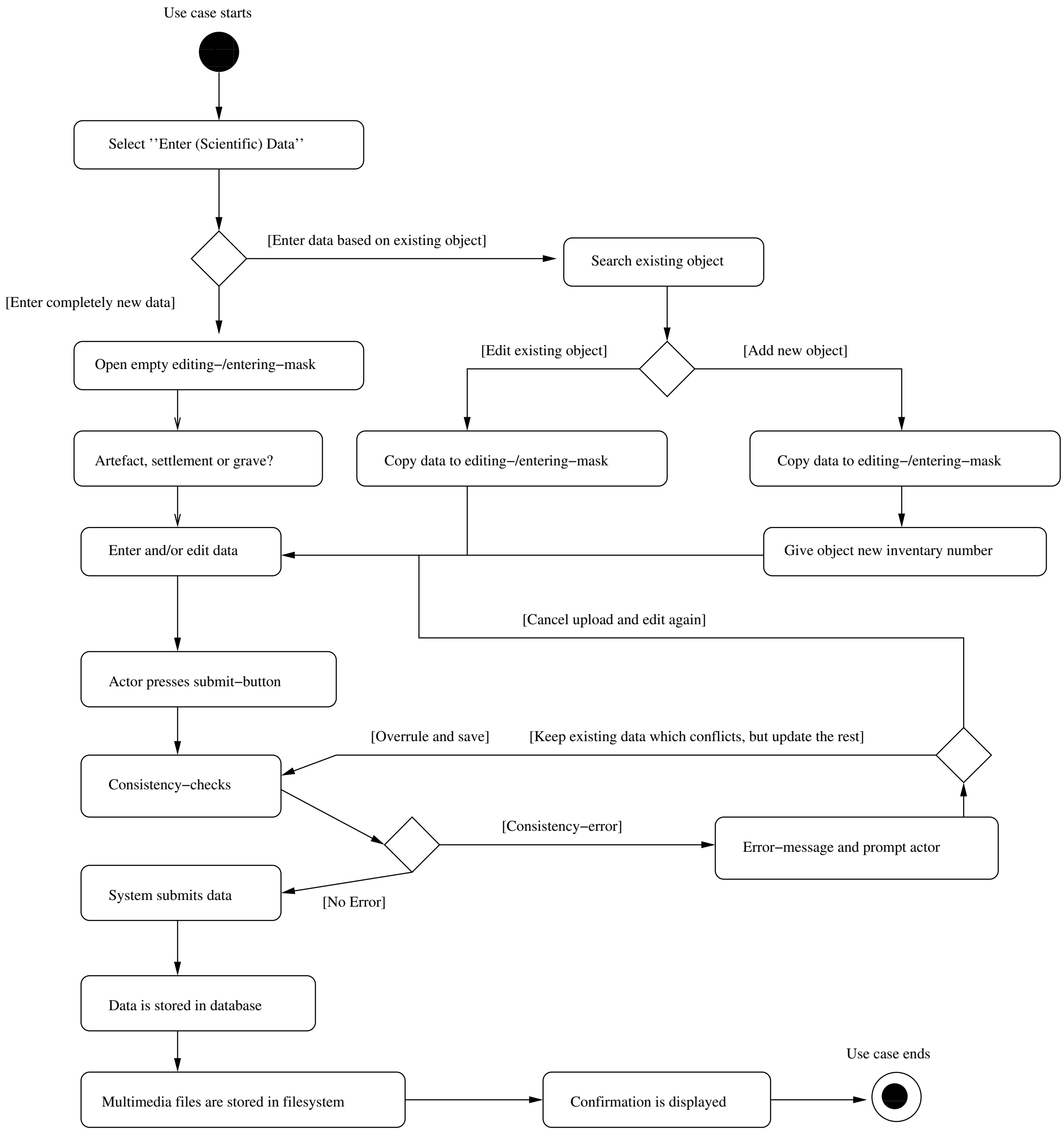


Figure 2.5: Activity diagram for use cases "Enter Data" and "Enter Scientific Data"

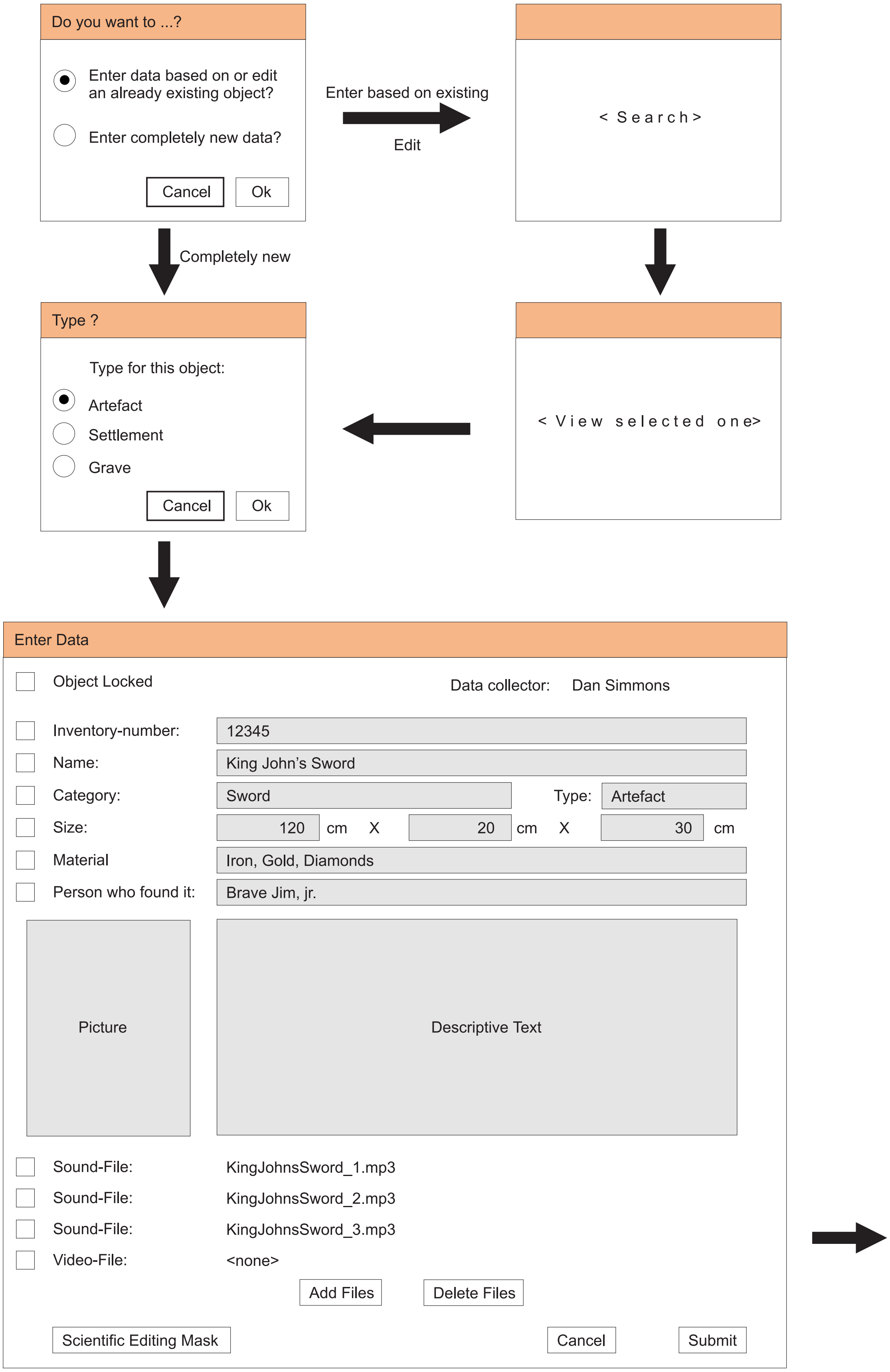


Figure 2.6: Storyboard for use cases "Enter Data" and "Enter Scientific Data", 1st figure

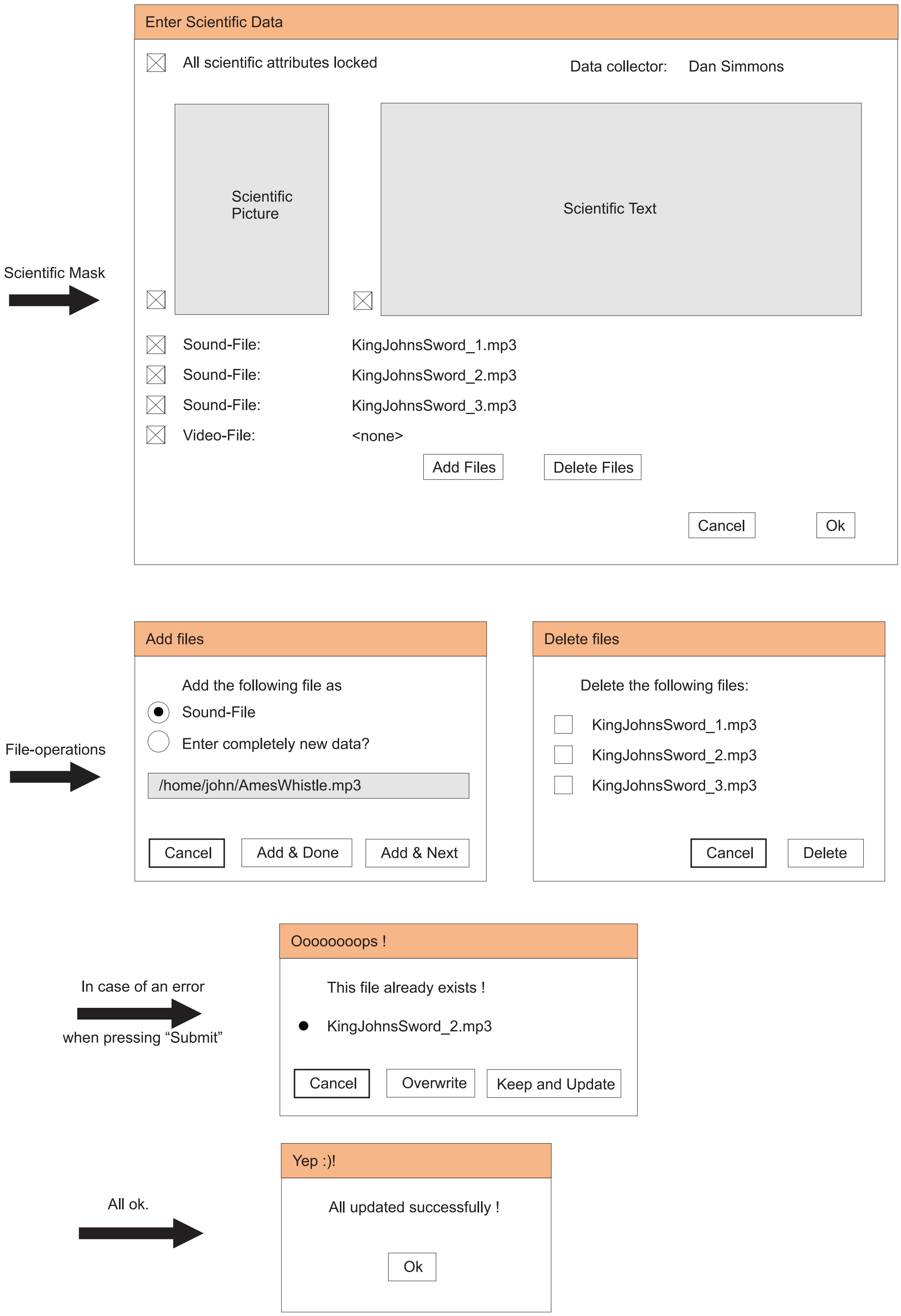


Figure 2.7: Storyboard for use cases "Enter Data" and "Enter Scientific Data", 2nd figure

Actors

- Scientific Data Collectors

Priority

Priority: 1 Reason: Entering Data is an essential part of the system.

Status

Filled

Preconditions

- User has correctly logged on to the system. See use case "Authenticate".

Postconditions

- Data is stored in the database or use-case fails or is cancelled. In any case the actor will be presented with the normal logged-in screen.

Extension Points

Enter Data.

"Used" Use Cases

- Use Case Authentication
- Use Case Search

Flow of Events

<u>Primary Scenario</u>

1. The actor selects "Enter Scientific Data"
2. if (Actor wants to enter data based on an already existing object)
   (a) Actor searches for object using the use case "Search" (this means by name or by inventary-number).
   (b) if (Actor accepts an object and wants to add a new one based on this existing one)
      i. Copy data to scientific editing-/entering-mask
      ii. Give object new inventary-number
   (c) if (Actor accepts an object and wants to edit the existing data)
      i. Copy data to scientific editing-/entering-mask
3. if (Actor wants to enter completely new data)

   (a) Actor decides if the new object is an artefact, a settlement or a grave
   (b) Empty scientific editing-/entering-mask is opened

4. Actor enters data into and/or edits already existing data in the user-interface

5. Actor presses submit-button

6. Consistency-checks (as no inventary-number or file-names may occur twice in the system, ...)

7. if (Consistency-Error)

   (a) Give error-message and - according to the error - go back to editing-/entering-mask (inventary-number) or prompt the actor to
      - Overrule and save
      - Keep already existing data (e.g. multimedia-file) wich conflicts, but update the rest
      - Cancel upload and go back to editing-/entering-mask

8. System submits data to the database

9. Data is stored

10. Multimedia files are stored using the filesystem

11. A confirmation is displayed

Secondary Scenario
The actor can cancel the process at any time when interacting with the system using the use case "Enter Scientific Data". The use case will end at this point and the system will display the standard-screen for logged-in actors.
As the use case "Search" is integrated into "Enter Scientific Data", cancelling the process of adding data while performing a search will be dealt by the cancel-options of "Enter Scientific Data" and therefore lead to the same result as just mentioned above.

Activity Diagram

See figure 2.5, page 29.

User Interface

See figures 2.6, page 30 and 2.7, page 31.

Sequence Diagrams

Not relevant to this task.

Subordinate Use Cases

Not relevant to this task.

View of Participating Classes

Not relevant to this task.

Other Artifacts

Not relevant to this task.

Other Requirements

Not relevant to this task.

2.3.4 Use Case: ViewData

This use-case is responsible for displaying single items of data on the screen. Data should be hided from unauthorized actors, so view data needs to check users rights to display data in a correct way. Every usergroup should be supplied with its specific working frames.

Actors

- InternetUser
- Designer
- DataCollector
- Scientific DataCollector

Priority

Priority: 1
Reason: Work without displaying data is not possible.

Status

- Filled

Preconditions

- Use case supplied with data.

Postconditions

- Data diplayed on screen, or
- Error message is displayed.

