PS Software Engineering I
WS 2001/02
January 28, 2002

InfoWarrior

Version 0.6

Task 6

Group Document

Team 4

Zenina Huskic (zehuskic@cosy.sbg.ac.at)
Theresa Krichhammer (Aphid@gmx.at)
Peter Gruber (pgruber@cosy.sbg.ac.at)
Albert Meixner (albert@subnet.at)
Michael Brachtl (mbrachtl@cosy.sbg.ac.at)

Academic Supervisor: Dr. Schwaiger Roland

Correspondence to:

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

Abstract

This document contains analysis and design of a computer based collecting and presenting information system with the working title "InfoWarrior".

Contents

1 Introduction **2**
- 1.1 Planning . . . 2
- 1.2 Basic Organization . . . 2
- 1.3 Goal . . . 2
- 1.4 Responsibilities . . . 3
- 1.5 Problem Statement . . . 3
- 1.6 First considerations(CRC Session) . . . 4
  - 1.6.1 Identifying classes . . . 4
  - 1.6.2 Some scenarios . . . 4
- 1.7 Open questions . . . 5

2 CRC Session **6**
- 2.1 Problem Statement (Version two) . . . 6
- 2.2 Open Questions . . . 7
- 2.3 CRC Card Session . . . 7
  - 2.3.1 General . . . 7
- 2.4 InfoWarrior CRC Session . . . 8
- 2.5 Essential Use Case Modeling . . . 9
  - 2.5.1 General . . . 9
  - 2.5.2 Use Cases in InfoWarrior . . . 10
- 2.6 Essential User Interface Prototyping (First considerations) . . . 10

3 Use Cases Description **12**
- 3.1 Identifying Actors . . . 12
- 3.2 Identifying Use Cases . . . 12
  - 3.2.1 Description . . . 13
- 3.3 Essential User Interface Description . . . 15
  - 3.3.1 Customer User Interface . . . 15
  - 3.3.2 Staff User Interface . . . 16
- 3.4 Diagrams/Drafts . . . 17
- 3.5 Open Questions (3) . . . 19

4 Classes **20**
- 4.1 Classes List . . . 20
  - 4.1.1 Classes Description . . . 21

5 Interface Prototype **34**
5.1 HTML Prototype . . . 34
5.1.1 Introduction . . . 34
5.1.2 Design Principles . . . 34
5.2 WML Prototype . . . 35
5.2.1 Introduction . . . 35
5.2.2 Design Principles . . . 35

6 Sequence and Activity Diagrams **40**
6.1 Sequence Diagrams in InfoWarrior . . . 40
6.1.1 User Specific Sequence Diagrams . . . 40
6.1.2 Customer Specific Sequence Diagrams . . . 44
6.1.3 Reporter Specific Sequence Diagrams . . . 46
6.1.4 Editor Specific Sequence Diagrams . . . 48
6.1.5 Administrator Specific Sequence Diagrams . . . 48
6.2 Activity Diagrams in InfoWarrior . . . 52
6.2.1 User Specific Activity Diagrams . . . 52
6.2.2 Customer Specific Activity Diagrams . . . 54
6.2.3 Reporter Specific Activity Diagrams . . . 56
6.2.4 Editor Specific Activity Diagrams . . . 57
6.2.5 Administrator Specific Activity Diagrams . . . 57

7 Implementation **60**
7.1 Introduction . . . 60
7.2 Implementation Technology . . . 60
7.2.1 Libraries/Existing Systems . . . 60
7.3 Organisation . . . 61
7.3.1 Packages . . . 61
7.3.2 Servers . . . 61
7.3.3 Classes . . . 61
7.3.4 Data Consistency . . . 62
7.3.5 Case Study SQL . . . 63
7.4 Design Patterns . . . 64
7.5 Evaluation . . . 64
7.5.1 Advantages . . . 64
7.5.2 Disadvantages . . . 65

A Protocols **67**
A.1 25.10.2001 . . . 67
A.2 31.10.2001 . . . 67
A.2.1 Answers to the open questions . . . 68
A.3 07.11.2001 . . . 69
A.3.1 Notices . . . 69
A.4 14.11.2001 . . . 70
A.4.1 Notices . . . 71
A.5 21.11.2001 . . . 71
A.5.1 Notices . . . 71

A.6 05.12.2001 72
A.6.1 Notices 72
A.7 12.12.2001 73
A.7.1 Notices 73

Chapter 1

Introduction

Information is a very important part of our life. Human beings always wanted to and want to be good informed on what is happening in their neighbourhood as well as in their cities, countries and continets. Fresh information has an influence on our decisions, and that is why people like to be fast and good informed. The most common way the people get informed today is by watching TV, reading newspapers, listening to the radio, etc. However all this information is to be collected and presented to the end-users. Computer-based system are very suitable for collecting and presenting of information.

1.1 Planning

1.2 Basic Organization

We are a group of 5 members. Each member has a certain task to perform. The tasks are determined through the role distribution. Each week there will be a session between the group members and a session between Mr Schweiger and the project leader. The project leader rotates from session to the session.The following dates for sessions are already set:

- Huskic Zenina on October 31, 2001
- ...

Each member of the group should prepare her/his own solution to the problem. These solutions will be worked through by the group. Group document contains parts of the single solutions. Project leader should present the group document to Mr Schweiger. The project leader makes a protocoll about this session. The protocolls are to be found as appendix to this paper.

1.3 Goal

Goal is to perform analysis of the system as good as possible in order to prepare a basis for future implementation. Further on, the members of the group should get a better understanding of the iteration process. The iteration process should result in an end document (project paper). This document should contain clear project definitions and requirements as well as implementation constraints.

1.4 Responsibilities

The project leader is Zenina Huskic. She is responsible for the overall organisation during the session. Michael Brachtl is responisble to watch the prepared time line. All other group members do not have any special responsibilities.

1.5 Problem Statement

The following text contains the theme and open questions to the theme. The questions are numbered and given as a footnote.
Eine Nachrichtenagentur ("e-News") stellt an uns die Anforderung den Informationssammlungsprozess und Informationspräsentationsprozess aufzubereiten mithilfe von EDV-technische Unterstützung (Projektname: "InfoWarrior"). Diese Prozesse sollten bewusst einfach [1] gehalten werden.
Als potenzielle Nutzer des Systems sollten natürlich Erfasser [2] [3] der Information [4] eingeplant werden plus Abrufer [5] von solchen Informationen. Die Abrufer von Informationen können einerseits Fremdsysteme [6] sein, andererseits Personen [7].
Daraus resultieren zwei mögliche Szenarien:

- Gesammelte Information wird von CodeWarrior [8] an andere Systeme weitergereicht (XML sollte dabei benutzt werden) [9]
- Gesammelte Information wird von CodeWarrior an Einzelpersonen weitergegeben [10]. Dies setzt natürlich eine entsprechende Aufbereitung [11] Information dar.

Als besondere Anforderung wird die einfache Wartbarkeit von CodeWarrior angemerkt, d.h. u.a. eine lokale Installation von Erfassungskomponenten und Darstellungskomponenten sollte nicht erfolgen [12]. Desweiteren sollte e-News die Möglichkeit haben, die Aufbereitung der Information für Einzelpersonen zu beeinflussen, ebenfalls ohne grossen Aufwand. Für die erste Phase von CodeWarrior wird eine Reduktion auf das Wesentliche [13] angestrebt aber mit den Optionen in weiteren Phasen Erweiterungen, in Kooperation mit unserer Gruppe, am System anzubringen.
Geplante Projektdauer ist bis Ende Jannuar 2002.

[1] What does "einfach = simple" mean, that is what is for customer simple enough?
[2] Who are Erfasser, "e-News" agency employees or we?
[3] How many data collectors are there?
[4] What is going to be prapared, that is, what is information content (text, pictures, multimedia)?
[5] Who are Abrufer, that is who can access the infomation, that is, are there different access levels?
[6] What kind of foreign systems?
[7] Which persons (e-News agency employees, we, or others)?
[8] What is exactly the name of the project, Infowarrior or CodeWarrior?
[9] Are there already other systems existing or not? If yes, does DTD for XML exist or not? That is, how is information delivered?
[10] How is information delivered to the end-users(per e-mail?)?
[11] How will information be prepared?
[12] Does it mean that this a web-based application , or what?
[13] What is for customer "wesentlich = important"?

1.6 First considerations(CRC Session)

In the following we give an overview of classes and few scenarios.
There probably may be many ways to select classes, we tried this method: Starting with the problem statement, we looked for nouns first as we considered them to be potential objects. As a next step, we thought about different scenarios to evaluate the suitability of our very first ideas and to enlarge the correspondence between concept and reality. Finally, we selected several questions that occured during the process. We hope many of them will be answered during this session in addition to the questions that are set as footnote(questons to the problem statement).

1.6.1 Identifying classes

- User
  - Data collector
  - Other users
    - Third parties
    - Foreign systems
- InfoWarrior-application
- E-News employees
- Information material
- E-News agency
- Access manager

Further on, the following canditate classes can be considered:

- Information Receiving Unit
- Information Storing Unit
- Request Receiving Unit
- Front End Unit
- System to System Unit

1.6.2 Some scenarios

- Can data collector XY manipulate all currently available data, and what happens if XY tries to access or change data out of scope of his access rights?
  Suggestion: There should be some kind of access manager that restricts access only to the field relative for collector XY. If XY tries to access date without permission there will be an error message shown.