Extension Points

- None.

“Used” Use Cases

- None.

Flow of Events

Primary Scenario

1. Actor calls Use Case
2. **IF** (Actor is Intenet user)
   (a) Check if item is locked
   (b) **IF** (Item locked)
      i. Send message info
   (c) **IF** (Item not locked)
      i. Read formating information out of the database
      ii. Filter all attributes that are locked(replace them by a default info)
      iii. Display frame
3. **IF** (Actor is staff)
   (a) **IF** (Actor is Administrator)
      i. **IF** (Search)
         A. Load data
      ii. Display Frame
   (b) **IF** (Actor is Designer)
      i. **IF** (View Museum)
      ii. see page 38
   (c) **IF** (Search)
      i. Load data
   (d) Display Frame
   (e) **IF** (Actor is DataCollector)
      i. **IF** (Search)
         A. Load data
      ii. Display Frame
      iii. **IF** (Scientific Data Collector)
         A. Diplay frame

Secondary Scenario

1. Actor calls Use Case.
2. System is not reachable.

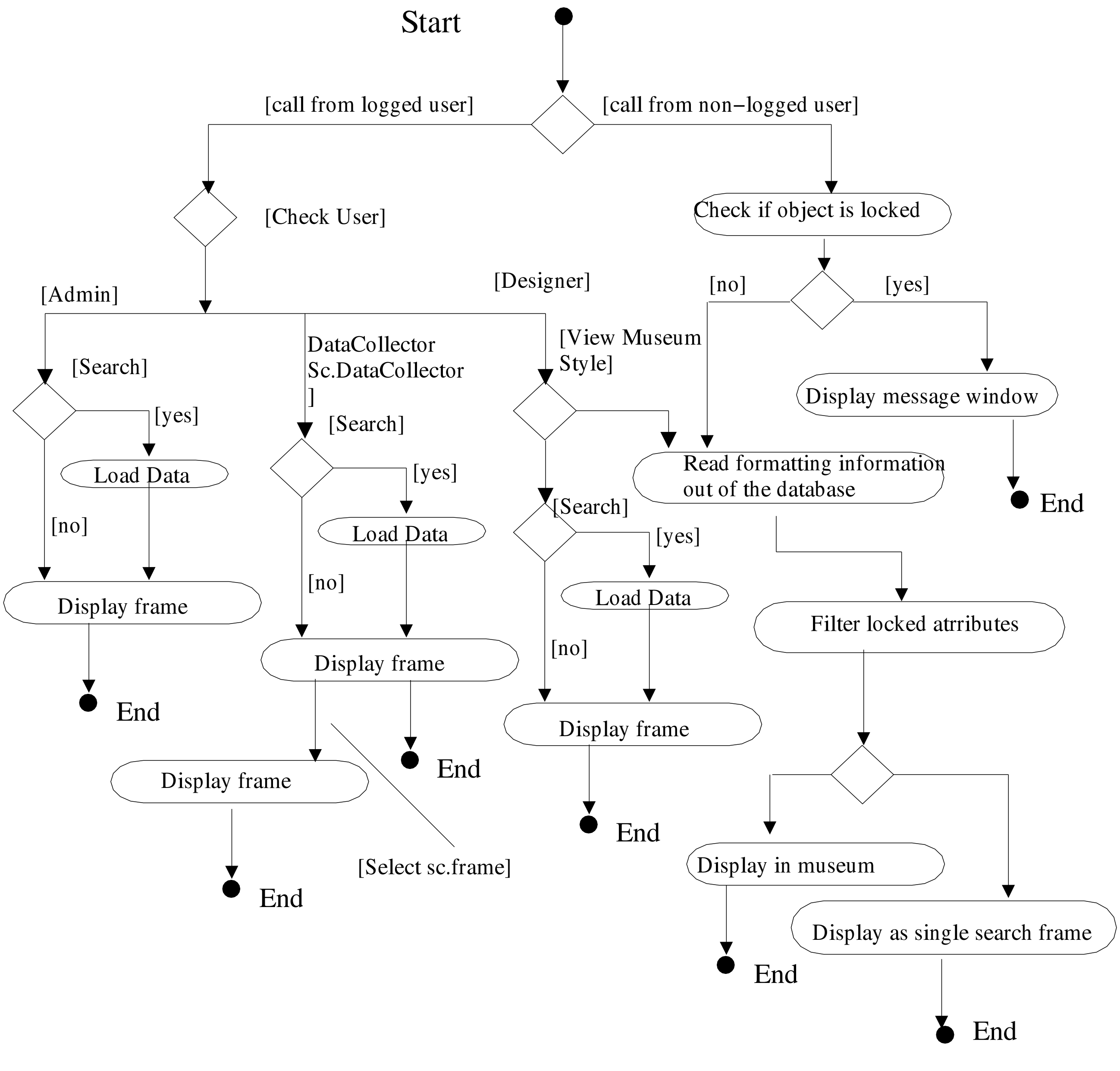


Figure 2.8: ViewData

Activity Diagram

see Figure 2.8, page 36.

User Interface

See user interfaces (storyboards) of the specific use cases.

Sequence Diagrams

Not relavant to this task

Subordinate Use Cases

None.

View of Participating Classes

None.

Other Artifacts

None.

Other Requirements

none

2.3.5 Use Case: ViewMuseum

This use case refers to the process of an actor viewing a museum. The actor is able to navigate through floors, enter rooms, look at show-cases and select items from show-case for more detailed display.

Actors

- InternetUser

Priority

Priority: 3
Reason: Museum will be implemented after several other use-cases have been implemented.

Status

- Filled.

Preconditions

- Actor selects "View Museum"

Postconditions

- Museum or an error-message is displayed.

Extension Points

None.

"Used" Use Cases

- ViewData.

Flow of Events

Primary Scenario

1. Actor clicks on "View Museum".
2. Actor can select one main topic(floor) of interest.
3. Actor can enter a specific room or change to the next/prvious floor.
4. Actor can view at vitrinas of interest.
5. Actor can simply navigate to the next/previous vitrina by clicking on the assigned arrows.

Secondary Scenario A

1. Actor clicks on "View Museum".
2. No museum is aktiv.
3. An error message is displayed.

Secondary Scenarion B

1. Actor clicks on "View Museum".
2. Actor can leave the museum any time(closing the browser, entering another URL).

Secondary Scenario C

1. Actor klicks on "View Museum".
2. System does not respond.
3. An error message is displayed.

Activity Diagram

see Figure 2.9, page 39.

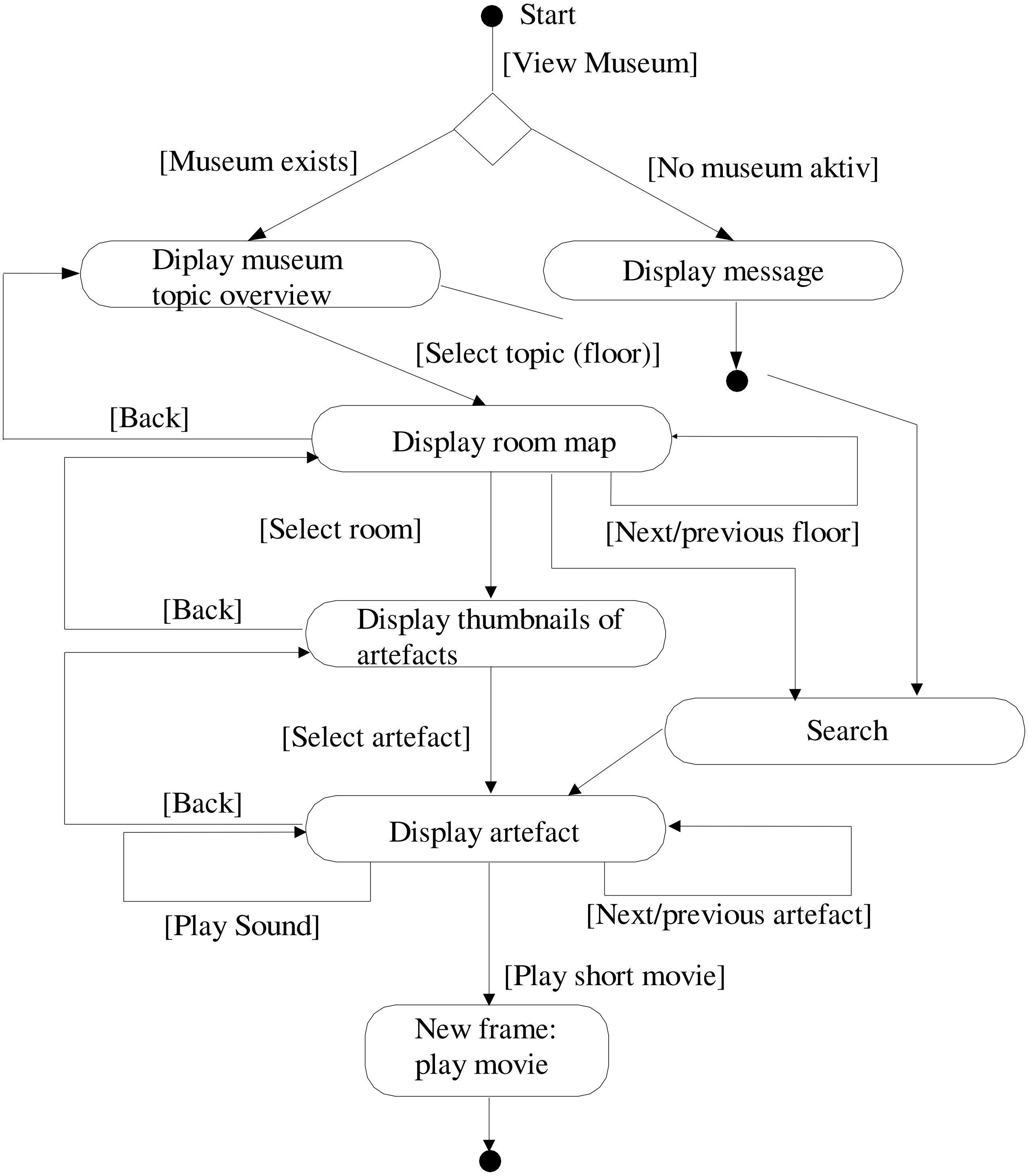


Figure 2.9: ViewMuseum

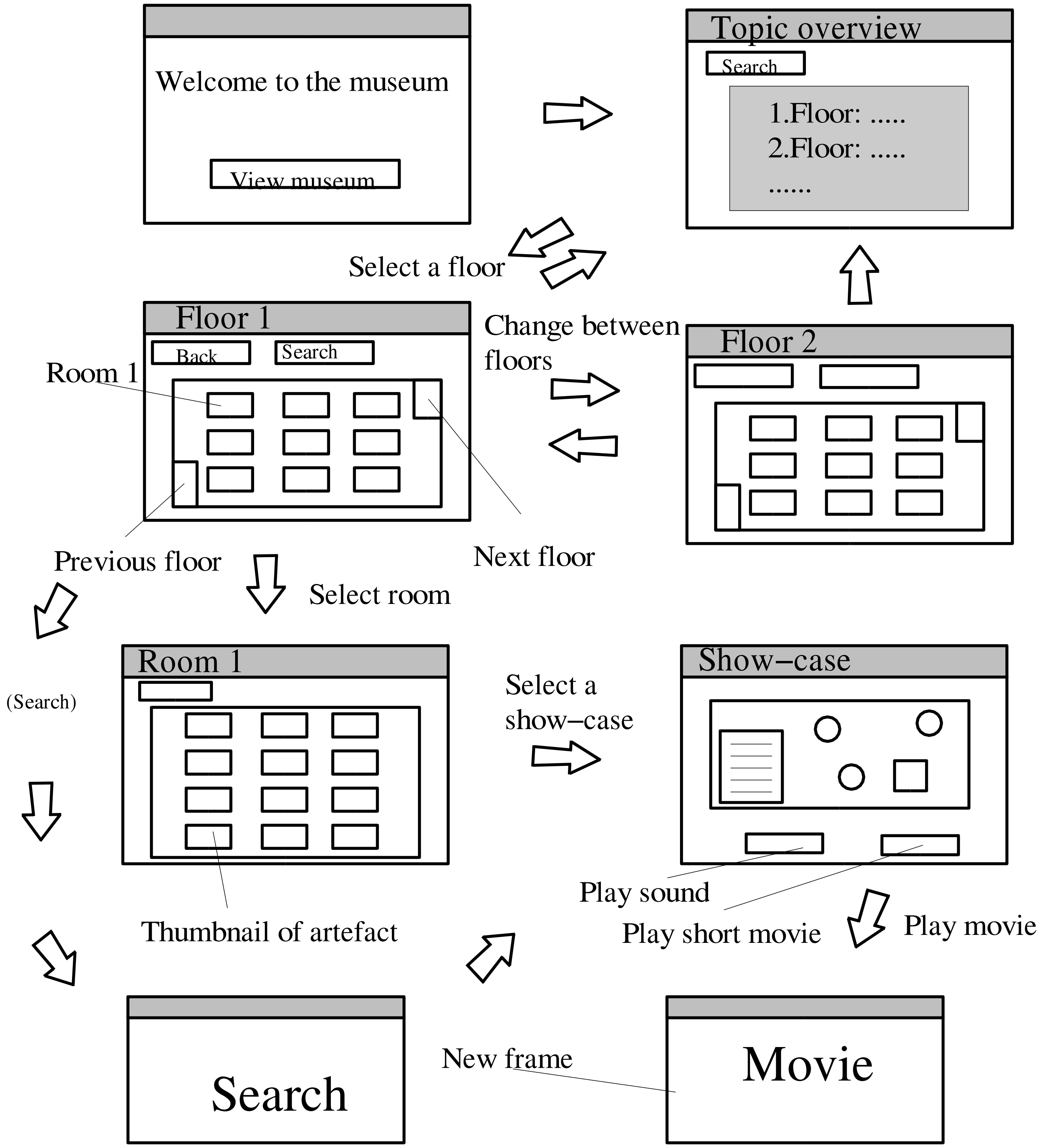


Figure 2.10: ViewMuseum

User Interface

see Figure 2.10, page 40.

Sequence Diagrams

None.

Subordinate Use Cases

None.

View of Participating Classes

None.

Other Artifacts

None.

Other Requirements

None.

2.3.6 Use Case: Edit Museum

Actors

- Designer
- Database

Priority

Priority: 3

Status

Filled

Preconditions

- "Edit Museum" is clicked by the actor
- Actor has the right to edit museums

Postconditions

- Museum data is edited or an error-message is displayed

Extension Points

ErrorHandling

"Used" Use Cases

Authenticate

Flow of Events

This account of the flow of events represents a macro-view of the events. Refer to the activity-diagram (Figure 2.1 on page 20) for a more detailed view.

Primary Scenario

1. Actor selects a museum (if there are several)
2. System checks actor's rights
3. Actor has the right to edit museum
4. Museum data is displayed
5. Actor modifies data (i.e. creates/modifies rooms, vitrinas, or delete museum)
6. Actor presses "submit"
7. System updates Museum

Secondary Scenario A

1. System checks actor's rights
2. Actor has the right to edit museum
3. No museum is established yet
4. System adds museum
5. Flow of events continues as from step 4. in primary scenario

Secondary Scenario B

1. Actor selects a museum (if there are several)
2. System checks actor's rights
3. Actor does not have the right to edit museum
4. An error-message is displayed

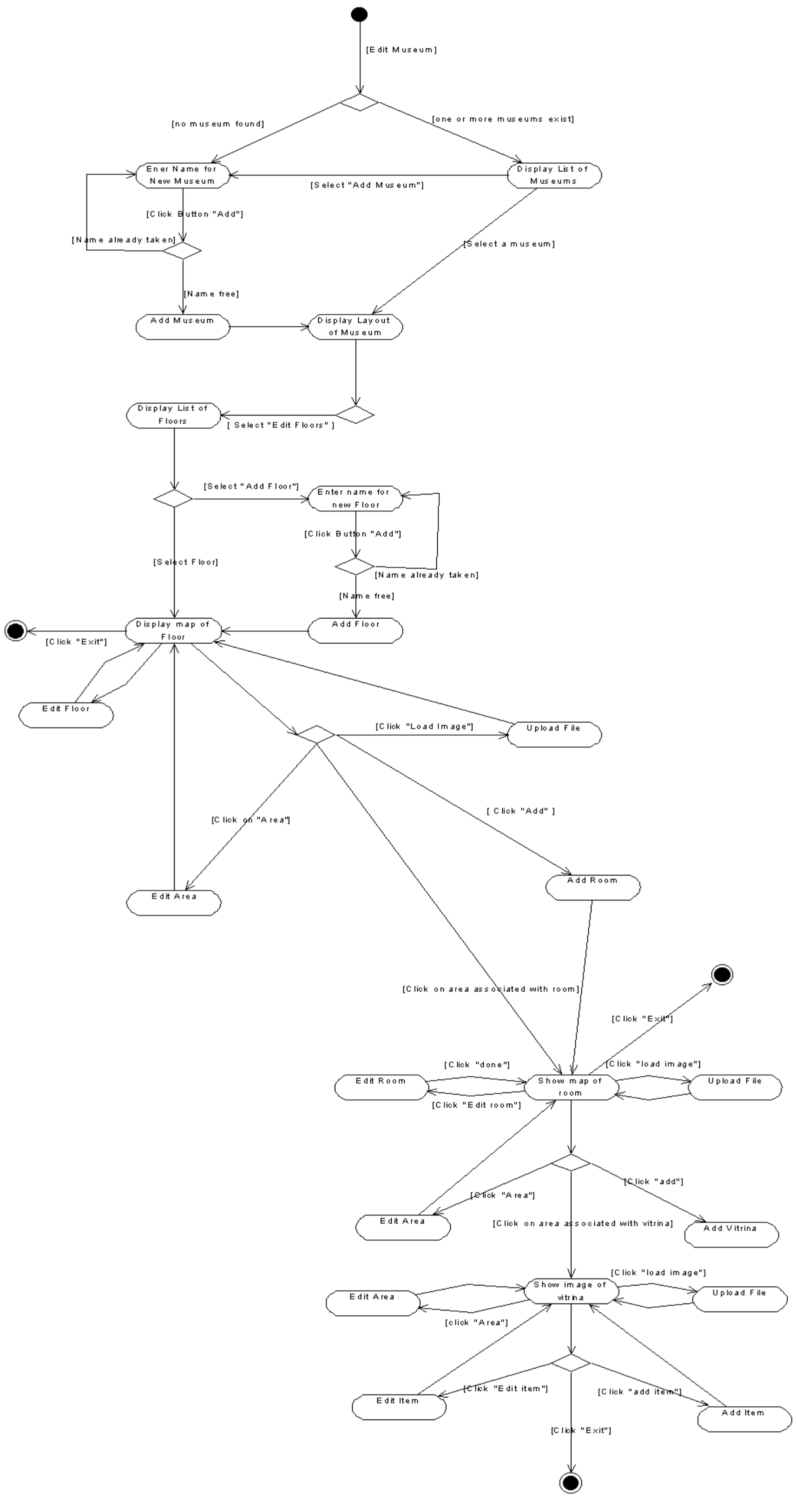


Figure 2.11: Edit Museum

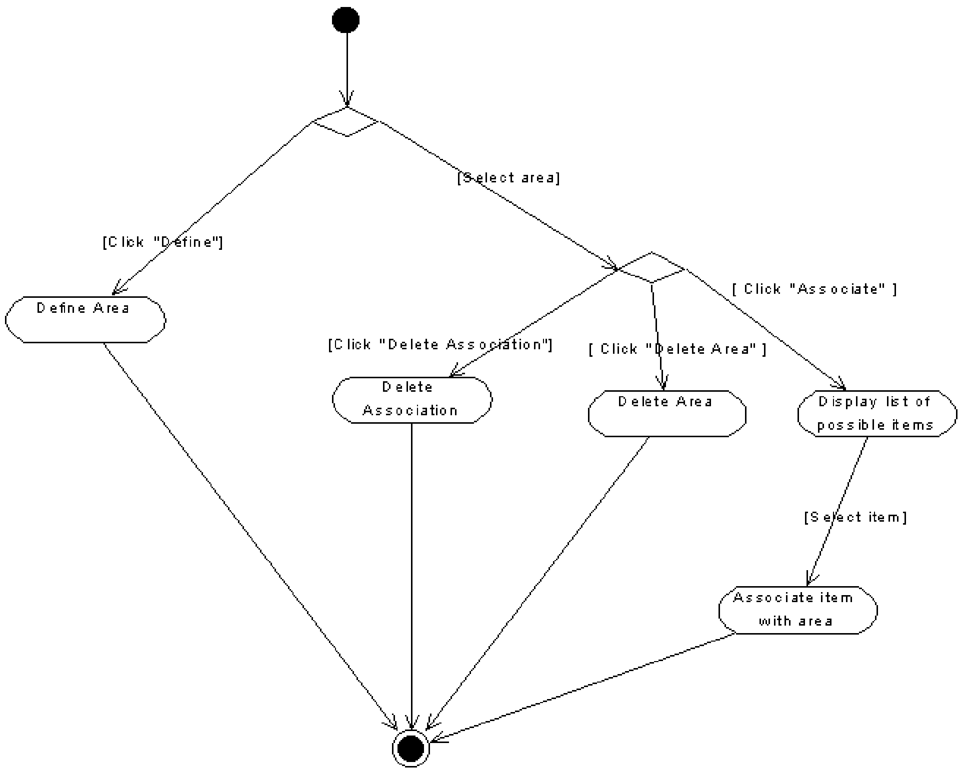


Figure 2.12: Edit Area

Activity Diagram

see Figure 2.11 on page 43

User Interface

see Figure 2.13 on page 46

Sequence Diagrams

None.

Subordinate Use Cases

- AddMuseum.

View of Participating Classes

None.

Other Artifacts

None.

Other Requirements

None.

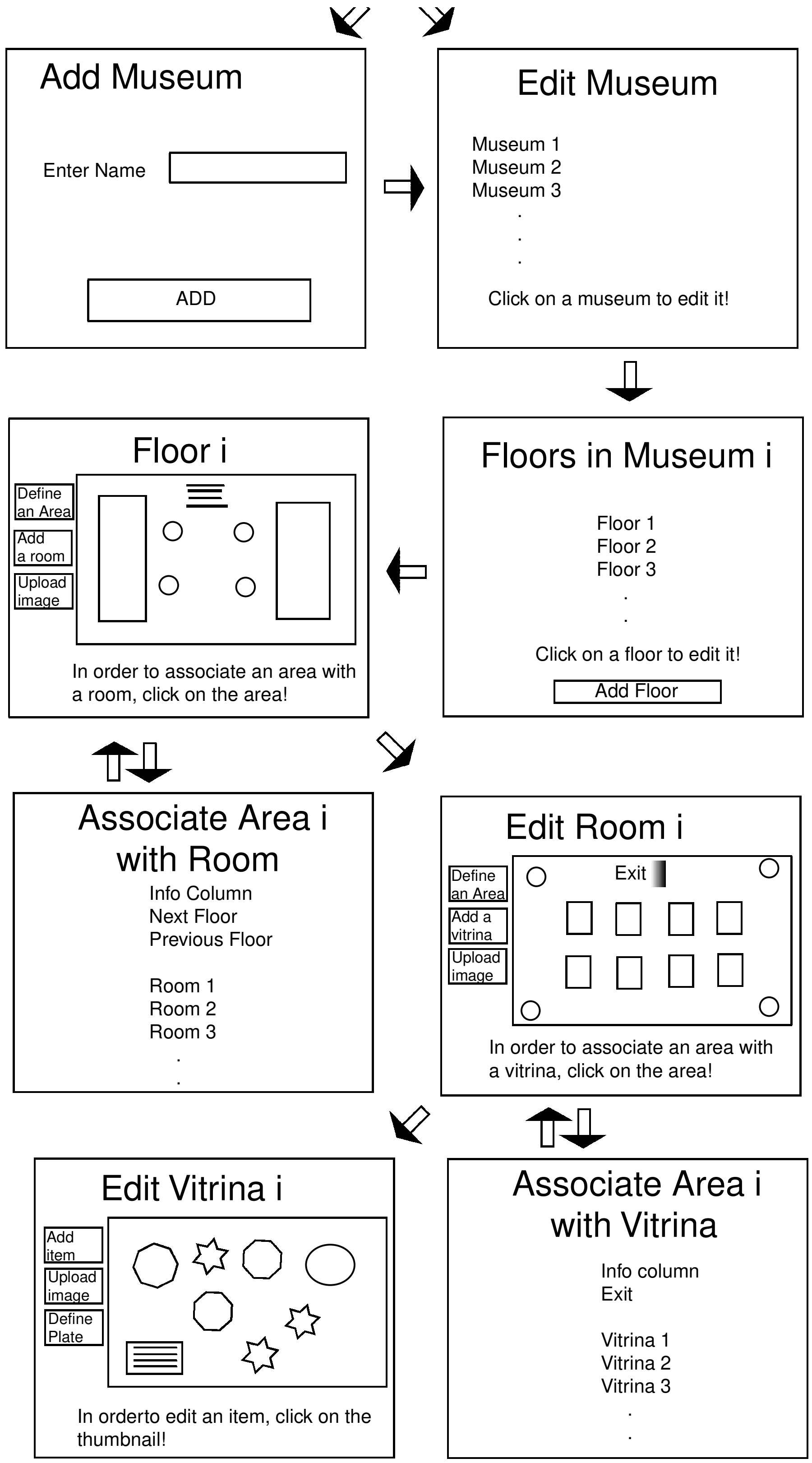


Figure 2.13: Storyboard EditMuseum

2.3.7 Use Case: SearchData

The actor searches the whole data he is entitled to see or searches a museum for specific entries. There are two kinds of searches: basic search and complex search.

Actors

- InternetUser
- Designer
- Data Collector
- Scientific Data Collector
- Administrator

Priority

Priority: 1
Reason: Searching the museum will make navigation a lot easier for the users

Status

- Filled

Preconditions

- Actor selects "Search data"

Postconditions

- System displays search results or an error-message

Extension Points

None.

"Used" Use Cases

- ViewData

Flow of Events

1. Actor selects "Search data"
2. Actor chooses query-mode ("simple" or "complex")
3. **IF** (query-mode is "simple")
   (a) Display simple search mask (only one search field)
   (b) **WHILE** (no keyword has been entered)

i. Actor enters a keyword to search for

(c) **END**

(d) Actor activates search button

4. **IF** (query-mode is "complex")
   (a) Display complex search mask (all attributes have a search field of their own)
   (b) **WHILE** (not at least one search criteria has been entered)
       i. Actor enters criteria to search for
   (c) **END**
   (d) Actor activates search button
5. System compares search criteria to all non-locked db-entries
6. **IF** (search criteria matches any db-entry)
   (a) Display all non-locked, matching items as a search-result
   (b) **IF** (any matching items are exhibited in a virtual museum)
       i. Display the location in the museum of this item additionally
   (c) Actor selects an item of the search-result to be displayed (→ Use-Case: ViewData or Use-Case: ViewMuseum)
7. **IF** (search criteria matches no db-entry)
   (a) Display error message

Activity Diagram

see Figure 2.14

User Interface

see Figure 2.15

Secondary Scenarios

- The Actor can return to the previous point of the FOE at any time.
- The Actor can cancel the search at any time.
- The search result contains only data which the Actor is allowed to see (according to his user-group).

Sequence Diagrams

None.