- What happens if fresh news from some area is/are available?
  Suggestion: There are for example two possibilities:
  - User sets her/his preferences upon registration and gets news delivered per e-Mail automatically if there is something new in the area she/he is registered for.
  - User sets her/his preferences upon registration and when she/he visits homepage of e-news agency, upon login page is created with links to news that user is interested in.
  - others
- How does system respond if data collector XY enters data that already exist in database?
- What happens if user requieres news for which no fresh occurrence is available
  Sugestion:
  - System reports that no fresh news is available
  - System shows the most recent news, that user already knows about

Other scenarios that can be considered are:

- Enter News(manually)
- Delete News
- Edit existing News entries
- News request from automated system
- News browsing

Beyond these basic scenarios there can be some others, if user management is necessary, or we want to provide access control:

- Create User
- Login
- Modify User rights/preferences
- News entry not found
- Access violation

1.7 Open questions

- Can you specify the kind of sources from which information is likely to be received?
- Can you think of other scenarios you think are typical and/or important but not adequately represented in our first cut model?
- Do you have any aspects in mind that are completely missing in our concept?

Chapter 2

CRC Session

2.1 Problem Statement (Version two)

After the first session with the customer, the problem statement has been partly cleared.

- The news agency requires an information collecting and presenting software system (web-based application) that should be simple, which means data collecting, data presentation and meaningfull data input have to be simple organized.
- Data collectors are the reporters [1] employed at the e-News agency. Each reporter prepares at least one, and at most 5 news per day. The news are controlled by director. There are about 5 directors, and each director is in charge for approximately 50 reporters.
- Information should be accessed in two ways:
  - Pull: login
  - Push: via SMS, via e-Mail, ...

  There should be different access levels based on price model, that is:
  - free information available to the whole world
  - for very important, time dependent infromation one must pay in order to access it
- Information content should be all supported types (text, pictures, multimedia).
- Information will be distributed to other foreign systems and single persons. A foreign system is for example, the CNN page.
- There has to be a possibility for preparing information for each person (the preferences should be entered by the user upon registration).
- New information has to be faster available than the old one. Under "old information" is meant the information older than a month.
- In the first phase of this project the collecting and distributing process of information has to be clarified.

[1] Data collector = reporter

2.2 Open Questions

1. Which operating system is going to be used?
2. Who is responsible for hadware (who is going to buy hardware, who is going to maintain hardware, ...) ?
3. Which information is needed for registration?
4. Should a user be able to change her/his preferences at some point later on?

2.3 CRC Card Session

2.3.1 General

- CRC Cards
  CRC Cards are invented by Ward Cunningham and Kent Beck.They were, and are, concered about how to teach people that same deep knowledge of how to use Smalltalk that they had gained. From this question of how to teach objects came the simple technique of CRC (Class-Responsibility-Collaborataion) cards.
  Individual CRC Cards are used to represent objects. The class name is written at the top of the card, responsibilities are listed down the left side, collaborating classes are listed to the right of each responsibility.
  In CRC Session the developers will crowd around a table, each person picking up a card as they describe how the class participates in some use case.

- The Session
  The ideal group for a CRC card session is five or six people. It should be composed of developers, domain experts, and an object-oriented technology facilitator.
  Steps in a CRC card session:

  1. Creating class
     - Find all of the nouns and verbs in the problem statement
       The nouns are a good key to what classes are in the system, and the verbs show what their responsibilities are going to be.
     - Pass out cards and assign the class to the member of the group.
       Each person should be responsible for at least one class. Each person records the name of their class on a card. One class per card.
  2. Responsibilities
     - Add the responsibilities that are obvious from the problem statement or the name of the class. You don't need to find them at all or any. The scenarios will make them more obvious.
  3. Superclasses and Subclasses
     - Superclasses and subclasses can be defined at any time they become obvious.
  4. Attributes

   - Attributes of class are an implementation detail and are generally not defined at all until the design phase, but they can be defined at any time the group thinks it is appropriate.

5. Scenario execution
   - Scenarios are walk-throughs of the functions of the system in detail.
   - Start with scenarios that are part of the system's normal operation first, and then exceptional scenarios, like error recover, later.
   - Decide which class is responsible for this function. The owner of the class picks up his card. When a card is in the air it is an object and can do things.

2.4 InfoWarrior CRC Session

1. Creating class
   - Nouns
     - application
     - user interface
     - news
     - category
     - reporters
     - editors
     - access manager
     - users
     - external systems
     - storage
     - archive
   - Verbs
     - collect
     - distribute
     - input
     - send
     - login
     - control access

2. Some scenarios
   - Can data collector XY manipulate all currently available data, and what happens if XY tries to access or change data out of scope of his access rights? Suggestion: There should be some kind of access manager that restricts access only to the field relative for collector XY. If XY tries to access date without permission there will be an error message shown.

- What happens if fresh news from some area is/are available?
  Suggestion: There are for example two possibilities:
  - User sets her/his preferences upon registration and gets news delivered per e-Mail automatically if there is something new in the area she/he is registered for.
  - User sets her/his preferences upon registration and when she/he visits homepage of e-news agency, upon login page is created with links to news that user is interested in.
  - others
- How does system respond if data collector XY enters data that already exist in database?
- What happens if user requieres news for which no fresh occurrence is available
  Sugestion:
  - System reports that no fresh news is available
  - System shows the most recent news, that user already knows about
- Enter News(manually)
- Delete News
- Edit existing News entries
- News request from automated system
- News browsing
- Create User
- Login
- Modify User rights/preferences
- News entry not found
- Access violation

2.5 Essential Use Case Modeling

2.5.1 General

- Use Cases
  - "a behaviorally related sequence of transactions in a dialogue with the system",(Ivar Jacobson)
  - For example: Use Cases for a word processor might include building an index or inserting a picture.
- Use Case Diagrams
  - provide a way of describing the extarnal view of the system and its interactions with the outside world. In this representation the outside world is represented as actors.

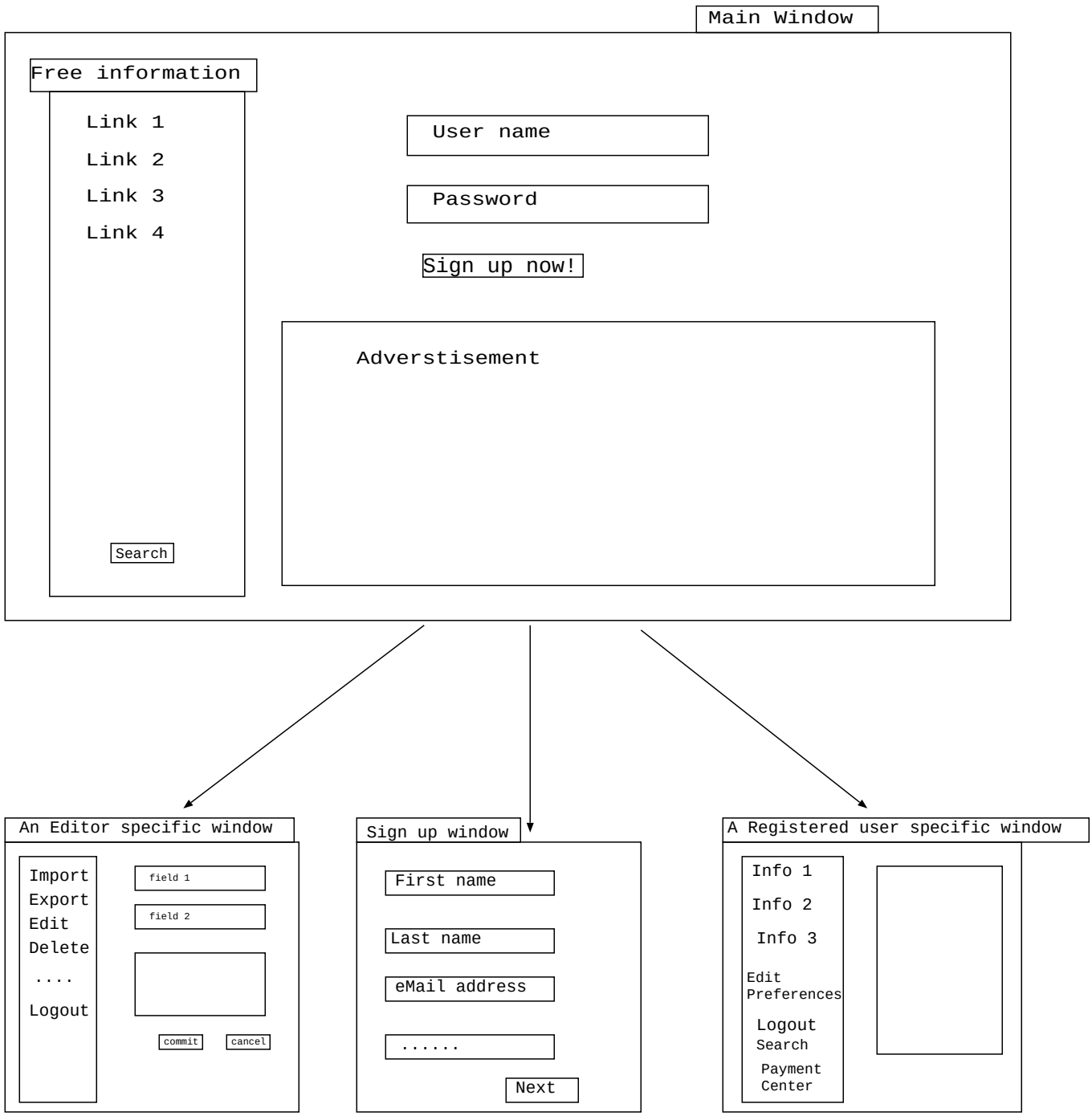
- Actors
  - Actors are roles played by various people, or other computer systems. One person may play many roles, and a role may have many people playing it. Use Cases are then typical interactions that the actor has with the system.
- Relationships
  - are represented through the lines between actors and Use Cases, and between Use Cases and Use Cases.