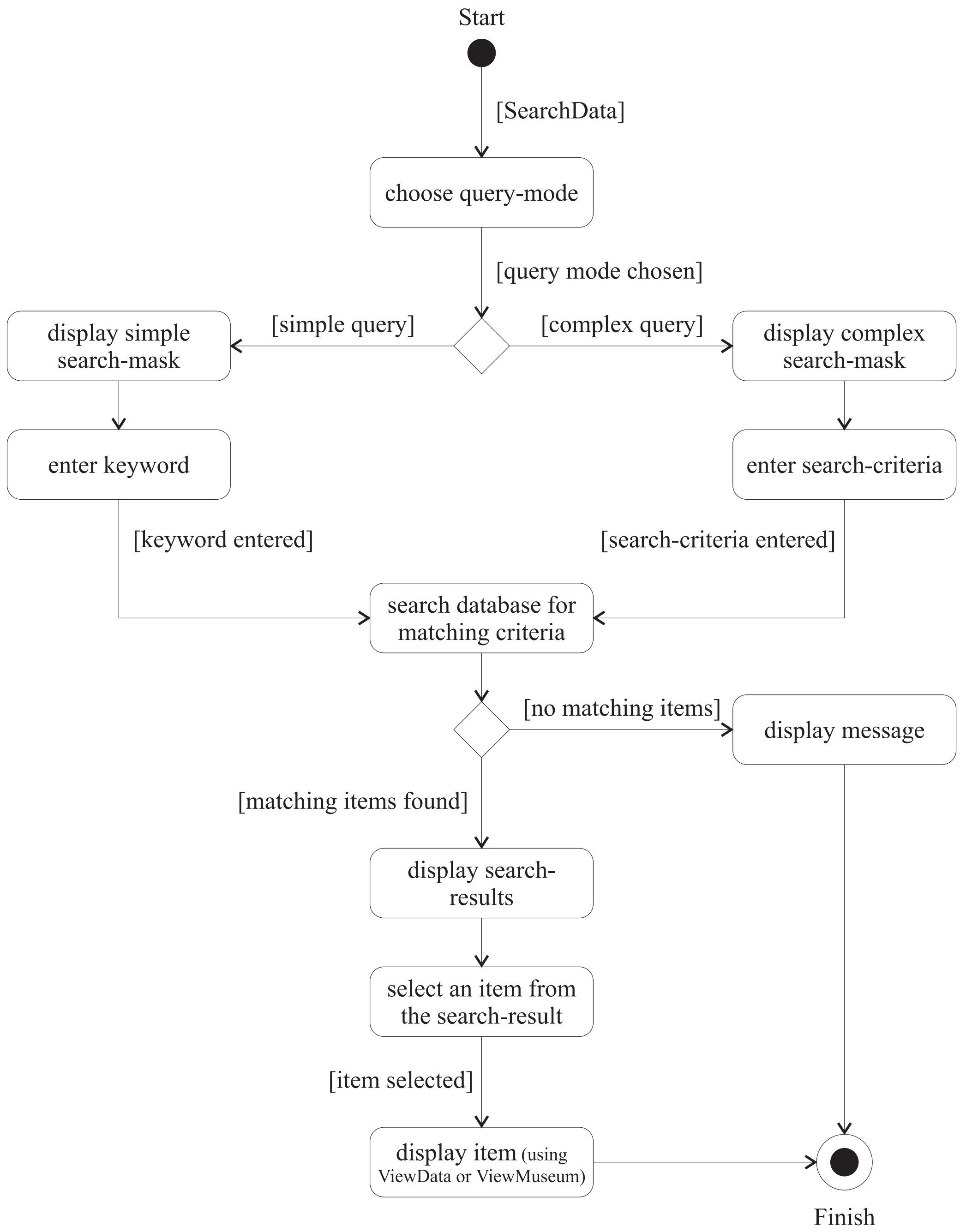


Figure 2.14: SearchData

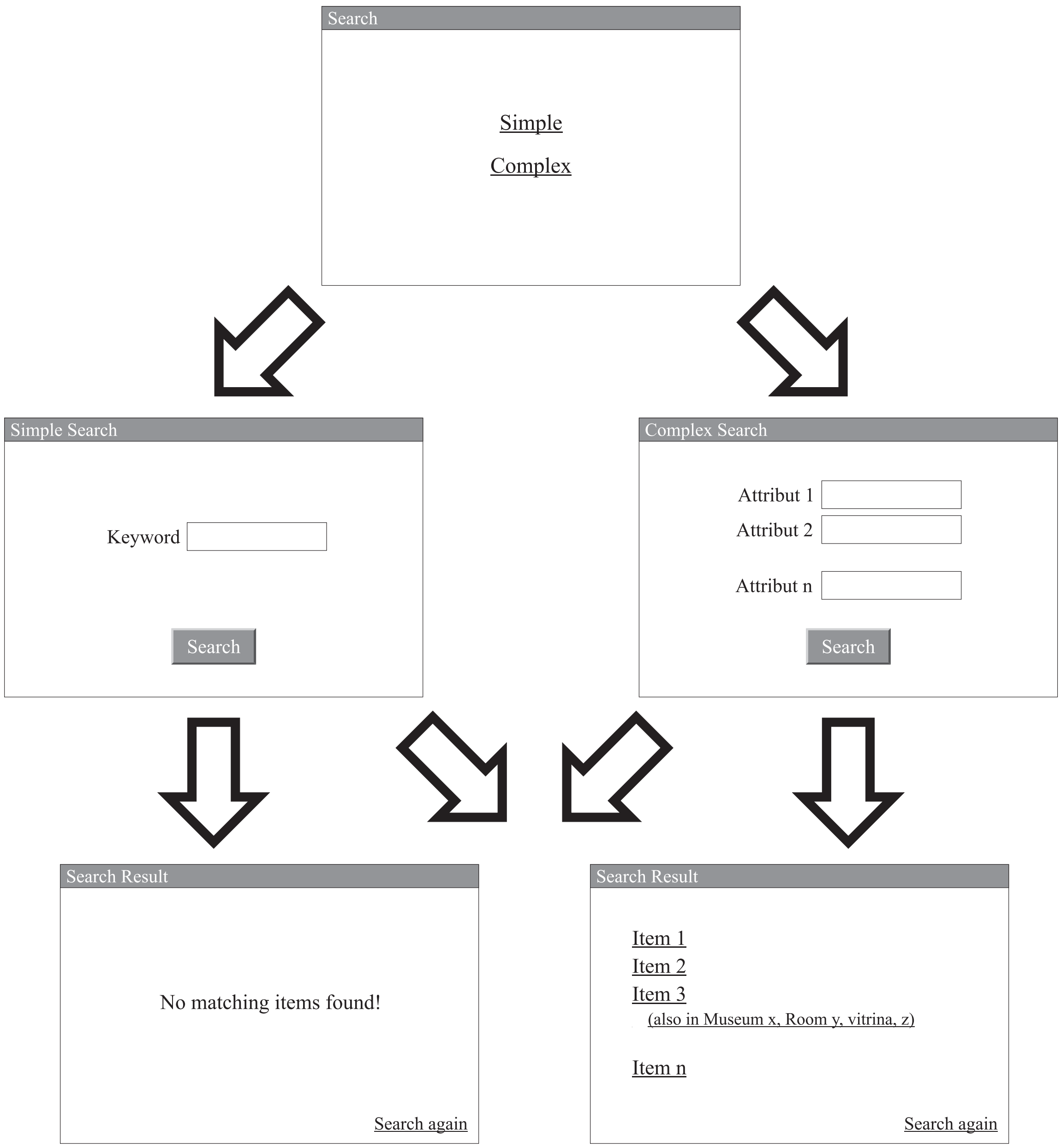


Figure 2.15: SearchData

Subordinate Use Cases

SearchData, SearchMuseum
N.B. These use cases will be documented in a later stage when we are dealing with use-cases in more detail.

View of Participating Classes

None.

Other Artifacts

None.

Other Requirements

None.

2.3.8 Use Case: CreatePublications

The actor selects a museum or a part of the data in order to publish it in a catalogue or a CD-Rom.

Actors

- Designer

Priority

Priority: 5
Reason: The creation of publications can be easily added after the main system is designed.

Status

- Filled

Preconditions

- Actor has to be authenticated as a Designer
- Actor selects "Publish Museum"

Postconditions

None.

Extension Points

None.

"Used" Use Cases

None.

Flow of Events

1. Actor selects "Publish Museum"
2. Actor chooses between publishing a CD-ROM or a catalogue
3. **IF** (more than one museum exists)
   (a) Display a list of museums
   (b) Actor chooses a museum to publish
4. Actor chooses output format
5. System exports data to the desired format
6. **IF** (creating a CD-ROM)
   (a) System creates a CD-image-file of the exported data

Activity Diagram

see Figure 2.16, page 53

User Interface

see Figure 2.17, page 53

Secondary Scenarios

- The Actor can return to the previous point of the FOE at any time.
- The Actor can cancel the publishing at any time.

Sequence Diagrams

None.

Subordinate Use Cases

None.

View of Participating Classes

None.

Other Artifacts

None.

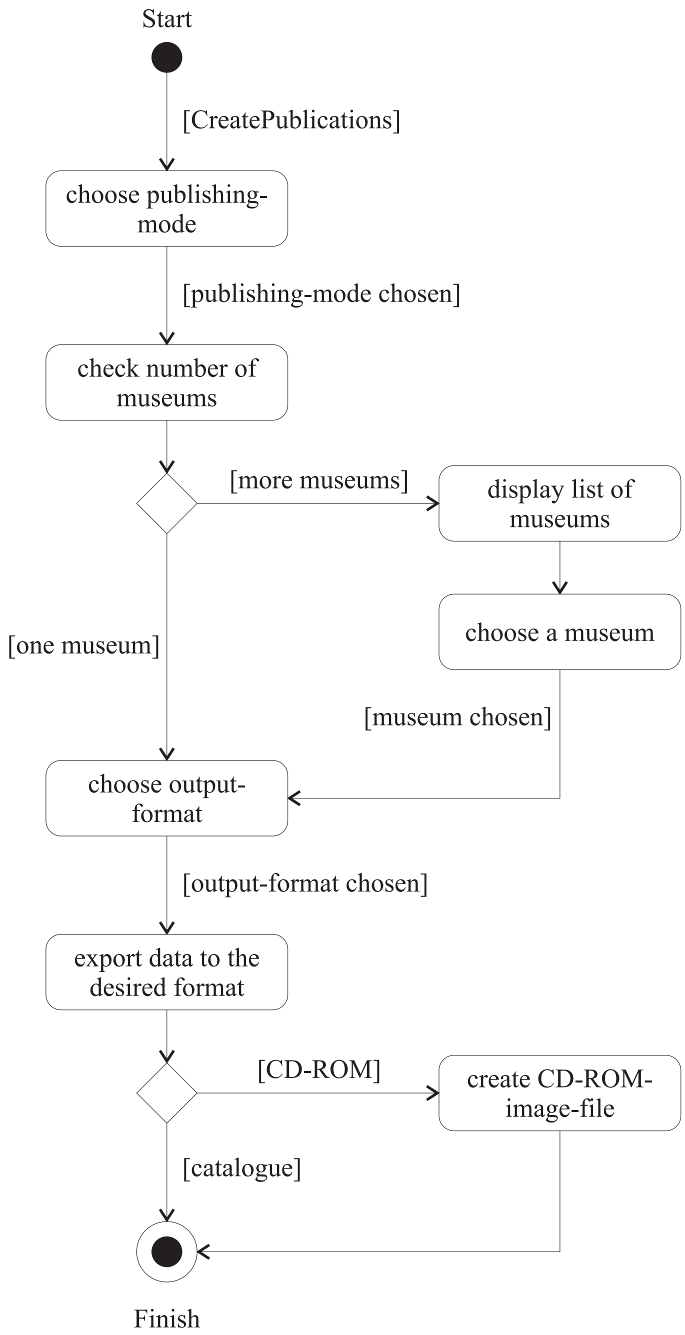


Figure 2.16: PublishMuseum

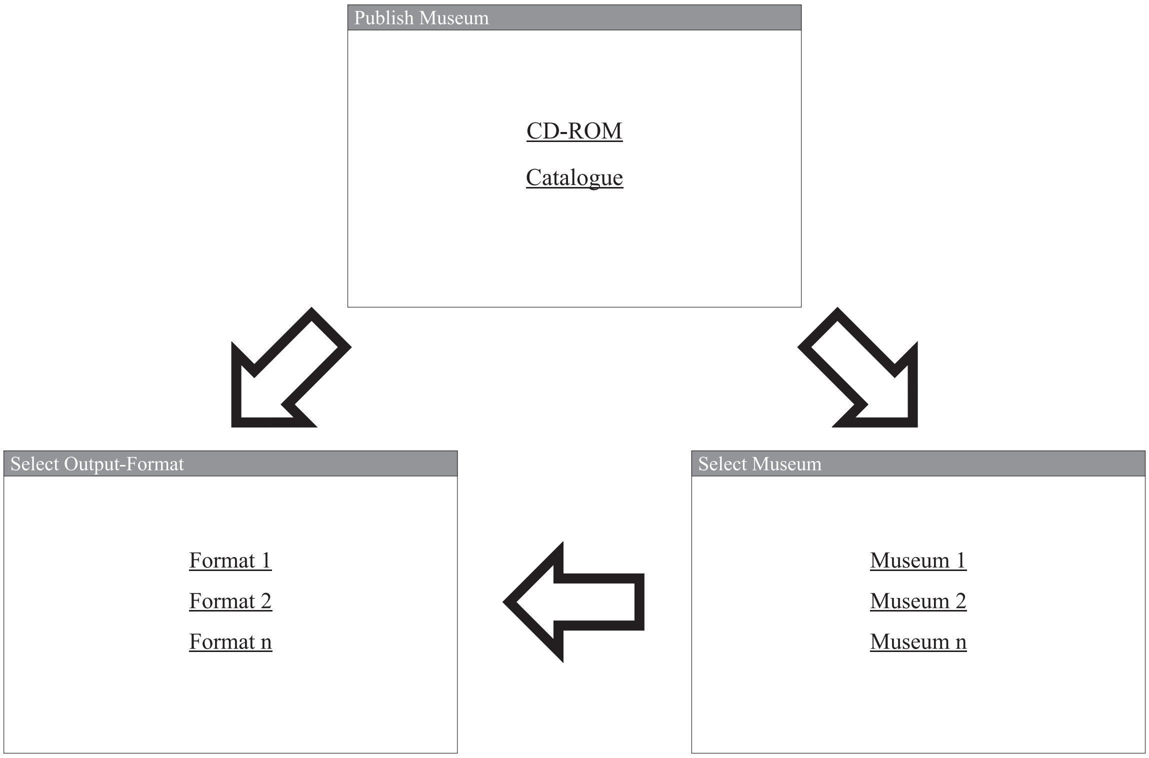


Figure 2.17: PublishMuseum

Other Requirements

None.

2.3.9 Use Case: Administrate

This use-case deals with the administration of the database-access rights. Furthermore it will provide the function to make backups or restore data. The system shall be accessible through well-defined interfaces with the idea, that people who don't know the operating system and even have little experience with computers shall be able to administrate the system.

Actors

- Administrator
- Database

Priority

Priority: 1

Status

Filled

Preconditions

- User logged in and successfully authenticated as administrator.

Postconditions

Possible changes of the access-rights matrix are saved.

Extension Points

Error Handling

"Used" Use Cases

none yet

Flow of Events

The Flow of events can be seen best on the storyboard for this use case

User Interface

See figure 2.18, page 56.

Sequence Diagrams

None.

Subordinate Use Cases

ManageRights, ManageGroups, CreateBackup, etc. (This list is by no means complete!)

View of Participating Classes

None.

Other Artifacts

None.

Other Requirements

- login as administrator has to be secure

None.

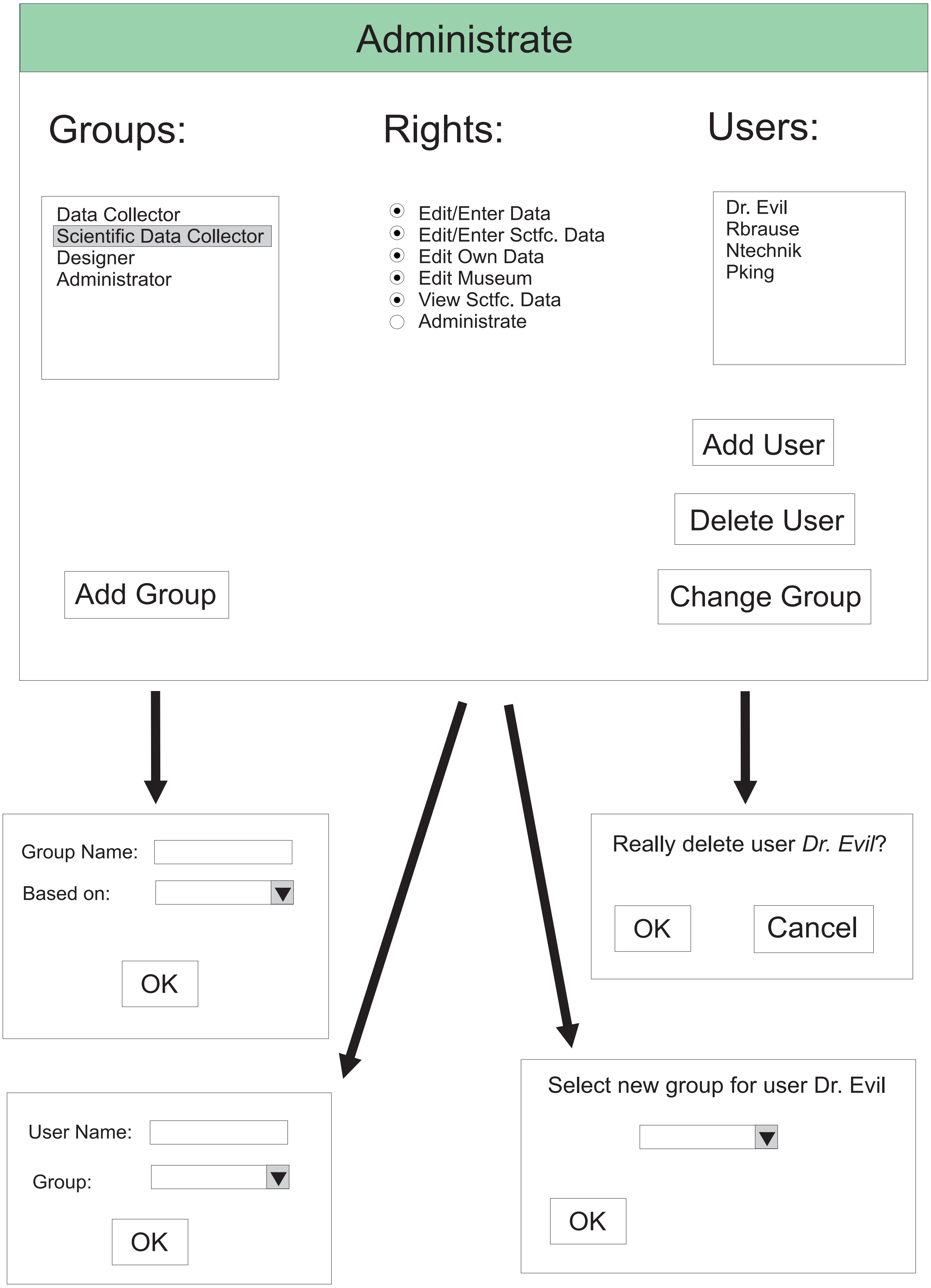


Figure 2.18: Activity diagram for use case "Administrate"

Chapter 3

Project Proposal

According to our additional information concerning "Use Cases and the Project Plan", we will now estimate the amount of work for our project.

This includes the following steps:

- Weighting Actors
- Weighting Use Cases
- Weighting Technical Factors
- Use Case Points
- Project Estimate

3.1 Weighting Actors

In this first step, we look at the actors, which we have located earlier and allocate them to a certain group, depending on the weight of the actor, which can be simple, average or complex. Then we multiply the number of actors in each group by a weighting factor(1, 2 or 3).

- InternetUser – complex
  Reason: The InternetUser utilizes the Virtual Museum (GUI).
- Designer – complex
  Reason: The Designer has to be able to modify the Virtual Museum (GUI).
- DataCollector – average
  Reason: The DataCollector extends the actor Designer but he does not have to be able to modify the Virtual Museum. He just operates with input-fields for text and file-uploads (ASCII-like).
- Scientific DataCollector – average
  Reason: Extending the DataCollector, same reason as above.

- Administrator – complex
  Reason: Extending the Scientific DataCollector, same reason as above. Although the addtional configuration- and administration tasks will also be covered by HTML-based ASCII-like interfaces, the complex structure of the tasks performed by this actor led to the decission to label it complex.

- Database – simple
  Reason: The Database is only another system with a defined API.

The assumptions made above result in:

1 simple * 1 = 1
2 average * 2 = 4
3 complex * 3 = 9

Ammounting up to a total of: 14.

3.2 Weighting Use Cases

This second step is similar to what we did with the actors. We again have 3 different groups in which the Use Cases will be devided. The decision is based on how many transactions have to be carried out in a Use Case.

- Authenticate – simple
  Reason: Login and authenticate procedures are of a standardized and rather simple nature.

- Enter data – complex
  Reason: As there are several different kinds of data to be entered and due to the web-based file upload that has to be implemented this seems to be a complex use case.

- Enter scientific data – simple
  Reason: Most transactions are covered by use case "Enter data". Only difference are the additional input-fields.

- View data – simple
  Reason: This use case is only responsible to display the selected data.

- View museum – average
  Reason: This is a more difficult task than just displaying objects. It is responsible for displaying the museum with its floors, rooms and showcases.

- Edit museum – complex
  Reason: As the designers will be able to edit the museum rather freely, it might prove to be a rather complex task to provide them with an easy-to-use but powerfull desing interface.

- Search – average
  Reason: Although there will be two differnt search functions to be implemented, this should not confront an experienced programmer with any problems.

- CreatePublications – average
  Reason: Needed to provide an easy way to export data for publication on CD-ROM or paper. The use of XML should simplify this task.

- Administrate – average
  Reason: The database-access-system will be a rather complex part of the system. Nevertheless, such procedures are well explored and thus not too difficult to implement.

The outcome of weighting the use-cases is:

3 simple * 5 = 15
4 average * 10 = 40
2 complex * 15 = 30

Ammounting up to a total of: 85.

Adding the actor's weights this gives a total of: 99 UUCP (unadjusted use case points).

3.3 Weighting Technical Factors

TCF = Technical complexity factor.

- Distributed System
  T1: 2 * 2 = 4
  Reason: There will be a seperate webserver and a database server which have to interact.

- Response or throughput performance objectives
  T2: 1 * 3 = 3
  Reason: Speed is likely limited by human input as well as it depends on network-speed.

- End-user efficiency
  T3: 1 * 5 = 5
  Reason: As there are many users at a time, efficiency is important.

- Complex internal processing
  T4: 1 * 1 = 1
  Reason: There are hardly any complex calculations to be made in our system.

- Code must be reusalbe
  T5: 1 * 4 = 4
  Reason: Nice and important.

- Easy to install
  T6: 0.5 * 0 = 0
  Reason: As our company is responsible for "installing" the system, the user has nothing to do with that.

- Easy to use
  T7: 0.5 * 5 = 2.5
  Reason: Our customer wishes a system which is intuitive and easy to use.

- Portable
  T8: 2 * 3 = 6
  Reason: It is important for our system to be portable. Nevertheless, thinking of our software's structure, it will not be very hard to implement that way (see XML).

- Easy to change
  T9: 1 * 2 = 2
  Reason: Good, if customer is not satisfied with the design.

- Concurrent
  T10: 1 * 5 = 5
  Reason: As our system will be multi-user oriented this is an important point.

- Includes Special Security Features
  T11: 1 * 5 = 5
  Reason: Security is important for authentication, group-management and encrypted data-upload (e.g. a plain web-browser SSL-encryption).

- Provides direct access for third parties
  T12: 1 * 5 = 5
  Reason: Our system is a customer-oriented software with direct access to the database.

- Special user training facilities are required
  T13: 1 * 2 = 2
  Reason: Just the administrator and maybe the designer will need training.

This results in a TFactor of 44.5. Therefore our TCF = 0.6 + (0.01 * TFactor) = 1.045.

3.4 Weighting Environmental Factors

- Familiar with Rational Unified Process
  F1: 1.5 * 4 = 6
  Reason: As we all have already worked a long time for SE AG we are very familiar with the Rational Unified Process.

- Application Experience
  F2: 0.5 * 4 = 2
  Reason: Most of the team are very experienced concerning databases and the other parts of our system.

- Object-oriented experience
  F3: 1 * 5 = 5
  Reason: Most of the team had excellent educational experiences.

- Lead analyst capability
  F4: 0.5 * 3 = 1,5
  Reason: The constant change of project-leadership may be a problem.

- Motivation
  F5: 1 * 5 = 5
  Reason: Our team is really eager.

- Stable Requirements
  F6: 2 * 4 = 8
  Reason: The customer could turn out to be a threat for the stability.

- Part-time workers
  F7: -1 * 2 = -2
  Reason: Mostly full time workers for this project.

- Difficult programming language
  F8: -1 * 2 = -2
  Reason: No final decission on the used programming lanuage has been reached yet.

This result in an EFactor of 23.5.
The environmental factor (EF) equals (1.4 + (-0.03 * 22)) giving a result of 0.695.

3.5 Conclusion

3.5.1 Use Case Points

Finally, we calculate the use case points (UCP).
UCP = UUCP * TCF * EF

In our case:
UCP = 99 * 1.045 * 0.695 = 71.9

3.5.2 Project Estimate

We use 20 man-hours per UCP giving us 1438 total man-hours which leads to about 36 weeks each 40 hours work.
As our team consists of 5 people, each will have about 8 weeks work plus 3 weeks for team issues. This gives about 11 weeks with 40 hours each. As we will invest about 20 hours a week for this project, our system should be ready within 22 weeks or 5 months.

3.5.3 Costs

Multiplying the total man-hours by 1.000,- ATS per hour, this gives a sum of about 1.4 Mio. ATS in total.

Chapter 4

Architecture and Project Plan

4.1 Architecture

In this section, we want to find an architecture of our software system in order to have a basic idea of what the system actually looks like and where the main dependencies are. With the help of the architecture to understand the system we should also be able to implement it more systematically as well as produce more reusable code.

One possible solution is the "4+1"-View [6]. The idea of P. Kruchten et al at Rational Software Corp. is to provide several views to look at the software's architecture. They say that mostly people try to show every bit and byte of the system within just one single diagram. According to them this is not possible. It results in confusion or even misunderstanding concerning the structure of the developed system. At least this leaves a diagram which covers everything but nothing correctly, completely and understandable. The four views are

- logical view,
- process view,
- physical view,
- development view.

The fifth one exists of the so-called scenarios where all the four we've just mentioned are put together. All in all this is a great way of constructing an architecture.

Nevertheless, in order to being able to profit from our detailed use cases and their descriptions we decided on a more use case-oriented way of designing our system's architecture. This one also seemed not as complex as the "4+1"-View and therefore easier to handle. Large quantities of information and a lot of examples have been produced on the topic of **patterns**. There is lots of documentation available - see [7] just to mention one source.

Nevertheless, we decided not to use a predefined pattern but to construct our architecture entirely ourselves. The use cases provided us a very good basis to categorize functions of our systems and then combine these forming "modules" or "subsystems". Next we found dependencies between the modules. This is the so-called "bottom up" procedure.

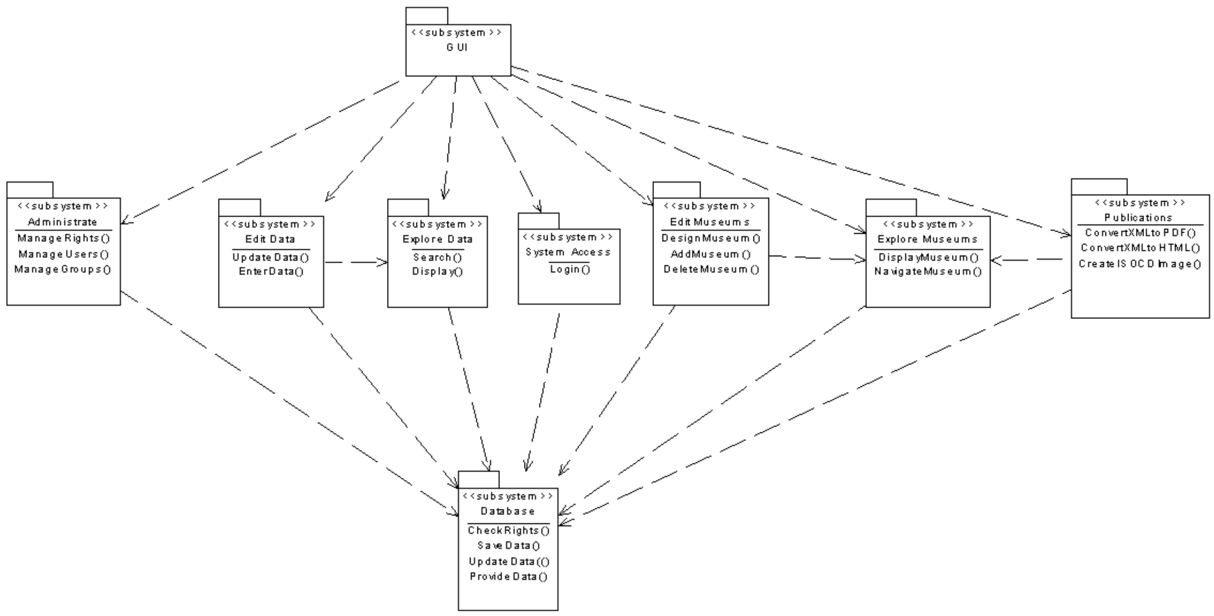


Figure 4.1: Architecture

The result of this process is diagram 4.1, page 63.
The arrows show a functional dependency relationship. This diagram does not contain any temporal dependecies such as "first we need to logon to being able to enter data".
We can see that the database is one of the most important subsystems as almost any other one needs something from it. The GUI of course is also a vital part as it uses functionality from the middle layer to provide it to the user.
Furthermore the main functionalities which are provided by each module are listed.

4.2 Project Plan

This section will deal will the project plan. Its aim is to give a plan "when to do what" in the upcoming Construction Phase.

A series of iterations for construction and definition of functionality will be planned. This functionality shall be delivered at the end of such an iteration.

In our Project Proposal (section 3, page 57) we estimated a total time of 22 weeks. Summed up we have already spent about 4 weeks working on the Inception Phase and the Elaboration Phase. This leaves 18 weeks for the rest of the project.