2.5.2 Use Cases in InfoWarrior

- Actors
  - User
  - Reporter
  - Editor
  - InfoWarrior
  - News
  - ...
- Relationships
  - Subscribe for News
  - Login
  - Change Preferences
  - Enter Information
  - Modify Information
  - Delete Information
  - Archieve Information
  - Send Information
  - ...

2.6 Essential User Interface Prototyping (First considerations)

System contains one main window with links/ buttons to the other pages/windows. It is the same both for normal users and for editors. Upon login new User Interface is shown with features available depending on access level.


Main Window
Free information
Link 1
Link 2
Link 3
Link 4
Search
User name
Password
Sign up now!
Adverstisement
An Editor specific window
Import
Export
Edit
Delete
....
Logout
field 1
field 2
commit
cancel
Sign up window
First name
Last name
eMail address
......
Next
A Registered user specific window
Info 1
Info 2
Info 3
Edit Preferences
Logout
Search
Payment Center

Chapter 3

Use Cases Description

3.1 Identifying Actors

- Editor
- Reporter
- End-user
- External System
- Administrator
- InfoWarrrior Application
- Access Control Manager
- News
- InfoWarrior Search Machine
- Storage Unit

3.2 Identifying Use Cases

- Subscribe to InfoWarrior Services
- Unsubscribe from InfoWarrior Services
- User logs in
- End-User edits Preferences
- Editor edits News
- User searches News
- Administrator modifies User Data

3.2.1 Description

1. **Name:** Subscribe to InfoWarrior Services
   Description: Somebody wants to subscribe to InfoWarrior services. She/he opens the official InfoWarrior web-page and clicks "Sing up now!" link and enters necessary information (username, password, e-mail address,...) in order to create an account. The user can decide weather she/he wants to use just free or billable InfoWarrior services as well. After submitting this information she/he becomes a user of the corresponding InfoWarrior services.

2. **Name:** Unsubscribe From InfoWarrior Services
   Description: A user wants to unsubscribe from InfoWarrior Services. She/he clicks "Unsubscribe From InfoWarrior Servceses" link. Then she/he will be asked does she/he want to unsubscribe from free InfoWarrior services, billable InfoWarrior services or from the both (free and billable) services. After submitting her/his answer, she/he won't be a user of corresponding InfoWarrior services anymore.

3. **Name:** User logs in
   Description: A user (editor, reporter, administrator, customer) wants to log in. She/he opens the official InfoWarrior web-page and enters her/his user name and password in the corresponding fields. If a user (editor, reporter, end-user) enters incorrect user name or password a login incorrect window will be opened, and a user can try to log in again.Upon successful login new User Interface is shown, with features available depending on access level. That means, if for example, a reporter logs in, a reporter specific window will be opened.

4. **Name:** User logs out
   Description: A user wants to log out. She/he selects a "Logout" link. Then will be verified by the system does she/he really wants to log out or not. She/he submits her/his decision to the system, and the message "Successfully logged out" will be shown and the InfoWarrior homepage will be opened.

5. **Name:** Customer edits Preferences
   Description: A customer wants to edit her/his preferences. After login every user has a link "Edit Preferences". When she/he clicks this link, a page with her/his preferences will be opened. The preferences page contains user specific information : user name, encrypted password, e-mail address, mobile phone number, etc., that is, it contains all information given upon subscribing to InfoWarrior services. The user can now modify her/his preferences. After editing, she/he clicks a button "Submit changes" in order to apply changes to her/his account.

6. **Name:** Editor edits News
   Description: An editor wants to edit news. That means she/he wants to add, delete, or alter news. Upon login every editor has buttons for each of these actions. When she/he clicks one of these buttons, she/he can browse the files, that she/he wants to add/delete/alter. After editing she/he has to commit "Submit changes" in order to apply changes to the system storage unit.

7. **Name:** User searches News in the Primary Storage System
   Description: A user(editor, reporter, end-user) wants to search for recent news.

She/he clicks "Search" link in order to start search page. A user enters wanted pattern in the corresponding field and clicks "Start Search" button. The InfoWarrior Search Machine outputs all news corresponding to the searching pattern.

8. **Name:** User searches News in the Secondary Storage System
   Description: A user(editor, reporter, end-user) wants to search for older news. She/he clicks "Advanced Search" link in order to start the advanced search page. Afterwards she/he enters wanted pattern in the corresponding field and clicks "Start Search" button. The InfoWarrior Search Machine outputs all news corresponding to the searching pattern.

9. **Name:** Administrator creates a User
   Description: The administrator wants to create a new reporter or editor in the system. From the admin main page he uses the create user page and enters the users essential data and privileges. After the admininistrator confirms the data, the system creates a new user.

10. **Name:** Administrator searches User
    Description The administrator wants to view existing users. He enters search criteria for the users to be returned and clicks search. The system returns a list of matching users, from which the administrator can select a user.

11. **Name:** Administrator modifies User Data
    Description: The administrator wants to modify user data or privileges. He searches the specific user as described in "Administrator searches User". He selects to modify the user and the user modification page is returned. There he changes the data and confirms them by clicking a "submit button. The system commits the changes to the user database.

12. **Name:** Administrator deletes a User
    Description: The administrator wants to delete a user. He searches the specific user as described in "Administrator searches User". He selects to delete the user. After confirmation from the administrator, the system user is removed from the user database.

13. **Name:** Administrator creates a category
    Description: The administrator wants to create a new category for news. He selects the link on the main page and enters the name of the category. After the administrator confirms the data, the system creates a new category

14. **Name:** Administrator deletes a category
    Description: The administrator wants to delete a category. He selects the link on the main page and on the next page the category he wants to delete. He gets a warning how much news will be deleted on cascade if he will delete the category. After the administrator confirms the data, the system deletes the category and all news assigned to it.

3.3 Essential User Interface Description

3.3.1 Customer User Interface

Since the costumer front-end is mainly web based, the user interface consists of web pages.

Start Page

the home page.
it is the first thing a (potential) costumer sees after typing in the url, so it should be attractive, clear and well-structured. It features the latest headlines and provides links to the most important subpages.
Important features: User Login, link to Sign Up Page.

Sign Up Page

main purpose: customer registration.
Features input boxes for user data, like name, surname, country, state, e-mail address, a login name and password etc, and options for pay level, payment mode.

Category Page

main purpose: structuring of articles, facilitating efficient browsing.
categories might be economy, science, politics, with subcategories world politics, regional politics etc. links lead to other category pages (related categories or subcategories), and article pages. Also provides an input box for quick search for articles within the category.

Search Page

provides advanced search options to fine-tune a search, like searching for older articles, AND/OR junction of keywords, searching within a specific category etc.
search results are displayed on Search Result Page.

Search Result Page

presents links to articles the search engine has found, also information about date, content, and category. may also show headlines of articels of a higher pay level than user's, with suggestions to sign up to higher level.

Welcome User Page

page that opens after user login, featuring headlines of user's preferred categories.

Preferences Page

main purpose: (registered) users can set different options like default search settings, preferred categories etc and subscribe to services that are within their pay level.

Article Page

Displays article text in attractive, readable form as well as pictures, offers accompaning multimedia files for download etc. Links to related articles.

3.3.2 Staff User Interface

As the Staff User has a different approach to the InfoWarrior system than the customer, the requirements to a user interface are also quite different. The Staff User needs an interface to get his/her work efficiently done. If the Staff User Interface will be partiallly or fully web based like the Customer Interface remains open at this stage of design. The login procedure could be done via the Customer Interface Start Page.

- **Reporter Interface**

  #### Reporter Page

  Each reporter's home page / main window, that is opened after login.
  It gives an overview on the reporter's recent work, listing article headlines with information about date and whether the article has yet been submitted and released. Links to Edit Article Page (to create/edit articels), a task list, a search page/window to search for articles, and maybe an emailprogram to send email to the responsible editor.

  #### Edit Article

  enables reporters to compose, edit, and save articles, add files that accompany the article text (like pictures, sound files, and other multimedia), and submit them to the responsible editor. Input boxes for text, abstract and headline.

- **Editor Interface**
  Since editors can also write articles, their interface provides the same functionality as the reporters', with additions.

  #### Editor Page

  Home page / main window. Similar to the Reporter Interface, it displays recent article headlines from the editor's section, with information about date, author, and whether the article was yet released. Article headlines link to View Article Page/Window. Link to email program to send tasks to reporters.

  #### View Article

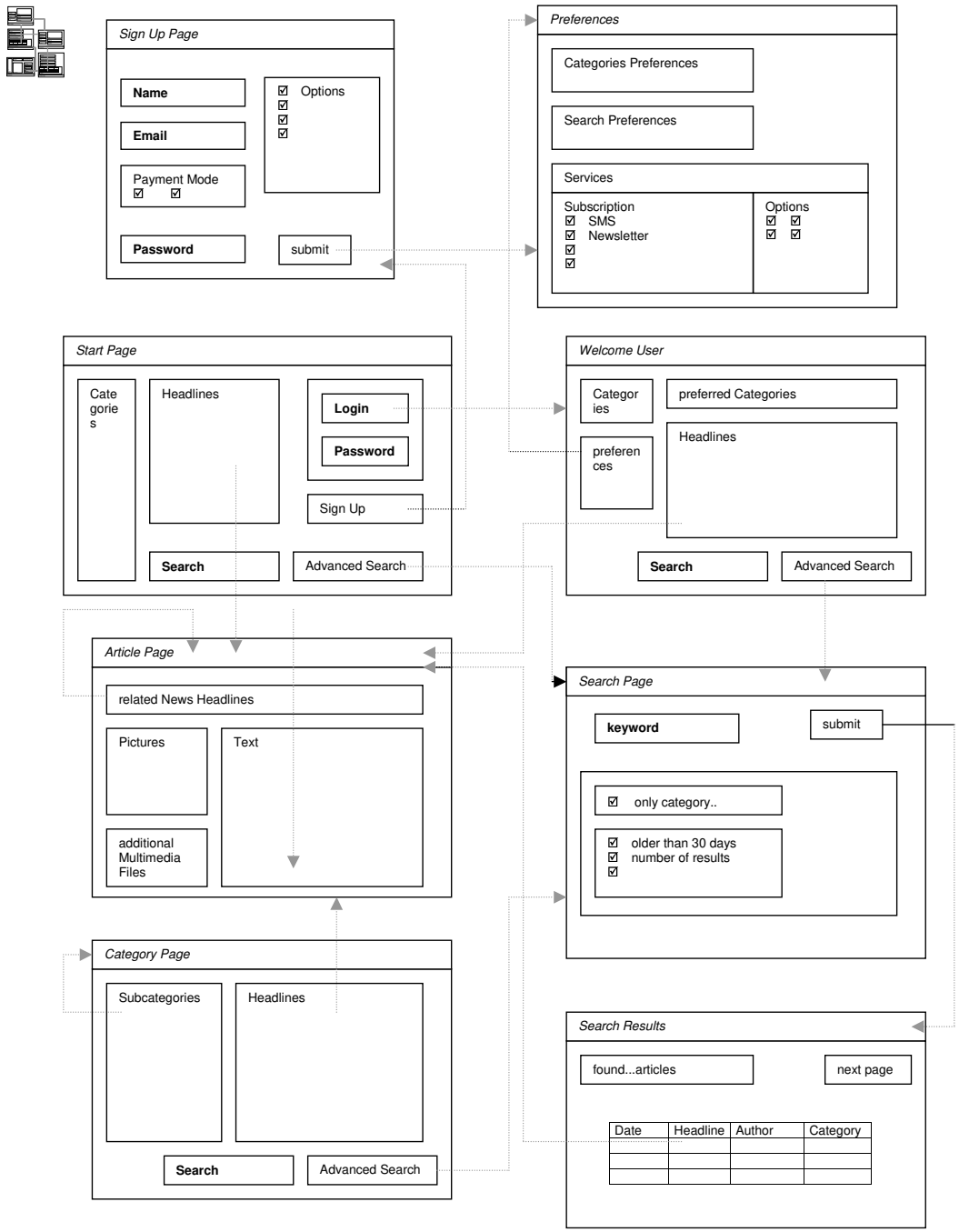
  Displays articles as they are seen on the homepage/Customer Interface (ie Article Page). Buttons to release, delete or edit article (-¿ Edit Article Page).

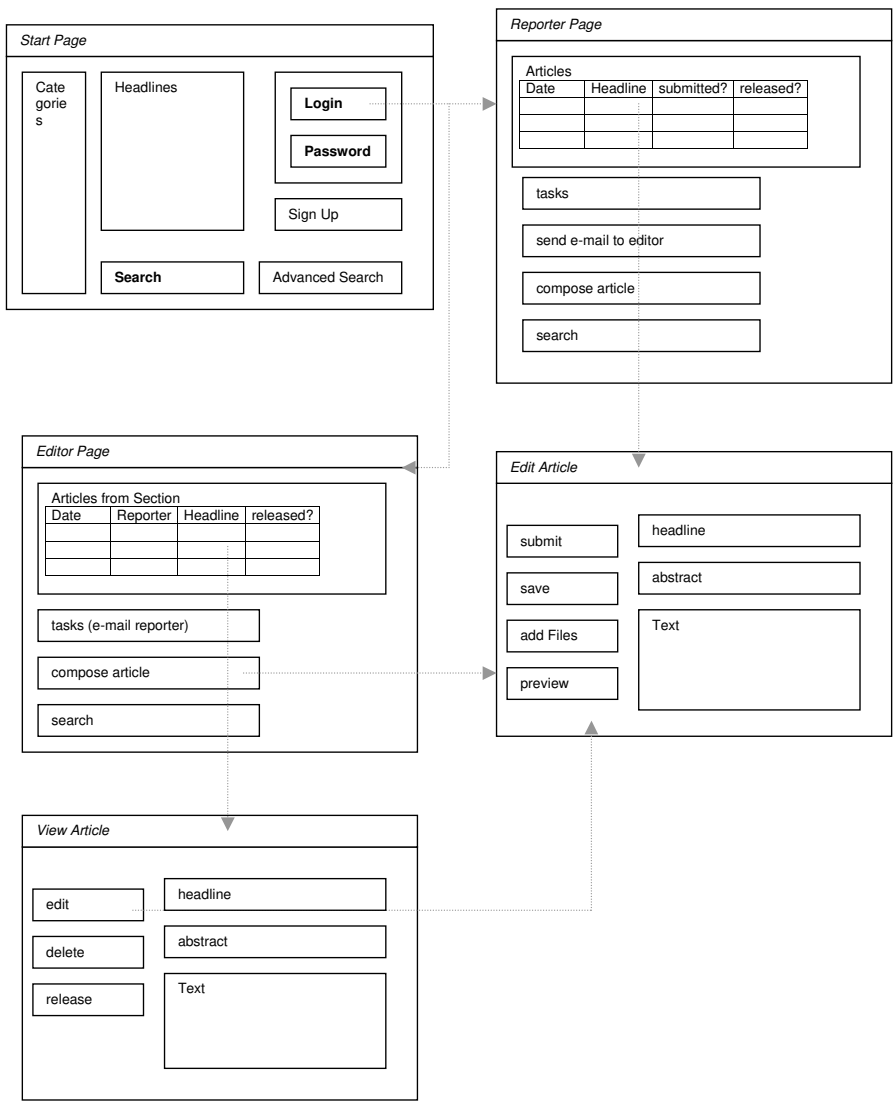
Edit Article

As in Reporter Interface. (See above)

- **Administrator Interface**

3.4 Diagrams/Drafts




Start Page
Cate
gorie
s
Headlines
Login
Password
Sign Up
Search
Advanced Search
Reporter Page
Articles
Date
Headline
submitted?
released?
tasks
send e-mail to editor
compose article
search
Editor Page
Articles from Section
Date
Reporter
Headline
released?
tasks (e-mail reporter)
compose article
search
Edit Article
submit
save
add Files
preview
headline
abstract
Text
View Article
edit
delete
release
headline
abstract
Text

3.5 Open Questions (3)

1. Can an editor edit her/his preferences?
2. Can a reporter edit her/his preferences?

Chapter 4

Classes

4.1 Classes List

- Storage
- UserStorage
- Message Storage
- Primary Message Storage
- Secondary Message Storage
- User
- Customer
- Reporter
- Editor
- Administrator
- External System
- UserInterface
- Customer Interface
- ReporterInterface
- EditorInterface
- AdministratorInterface
- SystemInterface
- MediaObject
- Picture

- Video
- Audio
- Message
- Messageset
- Paylevel
- Category
- ACLSecuredObject
- SecuredObject
- ACL
- Permission
- Roles
- UserCore
- MessageCore
- Request
- Response
- XMLConverter

4.1.1 Classes Description

1. **Storage**
   Abstract parent class for storing purposes.
   Public Operations:
   Connect () : Boolean
   Close () : Boolean
   GetReport () : REPORT
   GetSize () : Integer

2. **UserStorage**
   Class for Users.
   Public Operations:
   GetUser (id) : USER
   AddUser (USER) : boolean
   UpdateUser (USER) : boolean
   DeleteUser (id) : boolean
   GetRole () : ROLE
   DeleteRole (ROLE) : boolean

UpdateRole (ROLE) : boolean
AddRole (ROLE) : boolean
AddPaylevel(PAYLEVEL): boolean
UpdatePaylevel (PAYLEVEL) : boolean
DeletePaylevel (id) : boolean

3. **MessageStorage**
   Abstract class for storing messages.
   Public Operations:
   GetMessage (id) : MESSAGE
   AddMessage (MESSAGE) : boolean
   FindMessage (REQUEST) : RESPONSE

4. **PrimaryMessageStorage**
   Class for storing current messages.
   Public Operations:
   UpdateMessage (MESSAGE) : boolean
   DeleteMessage (id) : boolean AddCategory (CATEGORY) : boolean
   UpadteCategory (CATEGORY) : boolean

5. **SecondaryMessageStorage**
   Class for storing archived messages.

6. **User**
   Abstract class for storing real person users of the system.
   Public Attributes:
   Name
   Address
   EMail
   Phone
   Mobile
   CreditCard
   Password
   Subscriptions
   LastLogin
   Birthdate
   Company
   Operating System
   Browser
   Homepage
   Login

7. **Customer**
   Class for Customers of the System.

Public Attributes:

8. **Reporter**
   Class for Roprters.
   Public Attributes:
   Department
   Superior
   Public Operations:
   getArticles(): MESSAGES
   getArticlesPending: MESSAGES

9. **Editor**
   Class for editors of the system.

10. **Administrator**
    Class for administrators of the system.

11. **External System**
    Abstract class for external systems.

12. **UserInterface**
    Abstract class for UserInterfaces.

13. **CustomerInterface**
    Class for customers of the system.

14. **ReporterInterface**
    Class for reporters.

15. **EditorInterface**
    Class for editor.

16. **AdministratorInterface**
    Class for administrators of the System.