We will have 4 iterations with a length of 4 weeks each. This gives us a buffer of 2 weeks for miscellaneous delays of any kind.
Within each iteration we will basically deal with several use cases, see below for details.

At the end of each iteration we will contact our customer and present the project's current state. At the beginning of the next we will implement possible changes the customer

wishes.
Additionally to this, after about 2 or 3 weeks of work within every iteration we will consult our customer again and discuss the progress and present the current state of the project. This shall be done in order to prevent delays which could arise if we have to reconstruct and recode large parts of the project due to misunderstandings. This way, on the one hand we already have something to show the customer and talk with him about it, on the other hand it will not be that hard to change already existing work and plans for the rest of the iteration as the customer wishes.
Once we have agreed upon the way the project shall proceed, we will work on it and code it that way. Afterwards there will not be any major changes anymore.

The following gives an overview of the 4 iterations that will be covered within the Construction Phase.
We agreed on exactly this order of implementation, because it provides first of all the basics of our system which is needed by almost every use case - the database. Next we will start with topics which again other parts rely on. Furthermore it gives us the possibility to rather quickly have a great part of the project dealt with, as well as it enables our customer to already start entering data and get used to the software as soon as about 2 months after starting the coding-process. Combined with current communications between our customer and us, this can lead to the possibility of already correcting problems of any kind while the software still isn't officially delivered.

4.2.1 Iteration 1

1. **Basic structure of the Database – 1.5 weeks**
   The database is one of most important parts of the project as nearly all use cases need the database in some way. Therefore we will construct its basics initially, of course having the relevant demands in mind.

2. **Use Case "Enter Data" – 1.5 weeks**
   At the end of this iteration this use case will be useable. Its structure will and implementation will be the final one, although several small changes might be possible at a later time.

3. **Use Case "View Data" – 1 week**

4.2.2 Iteration 2

1. **Use Case "Enter Scientific Data" – 0.5 weeks**
   This use case already uses "Enter Data" and therefore only needs little additional work.

2. **Use Case "Search" – 1 week**

3. **Use Case "Authenticate" – 0.5 weeks**

4. **Database and basics for the Museum – 2 weeks**
   Programming-basics such as objects or parts of the code concerning the museum will be prepared. The database will also be enhanced and prepared for the museum.

4.2.3 Iteration 3

1. **Use Case "View Museum" – 1.5 weeks**

2. **Use Case "Edit Museum" – 2.5 weeks**
   "Edit Museum" seems to be and is very complex. Nevertheless, it uses large parts of "View Museum" and its work can therefore be estimated with this amount of time.

4.2.4 Iteration 4

1. **Use Case "Administrate" – 2.5 weeks**

2. **Use Case "Create Publications" – 1.5 weeks**

Chapter 5

Prototypes

To give the customer a little foretaste of ADS, and also to ensure that critical points of the system will run smoothly, we will design some prototypes.

5.1 The first prototype

5.1.1 General

To prove the interaction between the web-interface and the database (especially the file-upload) and the user-management we designed a first prototype, which can be viewed at

`http://dev3.subnet.at/adsproject/`

The prototype starts with a Login-Screen where one can authenticate himself. For users who just want to view the items in the database this isn't necessary. They can override this action to be shown a list a artefacts. By picking one of these, all the information available for this artefact (facts and maybe even an image of it) will be shown.
If one can authenticate himself as Data Collector he can also enter new artefacts and upload images for these artefacts. He can also delete artefacts entered by himself.
If one can authenticate as Administrator he can enter artefacts and delete any artefacts he wants.

5.1.2 Facts

According to the description in 5.1.1 the prototype contains fragments of the following Use-Cases:

- Authenticate
- Enter Data
- View Data
- Administrate

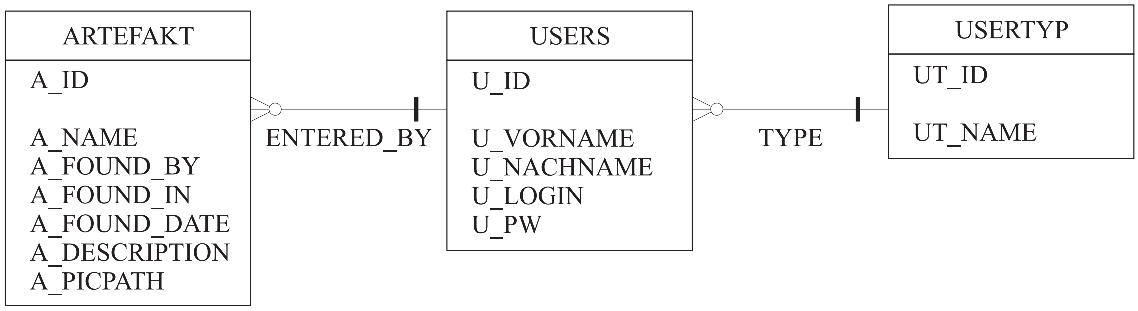


Figure 5.1: Data-model for the first prototype

We implemented a simple database (see Figure 5.1) using MS SQL Server 7.0. It consists of three tables:

USERTYP The user-groups mentioned in section 1.1.4 on page 7; There are only Scientific Data Collector, Administrator and Visitor at this time (but Visitors don't appear in the database, because they don't need to authenticate).

USERS All persons who work with ADS; Every of these persons must belong to a user-group.

ARTEFAKT Archeological Artefacts; Can be entered and manipulated by (Scientific) Data Collectors via the web-interface.

The connection between the web-interface and the database is done by ASP running on MS IIS.

Someone entering data can also upload images in JPG- or GIF-format for certain artefacts, which will also be displayed when viewing the corresponding artefact. The images aren't stored in the database itself, but in a file-system. There's only an information in which directory the image is stored in the attribute A_PICPATH of the table ARTEFAKT.

Chapter 6

Iteration Plans

6.1 Iteration 1

In the iterationplan for Iteration 1, which deals with the use-cases EnterData and ViewData, we present the preliminary model of the database. This is essential to the ADS-system, and is thus included along with the class-, interaction- and state-diagrams.
Iteration 1 deals with the use-cases EnterData and ViewData. The relevant classes for these are:

- Class RightManager
- Class DBManager
- Class DataClass *(abstract)*
- Class ViewData *(derived from DataClass)*
- Class EnterData *(derived from DataClass)*
- Class Layout

6.1.1 Database Diagram

see figure 6.1 on page 71

6.1.2 Class Diagram

see figure 6.2 on page 72

Class RightManager

This class deals with the management of rights. In the first iteration it is not too sophisticated, however, it will gain importance in later iterations.

Class DBManager

This class connects to the database and is able to read data from the database and write data to the database. It exposes an XML Interface for the reception of the date. Furthermore, it is able to check the data's consistency.

Class DataClass

This is the parent class for data-aware classes, like ViewData and EditData. It is defined abstract.

Class ViewData

ViewData is responsible for showing data. It holds a protected member DataSet which it uses to store the data. It instantiates Layout in order to layout the data. It uses DBManager to read data.

Class EnterData

ViewData is responsible for receiving data. It holds a protected member DataSet which it uses to store the data. It instantiates Layout in order to layout the data and uses DBManager to read and write data.

Class Layout

This class deals with the layout of data which it receives via an XML interface. It is connected to a StyleSheet, which it stores in the attribute CSS.

6.1.3 Sequence Diagrams

Viewing Data

see figure 6.3 on page 73
Note: The actor in this diagram need not be an InternetUser - as InternetUser is the base class for all other types of users, it may be used polimorphly. The instantiation of RighManager will see to the correct association with the right group.

Entering Data

see figure 6.4 on page 74

6.1.4 Collaboration Diagrams

Viewing Data

see figure 6.5 on page 75

Entering Data

see figure 6.6 on page 76

6.1.5 StateDiagram

Class EnterData

The StateDiagram for EnterData is realised in two ways. First there is an intrinsic view of the states of the class - see figure 6.7 on page 77.
Then there is also an approach that covers a more user-oriented, macro-state model, which is of advantage when discussing details with the customer - see figure 6.8 on page 78.