17. **SystemInterface**
    Class for access to systems by forgein systems.
    Public Operations:
    retrieveRequest(REQUEST): RESPONSE

18. **Media Object**
    Abstract class for storing media objects.
    Public Attributes:
    MIME Type

19. **Picture**
    Class for Pictures.
    Public Attributes:
    Heigth
    Width
    Depth
    Date
    dpi

20. **Video**
    Class for Videos.
    Public Attributes:
    Heigth
    Width
    Depth
    Length
    Date

21. **Audio**
    Class for Audio files.
    Public Attributes:
    Length
    SampleRate

22. **Video**
    Class for Pictures.
    Public Attributes:

23. **Message**
    Class for messages.
    Public Attributes:
    Author
    Title
    Abstract
    Text
    Date
    Editor
    Category

PayLevel
LastChanged
Released

24. **MessageSet**
    Class for messages.
    Public Operations:
    GetMessages():MESSAGES

25. **Paylevel**
    Class for paylevels.
    Public Attributes:
    Name
    Cost

26. **Category**
    Class for categorys.
    Public Attributes:
    Name
    Abstract

27. **ACLSecuredObject**
    Abstract class for accesscontrolledlist Objects .

28. **Secured Object**
    Interface for ACLSecuredObject.
    Public Operations:
    isAllowed (USER, ACTION) : boolean
    GetParent()

29. **ACL**
    Class for AccessControlledLists.
    Public Operations:
    GetPermission()
    GetRoles()
    HasPermission()

30. **Permission**
    Class for permissions.
    Public Attributes:
    Name

31.

32. Roles Class for roles.
    Public Attributes:
    Name
    Public Operations:
    GetMembers(): USER
    AddMembers(USER): boolean
    RemoveMembers(USER): boolean
    isMember(USER): boolean

33. **UserCore** Class for user operations.
    Public Operations:
    executeRequest(REQUEST):RESPONSE

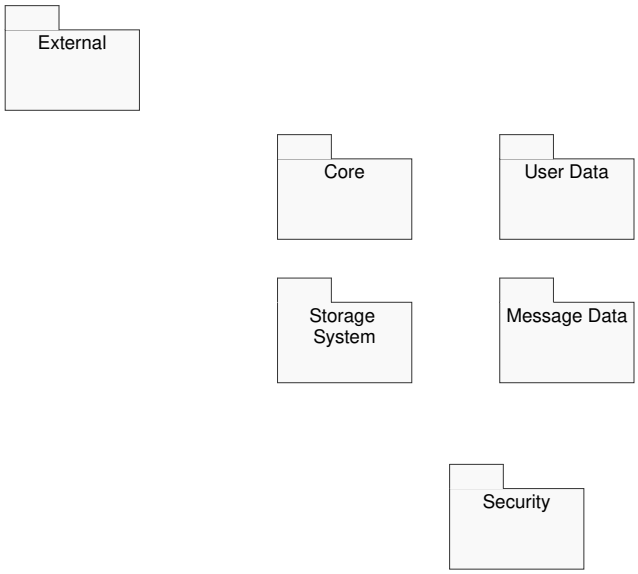
34. **MessageCore** Class for messages.
    Public Operations:
    executeRequest(REQUEST):RESPONSE

35. **Request**
    Class for requests.
    Public Attributes:
    Command

36. **Response**
    Class for search results.
    Public Attributes:
    Errorcode
    Results

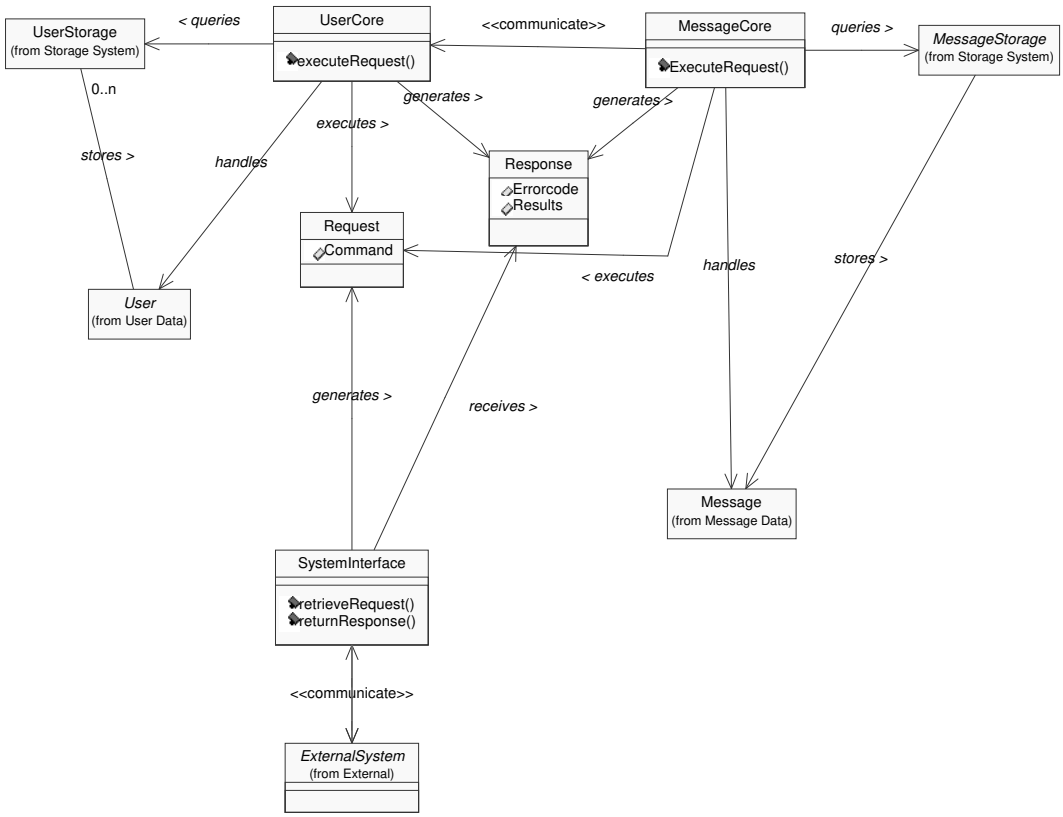
37. **XMLConverter**
    Interface for converting XML to Strings and vice versa.
    Public Operations:
    XML2Request() Response2XML()

InfoWarrior Packages



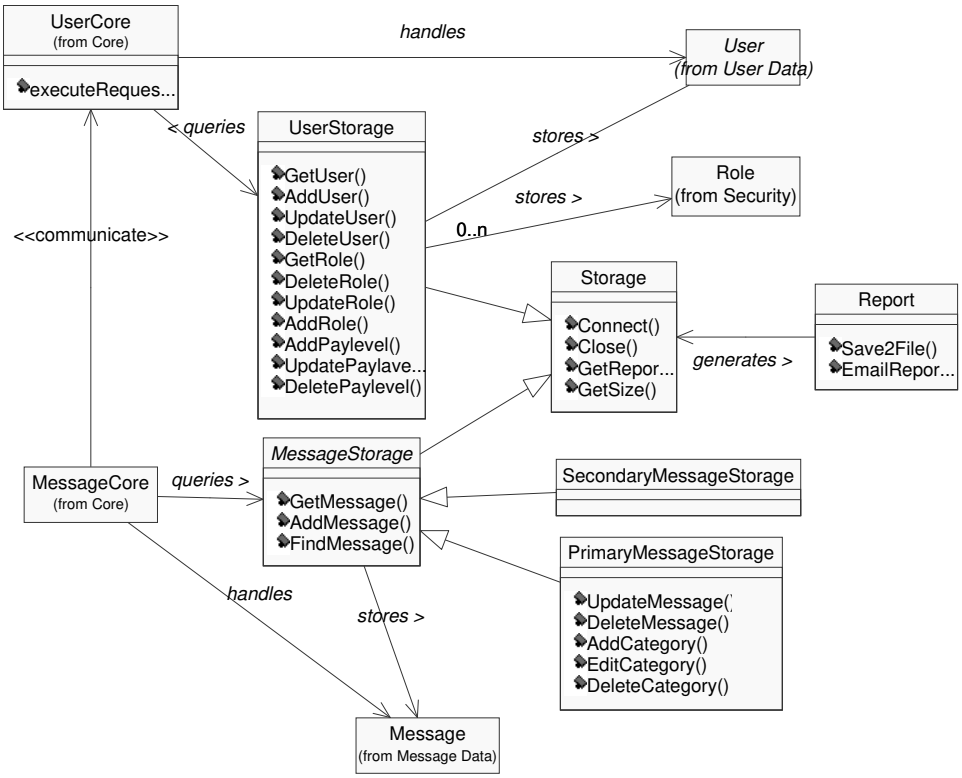
File: d:\users\albert\Class Diagram.mdl 16:12:41 Freitag, 30. November 2001 Class Diagram: Logical View / Main Page 1

Core Package



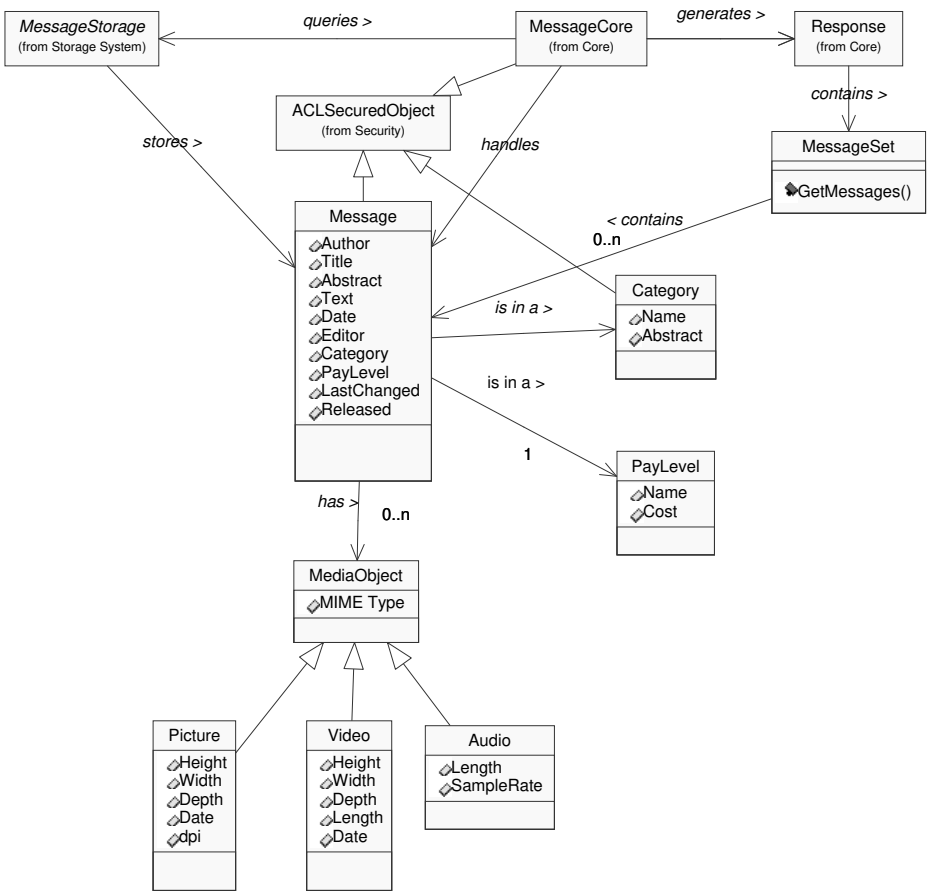
File: d:\users\albert\Class Diagram.mdl 16:14:25 Freitag, 30. November 2001 Class Diagram: Core / Core Page 1

Storage Package

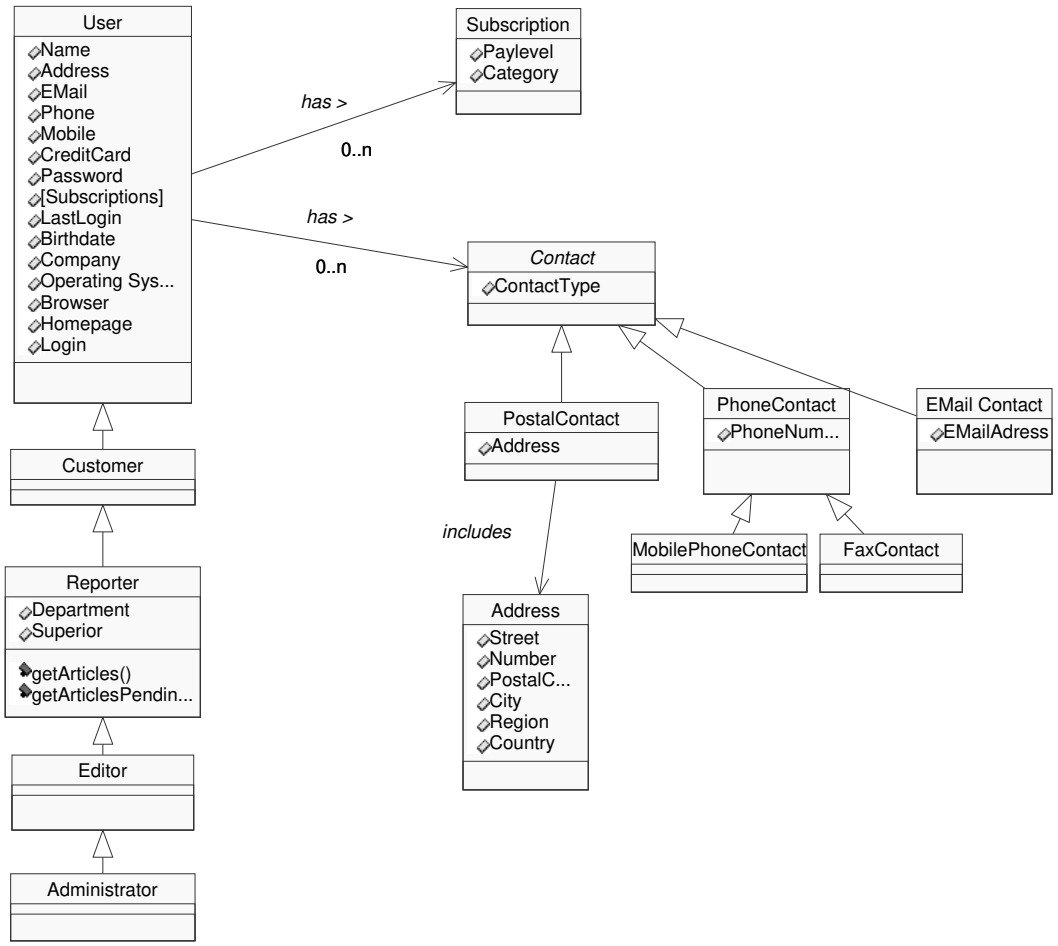


File: d:\users\albert\Class Diagram.mdl 16:18:52 Freitag, 30. November 2001 Class Diagram: Storage System / Storage Page 1

Message Package

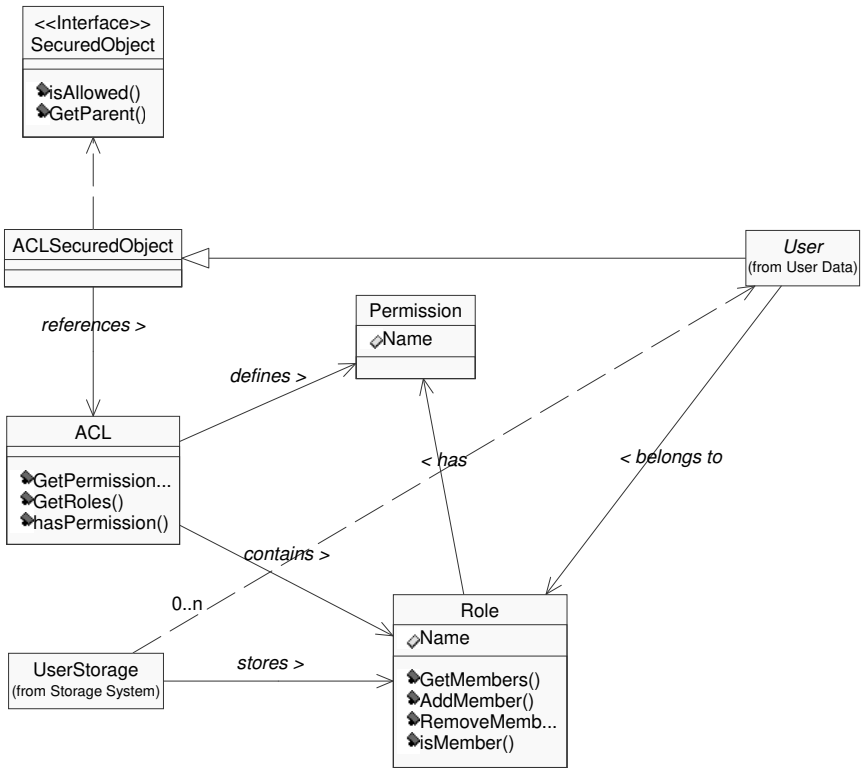


User Package



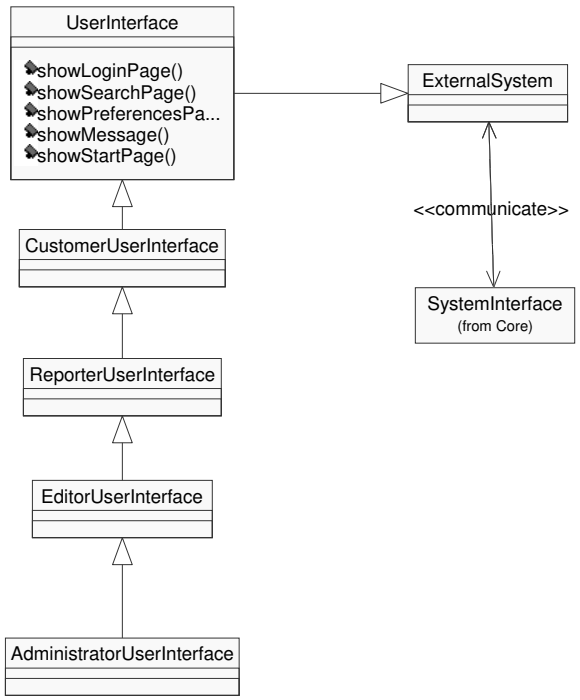
File: d:\users\albert\Class Diagram.mdl 16:19:59 Freitag, 30. November 2001 Class Diagram: User Data / User Data Page 1

Security Package



File: d:\users\albert\Class Diagram.mdl 16:18:15 Freitag, 30. November 2001 Class Diagram: Security / Security Page 1

External Package



File: d:\users\albert\Class Diagram.mdl 16:15:34 Freitag, 30. November 2001 Class Diagram: External / External Page 1

Chapter 5

Interface Prototype

5.1 HTML Prototype

5.1.1 Introduction

The HTML prototype is the major communication interface with the InfoWarrior system. Therefore it has to provide interfaces for all necessary tasks, including doucment editing and system administration. As this is exactly, what the essential ui design was done for, it could be modelled after the screens presented in the previous chapter.
As the functionality was already defined by the essential user interface, the focus of the prototype was to incorporate this functionality in a consistent and easy-to-use graphical interface.

5.1.2 Design Principles

- All pages should have a consistent look and feel.
- Important functions should be easily accessible, if possible with a single click.
- Each page must contain a navigation element, that allows the user to find out his current position in the site structure and to reach other important pages within the site
- For each section of the page (start page, category pages, ..) the most important content should be visible immediatly, lesser functions are reachable via a menu structure.
- The staff sites (editor, reporter, administrator) are designed to match the typical workflow. Important information and common tasks have to be visible on the main screen.