Figure 6.1: Conceptual Data-model

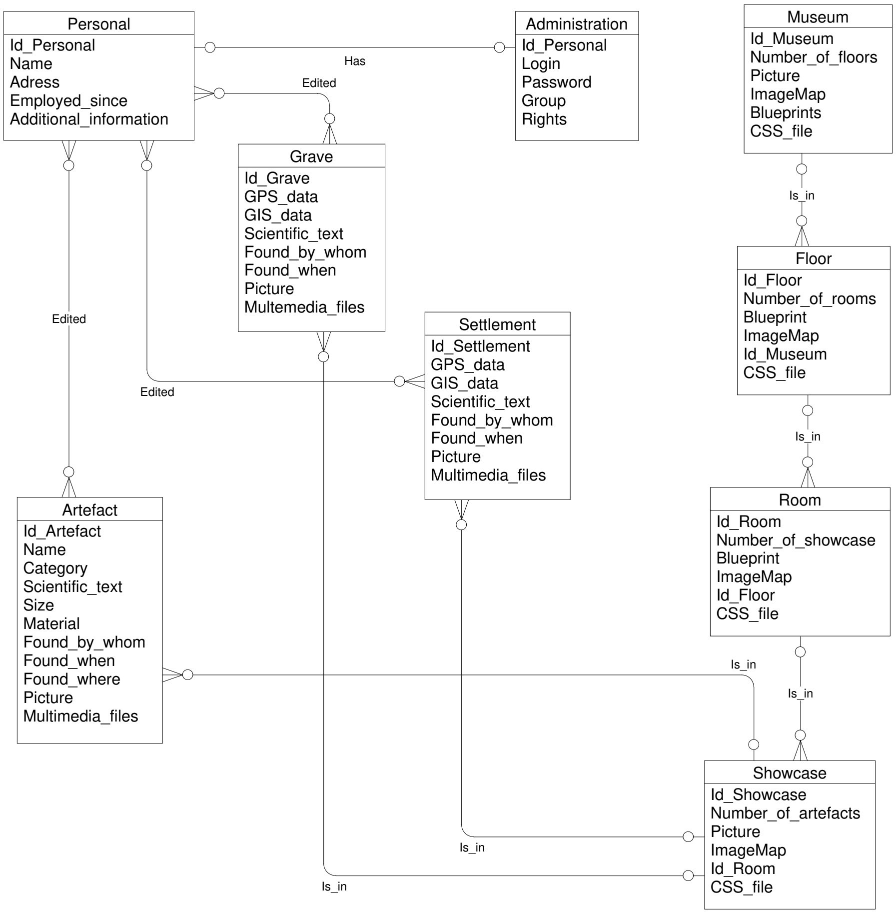


Figure 6.2: Class Diagram

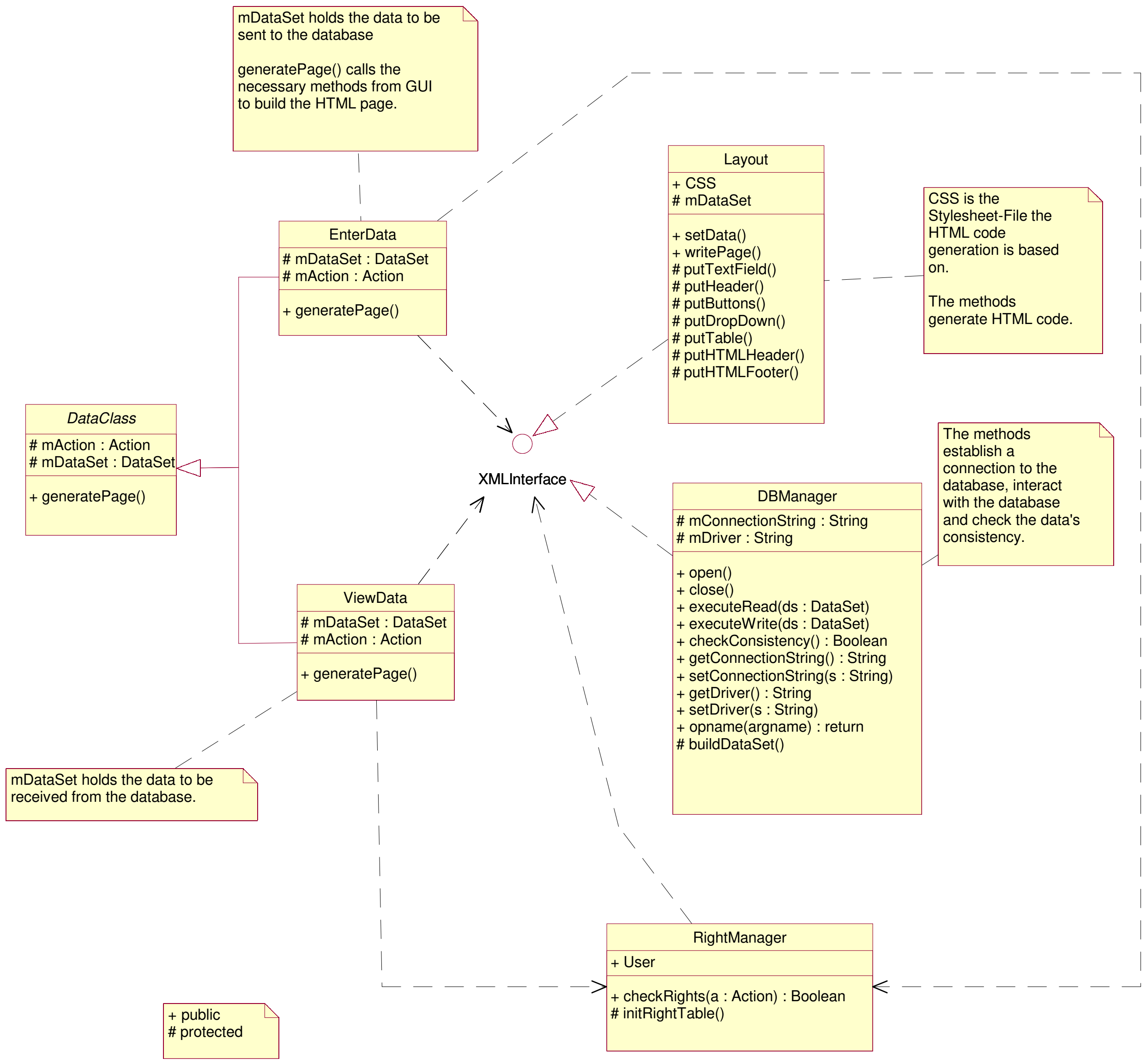


Figure 6.3: Sequence Diagram - Viewing Data

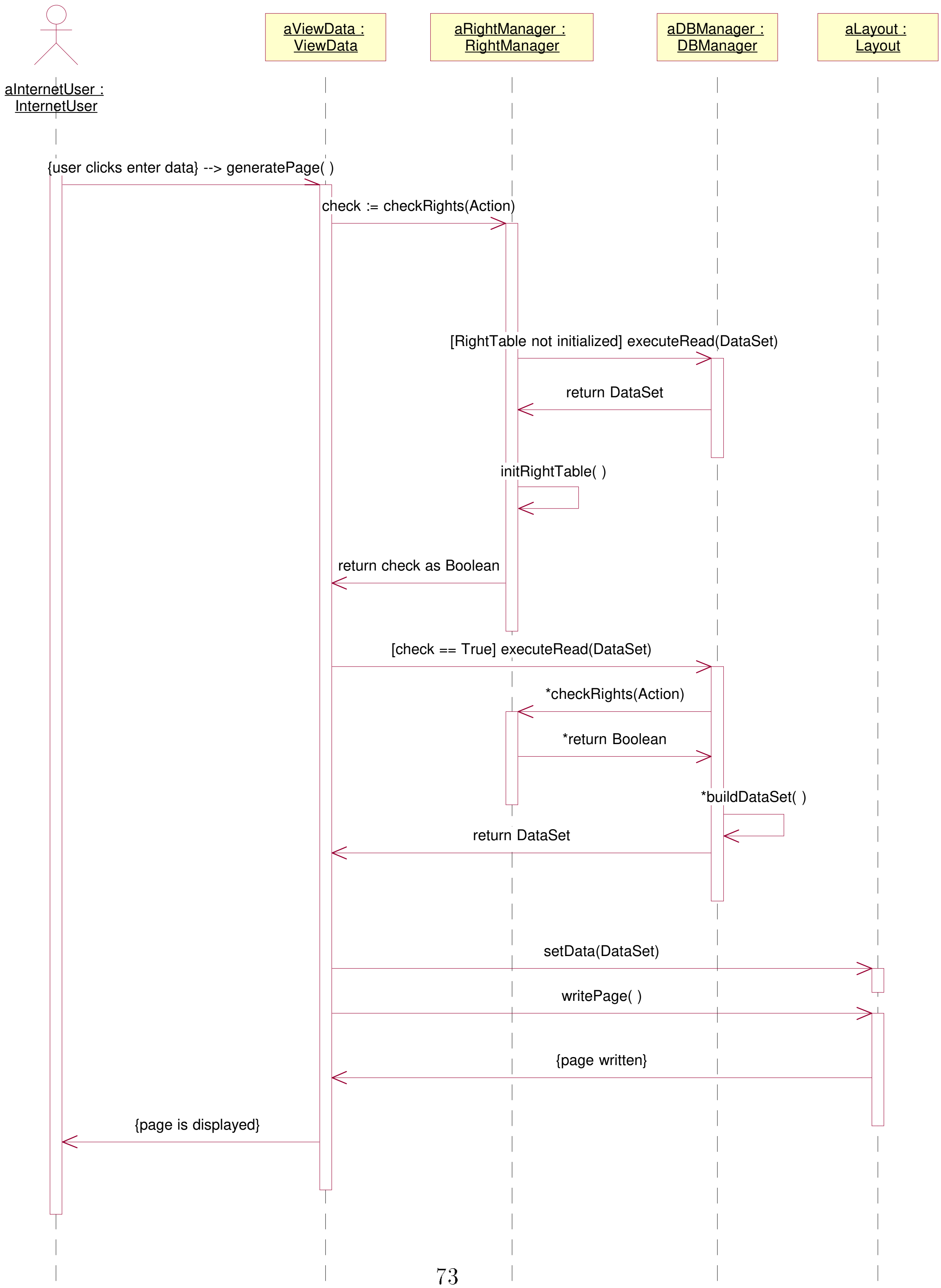


Figure 6.4: Sequence Diagram - Editing Data

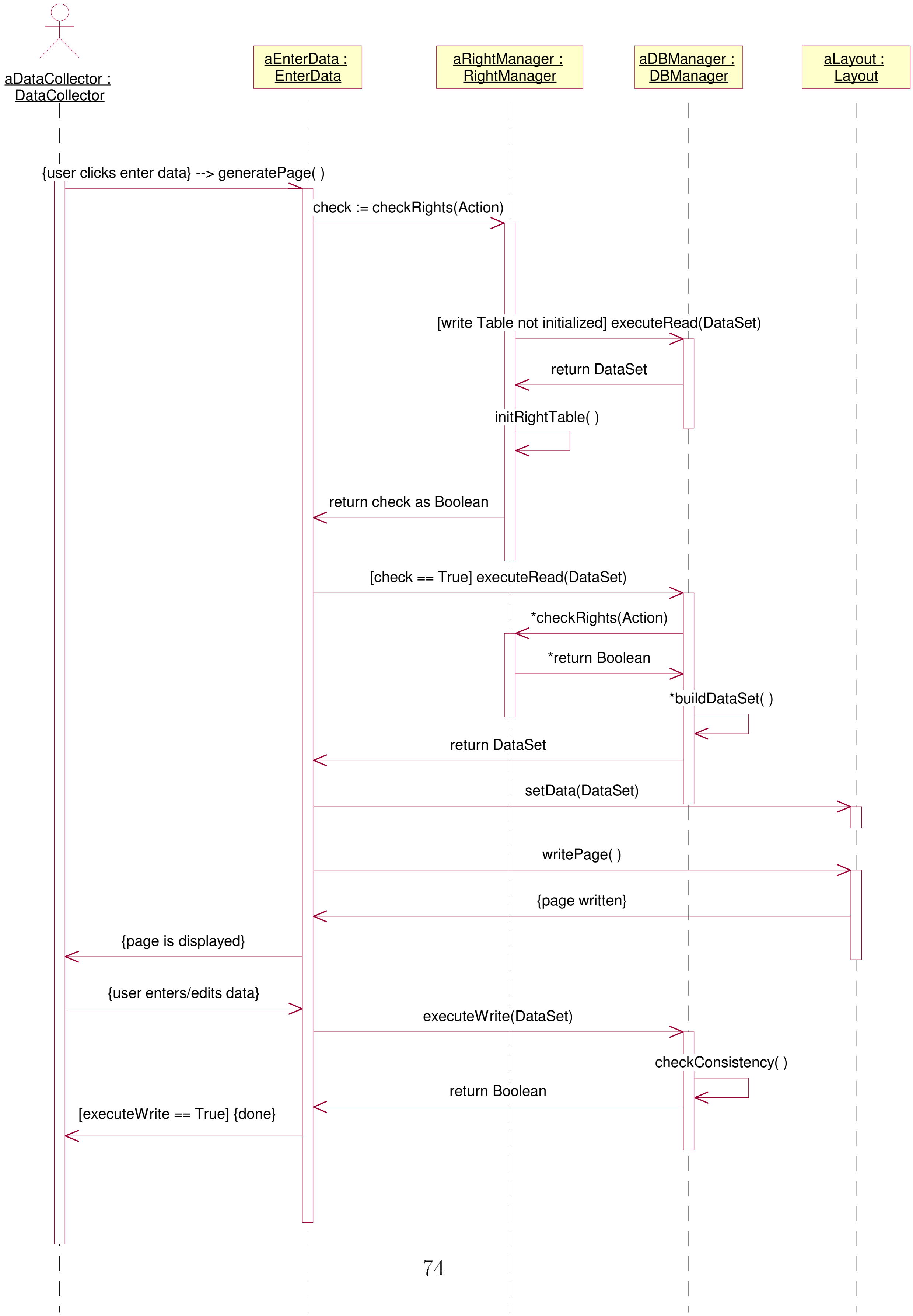


Figure 6.5: Collaboration Diagram - Viewing Data

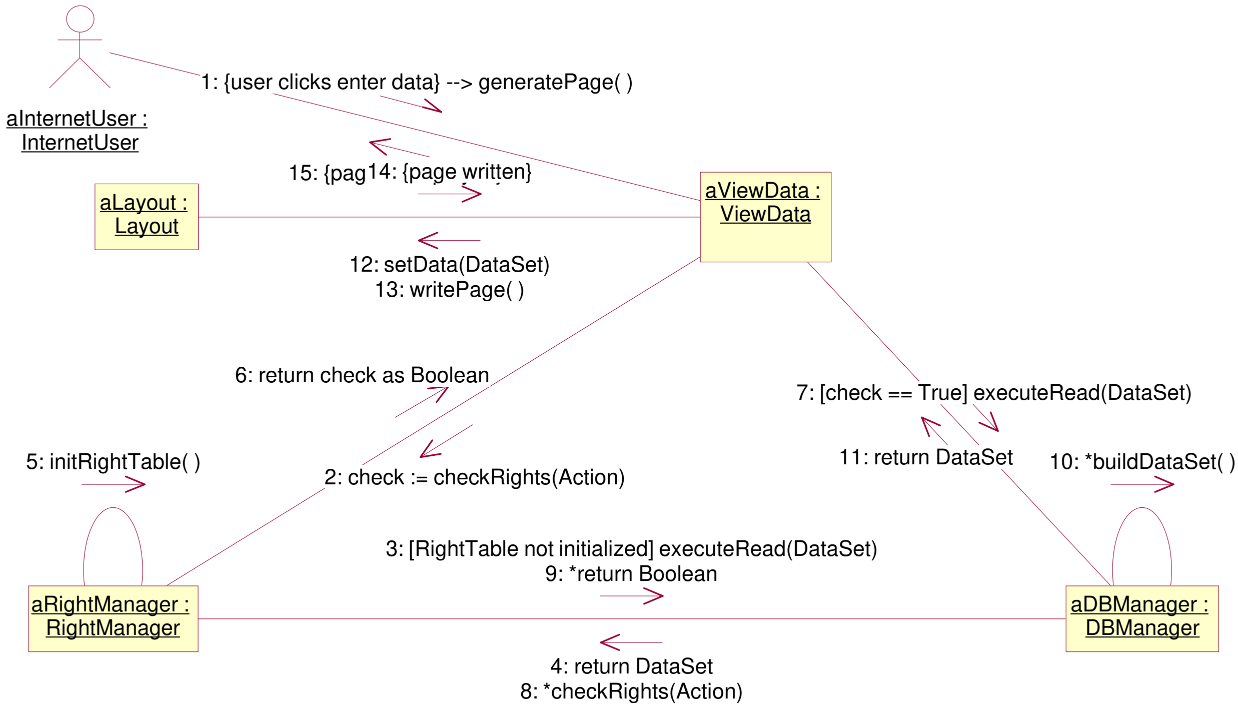


Figure 6.6: Collaboration Diagram - Editing Data

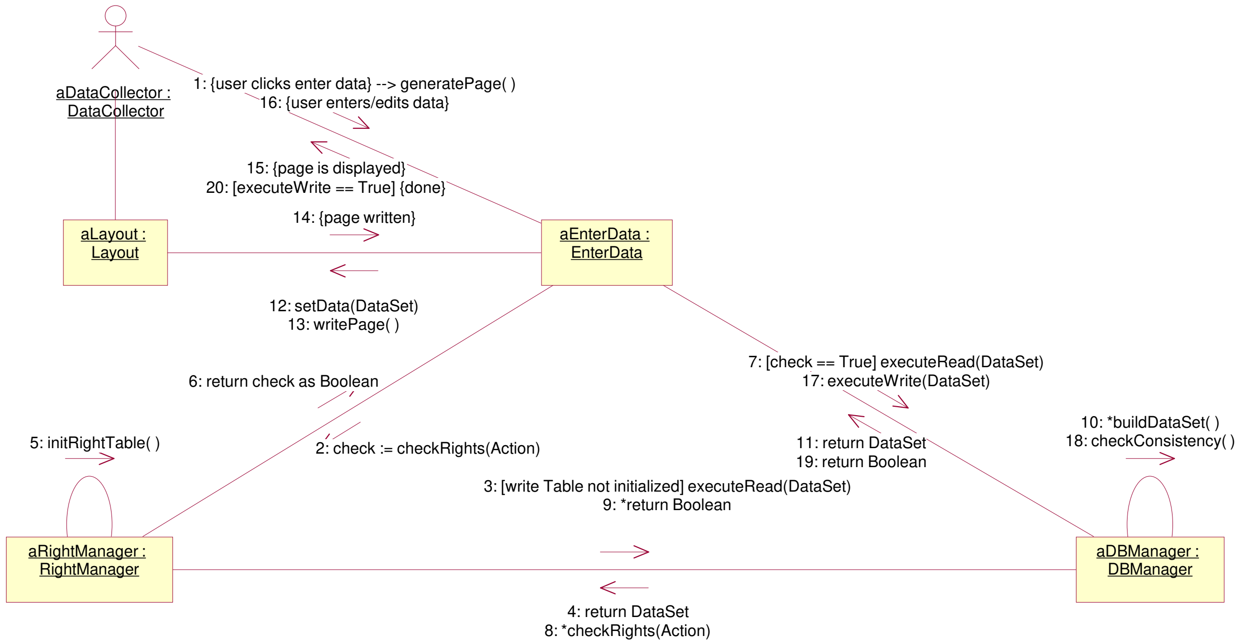


Figure 6.7: State Diagram - Class Enter Data

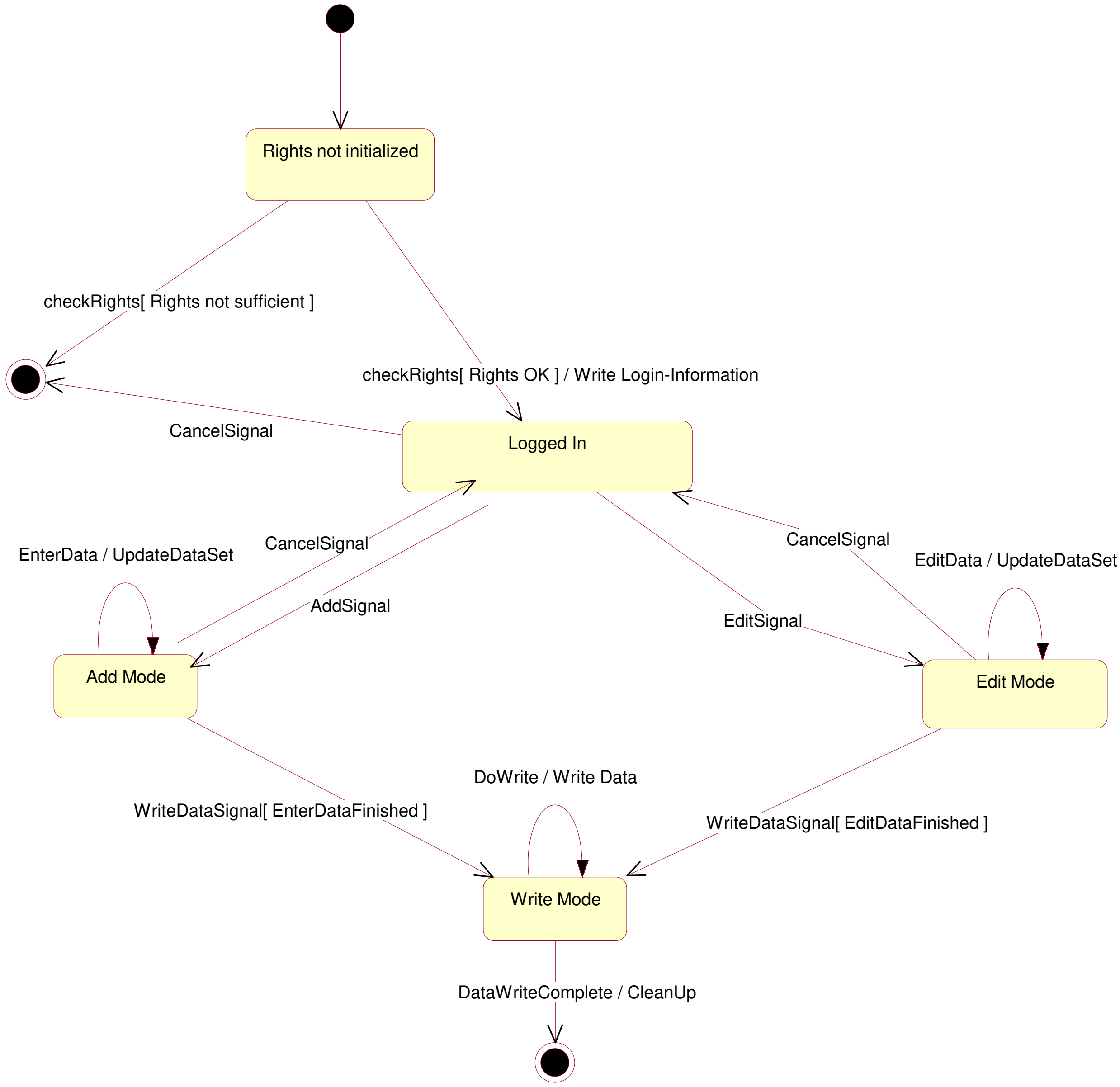
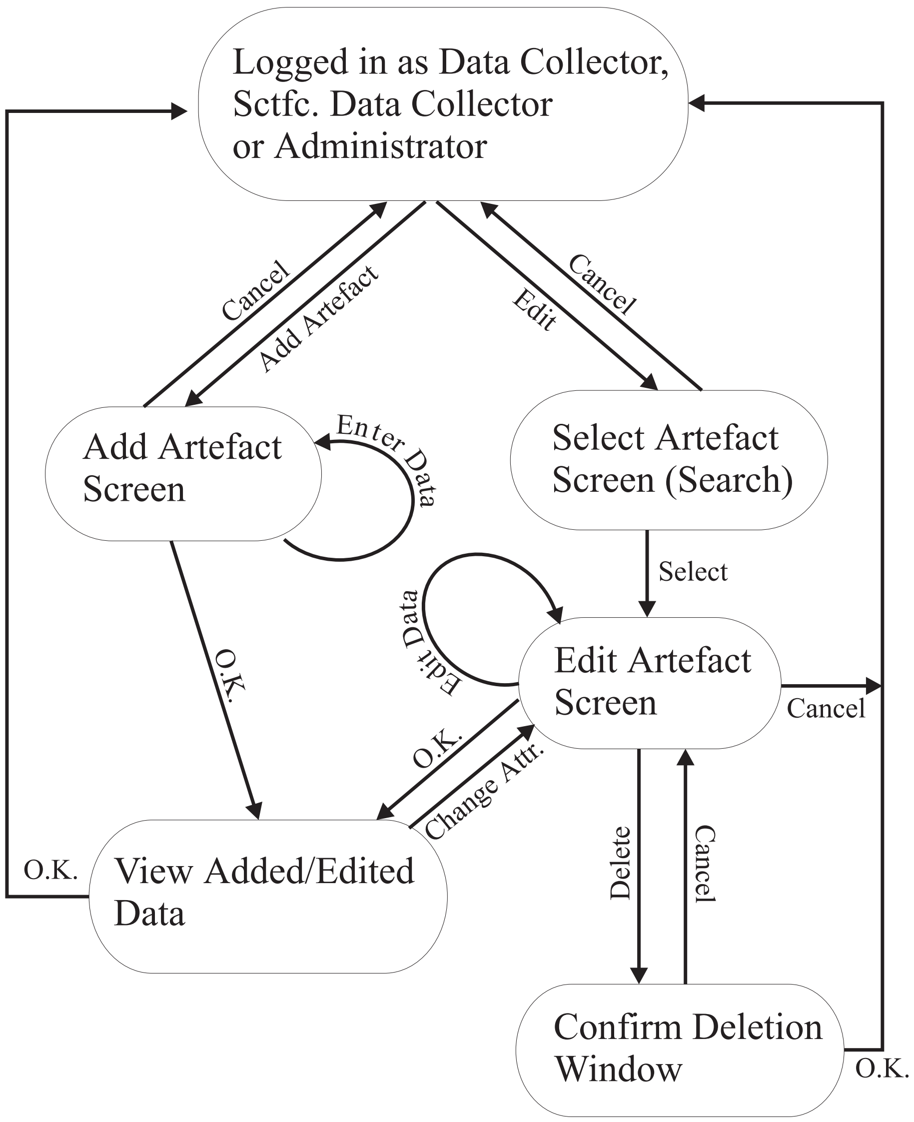


Figure 6.8: Macro State Diagram - Class Enter Data



Appendix A

Protocols

A.1 11th October 2000

In this first meeting our academic supervisor Mag. Dipl.-Ing. Schwaiger Roland introduced us to the project. Several groups will work on the project and each develop their own solutions. At the end of the semester, there will be presentations in order to let the others know, how each group solved their problems.

Person responsible for protocoll: Markus Amersdorfer.

Participating persons: The whole PS-group.

Topics talked about: Introduction to the project. General Information.

New assignments: Start with the product analysis and prepare questions for the upcoming meeting with our customer's representative.

Open questions: None yet.

A.2 18th October 2000

The purpose of this meeting with Mag. DI R. Schwaiger - who is our customer's representative - was first of all to answer questions which came up while working on this project. I will inform the other members of my group about the answers at the next meeting. The next steps of the whole project will be explained in detail at
http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html.
Mr. Schwaiger and I also discussed some organizational topics.

Person responsible for protocoll: Markus Amersdorfer.

Participating persons: Mag. Dipl.-Ing. Roland Schwaiger and Markus Amersdorfer

Topics talked about: Questions to the customer which came up when working on the project since the first meeting. Some organizational topics.

New assignments: See http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html, Inception Phase, A2.

A.3 31st October 2000

Since we hadn't prepared any questions, we discussed some topics that came up spontanously. I will inform the other members in an upcoming briefing that the meetings with Mr. Schwaiger should be used to discuss problems, questions and ideas about the project. The next steps of the whole project will be explained in detail at http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html.

Person responsible for protocoll: Christoph Obermair.

Participating persons: Mag. Dipl.-Ing. Roland Schwaiger and Christoph Obermair.

Topics talked about: Database, XML, organizational topics.

New assignments: See http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html, A3.

A.4 8th November 2000

In the first part of this meeting the use-case model was presented. Some changes were suggested, which will be updated in the upcoming version of this document. Furthermore some design issues where covered, especially regarding the database design and the visualisation of the virtual museum.

Person responsible for protocoll: Dominik Engel.

Participating persons: Mag. Dipl.-Ing. Roland Schwaiger and Dominik Engel

Topics talked about: Use-Case Model, Design Topics, Database, structural issues

New assignments: http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html, A4.

A.5 16th November 2000

We presented the outcome of our Project Proposal and clearified open questions that occured during this process. Some minor inconsistencies became obvious. They will be dealt with in the newest version of this document

Person responsible for protocoll: Martin Angerer.

Participating persons: Mag. Dipl.-Ing. Roland Schwaiger and Martin Angerer

Topics talked about: Outcome of Project Proposal, Improvement Proposals

New assignments: http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html, A5.

A.6 7th December 2000

Discussion started with a general view on software design. Meanwhile we have reached a point of development where the question of "How should the system be implemented" takes the place of the question "What should the system perform". So the very interesting point was how to project use-cases to software implementation. After this project presentation followed.

Person responsible for protocoll: Sandor Herramhof.

Participating persons: Mag. Dipl.-Ing. Roland Schwaiger and Sandor Herramhof

Topics talked about: Other methods to encounter structured software design (defining classes, CRC-cards). Project presentation.

New assignments: http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html, A6.

A.7 20th December 2000

We discussed the archtitecture and the project plan. Concerning the first, we managed to find some errors which are to be corrected until the next meeting.

Person responsible for protocoll: Markus Amersdorfer

Participating persons: Mag. Dipl.-Ing. Roland Schwaiger and Markus Amersdorfer

Topics talked about: Architecture, project plan, upcoming tasks.

New assignments: http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html, A7.

A.8 11th January 2001

The prototype from assignment A7 was presented and discussed.

Person responsible for protocoll: Christoph Obermair

Participating persons: Mag. Dipl.-Ing. Roland Schwaiger and Christoph Obermair

Topics talked about: Everything about the prototype.

New assignments: http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html, A8.

A.9 25th January 2001

Presentation of the iteration plans. Some details have to be revised, the overall design was accepted. The revised diagrams have been updated in this document.

Person responsible for protocoll: Dominik Engel

Participating persons: Mag. Dipl.-Ing. Roland Schwaiger and Dominik Engel

Topics talked about: Iteration Plans, Class-, Sequence-, Collaboration and Statediagrams

New assignments: Prepare a WEB-presentation of the project and provide the relevant documents (http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/abschluss.html).

http://www.cosy.sbg.ac.at/~rschwaig/teaching/se/se2000/aufgaben.html, A9.

Appendix B

Points to clarify (November 8, 2000)

This section contains questions and answers for the meeting for phase on November 8, 2000. It is included in the appendix for reference and information.

B.1 Use-Case

- Can a user be member of more than one group (i.e. DataCollector + Designer)
  Yes, it depends what the member wants to do

- Two classes of visitors: Searching and Viewing?
  Not really necessary, as it depends what use case the actor will choose

- Should the Database be an actor?
  Yes

- Should potential maintanance software be an actor?
  No, as this is covered by the administrator (the software will thus act as administrator)

- Does the customer want maintanance software at all?
  Customer doesn't know too much about these topics. Should be decided internally by SE-AG

- Extension Points?
  A use-case extending another use-case, i.e. a use case that **might** *occur, but doesn't have to occur (e.g. Error-Handling)*

- Subordinate Use Case?
  A use-case used by another use-case, i.e. a use case that **always** *occurs, when the other-use case is addressed*

- Suggestion: Store files in the filesystem $\longrightarrow$ it's easier than storing BLOBS in the database
  Eventually this point will have to be decided internally.

B.2 Webfrontend

- File upload: for which types of files?
  image: jpeg + gif
  video/audio: real

- View MM-Files: won't be possible in a standard webbrowser $\longrightarrow$ only provide download?
  Yes

- When a new file is uploaded, should the old file be deleted?
  A message should be displayed letting the user choose

B.3 Database

- Are entries associated with the actor who entered them?
  Yes

- If so, will entries be deleted if the actor is deleted?
  It's generally not a good idea to delete from an entity that is as essential as actor, rather a flag should be used. The files of the deleted person can be stored in an archive (which also could be external - CD-Rom)

- Will deleting be possible at all? $\longrightarrow$ Suggestion: It's better to set a flag rather than deleting records, as this may lead to inconsitencies in the database
  see above

B.4 Management of Rights

- Only administrator or hierarchical? $\longrightarrow$ Suggestion: possibly hierarchical
  see below (3)

- Suggestion: Define rights for different groups via cross-table between available rights and existing groups
  yes

- Suggestion: To forestall deadlocks, administrator is always granted maximum possible rights
  yes

  To remain flexible, the management of rights should be done using the database. The management of rights seemed like a small topic to begin with, but is now expanding. We should keep an eye on it's importance.

B.5 Museum

- Is there one museum or are there several?

- While a muesum is being edited, what happens if somebody wants to see it $\longrightarrow$ Suggestion: lock it
  Do not lock the whole museum, only the area (room) that is being edited.

- Suggestion: The visualization of the museum will basically be done using blueprints of floorplans. This will be shown as an image map, with links to rooms. There will be an area calles "stairs" where it will be possible to move between adjacent floors. There will be info-columns, which will display the theme of the floor.
  Rooms basically are imagemaps as well. The default will be a blueprint of the floor. Vitrinas can be viewed from above and clicked at. There is an area called "exit" to get back to the floor. Furthermore there are info-columns again which will diplay the theme of the room.
  Vitrinaes basically are imagemaps as well. The default will be a front-side-view. The artefacts will be displayed with an icon according to their category. This icon can be clicked for a detailled display. There will be an area called "plate" which will show the theme of the vitrina when clicked at.

  There will be the possibilities for users (Designers) to upload there own images and define maps upon them. This can be used for the design of floors, rooms and vitrinas. Furthermore icons for different categories can be uploaded. Thus the design will be very flexible. A room, for example, does not necessarily have to be a blueprint, but might be a photograph of an actual room containing vitrinas. By defining areas on the image the designer can associate part of the picture with a vitrina in the system.

  The basic idea was accepted as regards the VM ...

B.6 Publishing of CD-Rom and Catalogue

- Which formats does the system have to capable of exporting XML to?
  As stated before: XML is an intermediate, tranparent layer. On the media, XML won't be used, but will be exported to HTML and PDF format.

- How does our client acquire the exported data? Is it a good idea to simply download the data?
  Yes, this should be possible.

Bibliography

[1] R. T. Yeh and P. Zave, *Specifying Software Requirements.* Proceedings of the IEEE, vol. 68 ed., 1980.

[2] E. Yourdon, *Modern Structure Analysis.* Yourdon Press, 1st ed., 1989.

[3] S. Vollmann, *Aufwandschaetzung im Software Engineering.* IWT, 1st ed., 1990.

[4] M. Fowler, "Use cases and abuse cases," *www.softwarefuture.com*, June 1998.

[5] E. Kenworthy, "Use case modelling - capturing user requirements," *www.zoo.co.uk*, December 1997.

[6] P. Kruchten, *Architectural Blueprints - The "4+1" View Model of Software Architecture.* IEEE Software, 12 (6) ed., 1995.

[7] *The Patterns Home Page.* http://hillside.net/patterns/.