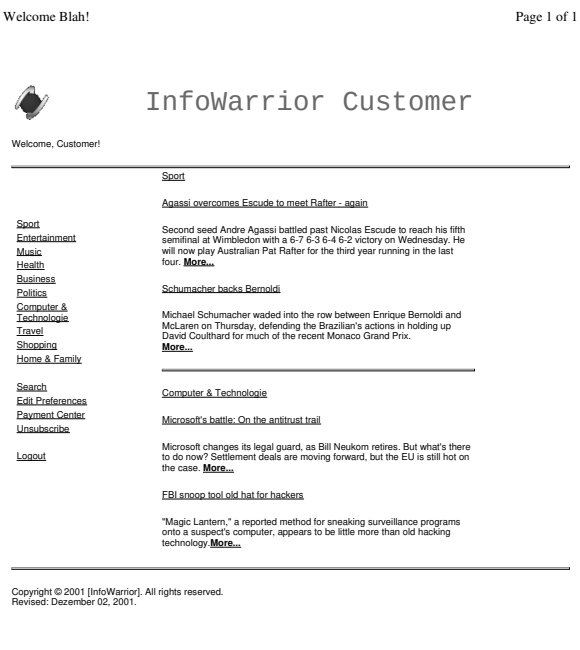


Figure 5.1: Customer Page

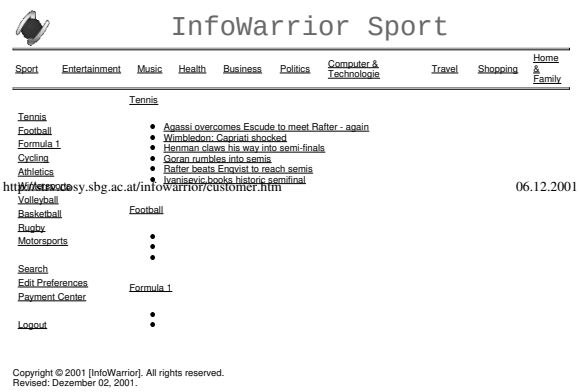


Figure 5.2: Sport Category

5.2 WML Prototype

5.2.1 Introduction

The WML prototype is baed on a reduced version of the essential user interface design. The original design had to be modified to fit on the small screen of a mobile device. Only the Client Interface was modelled in WML, as we agreed, that the staff interfaces would be unusable on a cell phone.

5.2.2 Design Principles

We used a set of basic principles to get a usable desgin.

- The WML site should not be a port of the HTML page, but a completely seperate design
- The WML site should still be similar enough to the web site to allow users of the web page to utilize it without an extra learning process

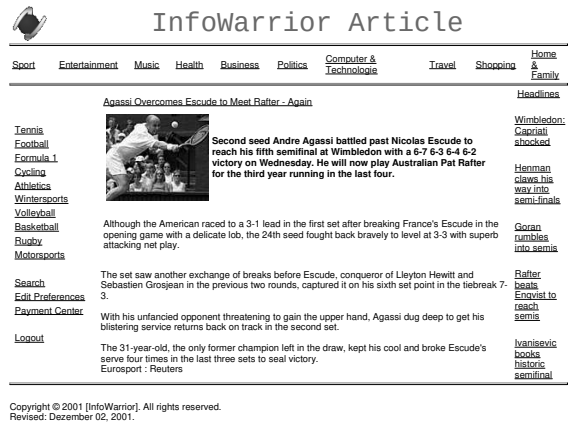


Figure 5.3: Sport Article

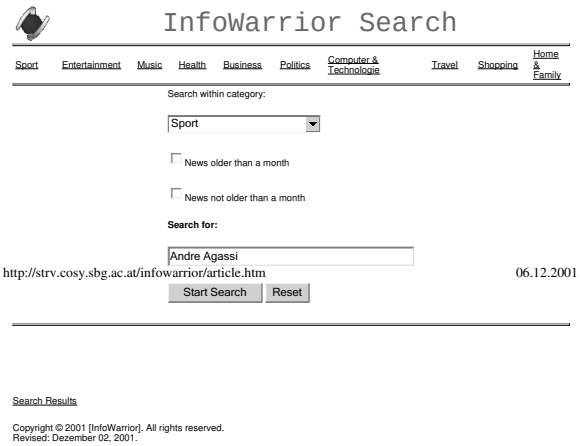
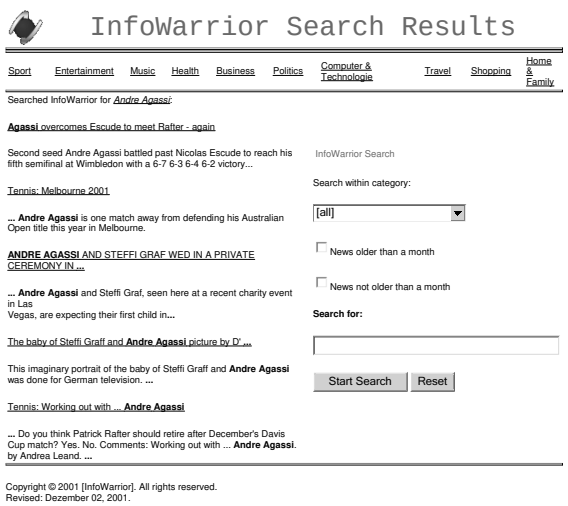


Figure 5.4: Basic Search

- All functionality of the system should be available via WAP
- All main items of the site should be reachable from every page
- Each page's information should be visible immediatly without scrolling. Navigation elements may never hide content
- Each page should have self-explanatory title and header



InfoWarrior Search Results

Sport | Entertainment | Music | Health | Business | Politics | Computer & Technologie | Travel | Shopping | Home & Family

Searched InfoWarrior for *Andre Agassi*:

Agassi overcomes Escude to meet Rafter - again

Second seed Andre Agassi battled past Nicolas Escude to reach his fifth semifinal at Wimbledon with a 6-7 6-3 6-4 6-2 victory...

Tennis: Melbourne 2001

... Andre Agassi is one match away from defending his Australian Open title this year in Melbourne.

ANDRE AGASSI AND STEFFI GRAF WED IN A PRIVATE CEREMONY IN ...

... Andre Agassi and Steffi Graf, seen here at a recent charity event in Las Vegas, are expecting their first child in...

The baby of Steffi Graf and **Andre Agassi** picture by D' ...

This imaginary portrait of the baby of Steffi Graf and **Andre Agassi** was done for German television. ...

Tennis: Working out with ... **Andre Agassi**

... Do you think Patrick Rafter should retire after December's Davis Cup match? Yes. No. Comments: Working out with ... **Andre Agassi**. by Andrea Leand. ...

InfoWarrior Search

Search within category:

[all]

News older than a month

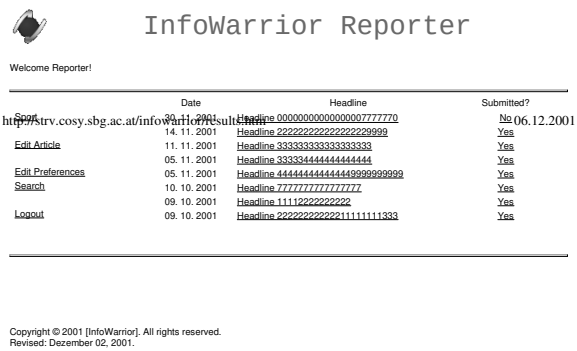
News not older than a month

Search for:

Start Search | Reset

Copyright © 2001 [InfoWarrior]. All rights reserved.
Revised: Dezember 02, 2001.

Figure 5.5: Search Results



InfoWarrior Reporter

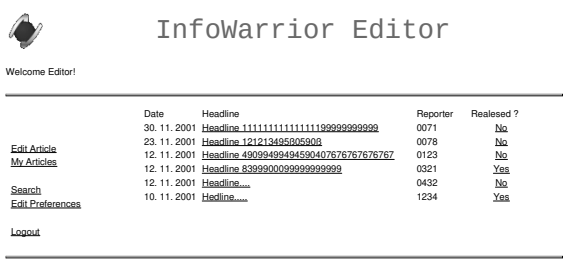
Welcome Reporter!

	Date	Headline	Submitted?
Sport	30. 11. 2001	Headline 000000000000007777770	No
	14. 11. 2001	Headline 22222222222222229999	Yes
Edit Article	11. 11. 2001	Headline 333333333333333	Yes
	05. 11. 2001	Headline 33333444444444444	Yes
Edit Preferences	05. 11. 2001	Headline 444444444444499999999	Yes
Search	10. 10. 2001	Headline 77777777777777	Yes
	09. 10. 2001	Headline 11112222222222	Yes
Logout	09. 10. 2001	Headline 2222222222211111111333	Yes

http://strv.cosy.sbg.ac.at/infowarrior/results.htm 06.12.2001

Copyright © 2001 [InfoWarrior]. All rights reserved.
Revised: Dezember 02, 2001.

Figure 5.6: Reporter Interface



InfoWarrior Editor

Welcome Editor!

	Date	Headline	Reporter	Realesed ?
	30. 11. 2001	Headline 11111111111111199999999999	0071	No
	23. 11. 2001	Headline 1212134950505903	0078	No
Edit Article	12. 11. 2001	Headline 430994994945904076767676767	0123	No
My Articles	12. 11. 2001	Headline 8399900099999999999	0321	Yes
Search	12. 11. 2001	Headline....	0432	No
Edit Preferences	10. 11. 2001	Hedline.....	1234	Yes
Logout				

http://strv.cosy.sbg.ac.at/infowarrior/reporter.htm 06.12.2001

Copyright © 2001 [InfoWarrior]. All rights reserved.
Revised: Dezember 03, 2001.

Figure 5.7: Editor Interface

Figure 5.8: Administrator Interface

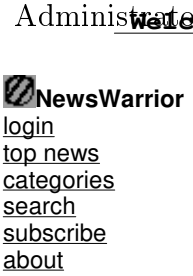


Figure 5.9: Start Page

Figure 5.10: Customer Page

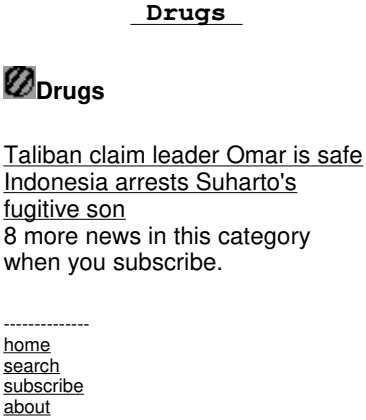


Figure 5.11: Category

Drugs

Taliban claim leader Omar is safe

The Taliban on Wednesday denied that U.S. airstrikes had hit a leadership compound a day earlier and said the Taliban's supreme leader, Mullah Mohammad Omar, was safe and unharmed.

[full text]
---
other drug news
home
search
subscribe
about

Figure 5.12: Article

Search Results

Results for "Taliban" in

Taliban claim leader Omar is safe
[search again]

---
home
categories
subscribe
about

Figure 5.13: Search Results

Chapter 6

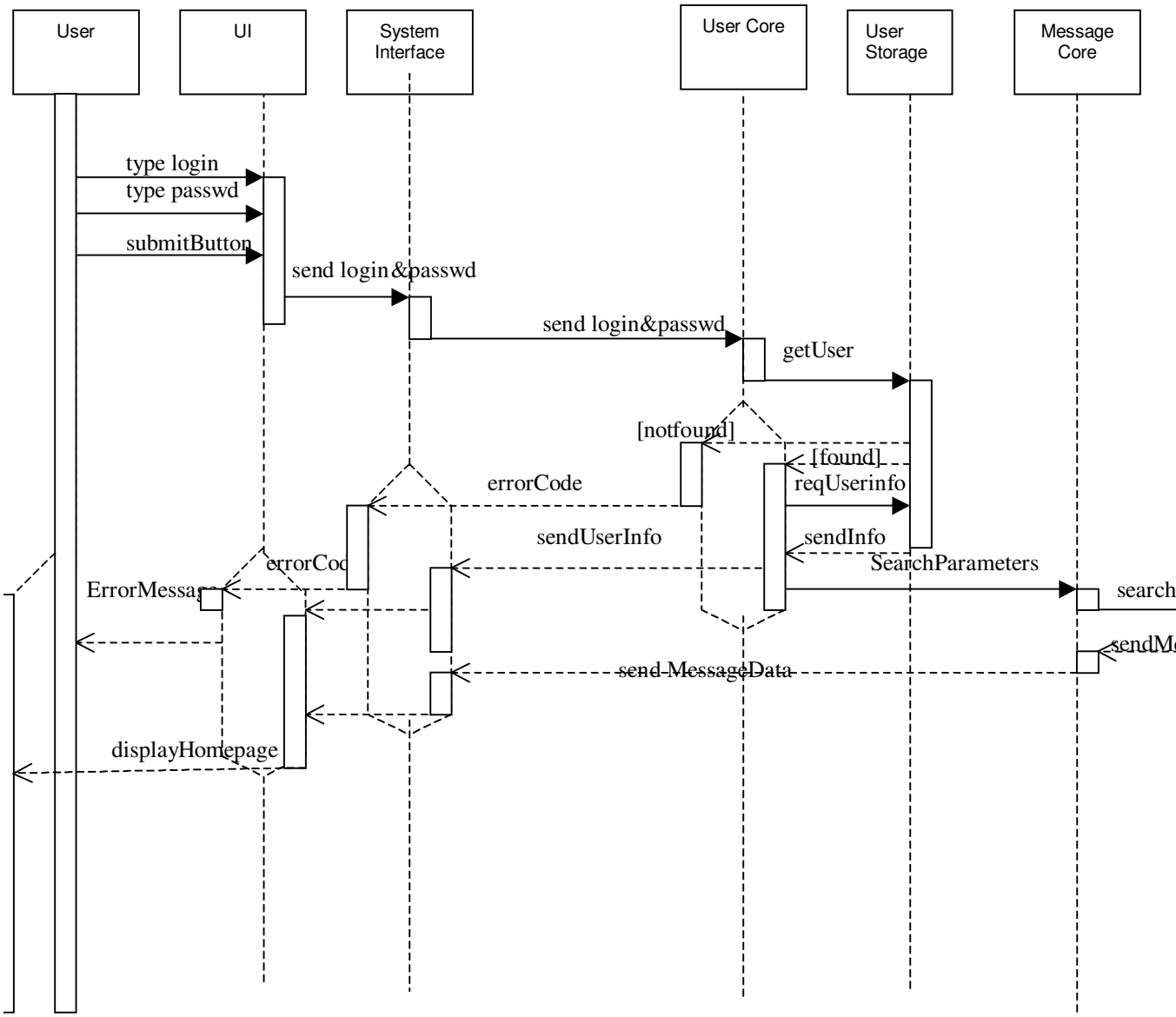
Sequence and Activity Diagrams

Sequence diagrams are used to model the logic of usage scenarios (indicates the description of a potential way your system is used.).
Activity diagrams are used to model the logic of use cases, business proceses or methods. Activity diagrams are equivalent to the flow diagrams in procedural programing.
Sequence and activity diagrams in InfoWarrior we modelled after Use Cases presented in Chapter 3, Class Diagrams presented in Chapter 4 and User Interface Prototype presented in Chapter 5.

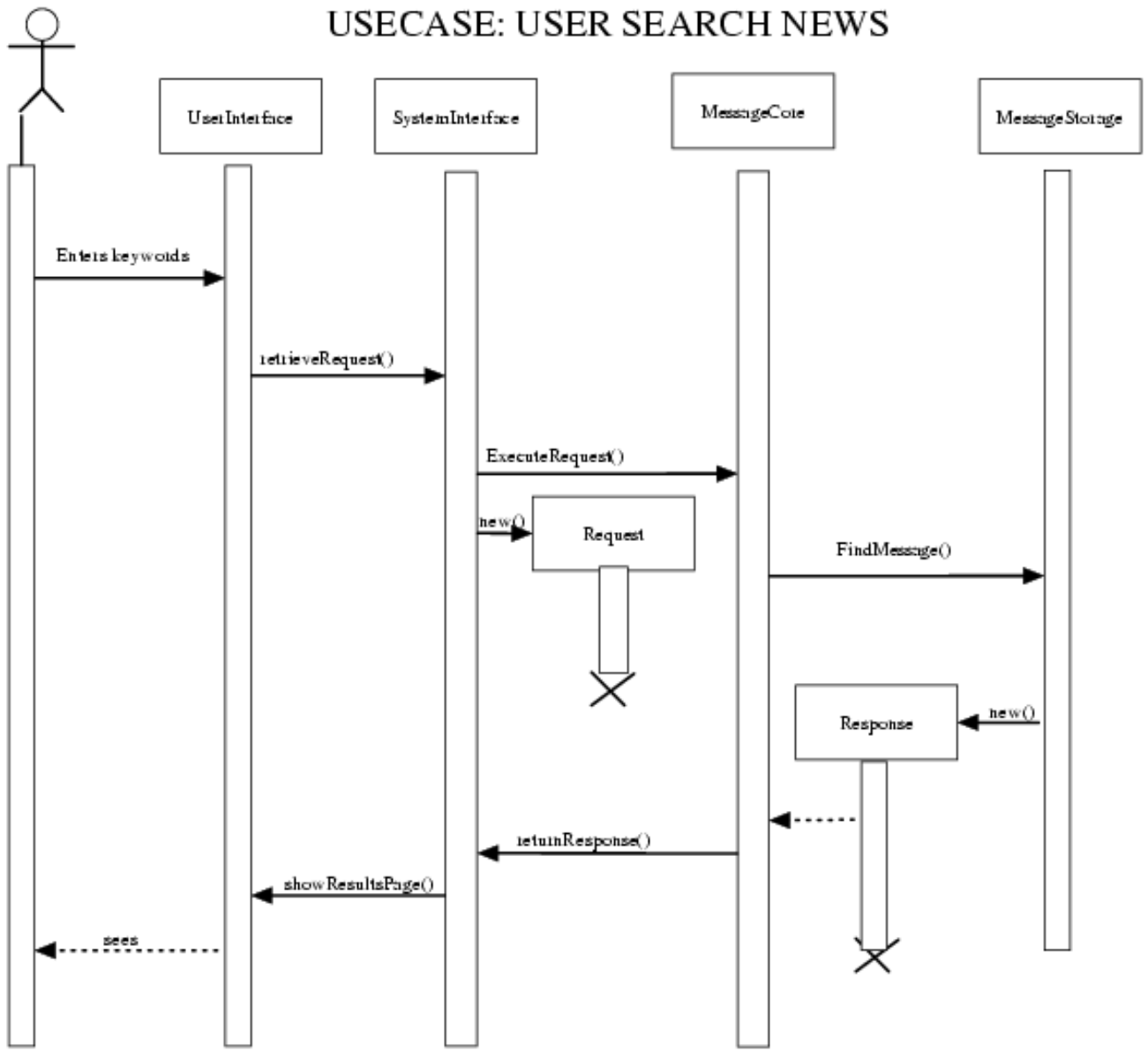
6.1 Sequence Diagrams in InfoWarrior

6.1.1 User Specific Sequence Diagrams

1. Log in

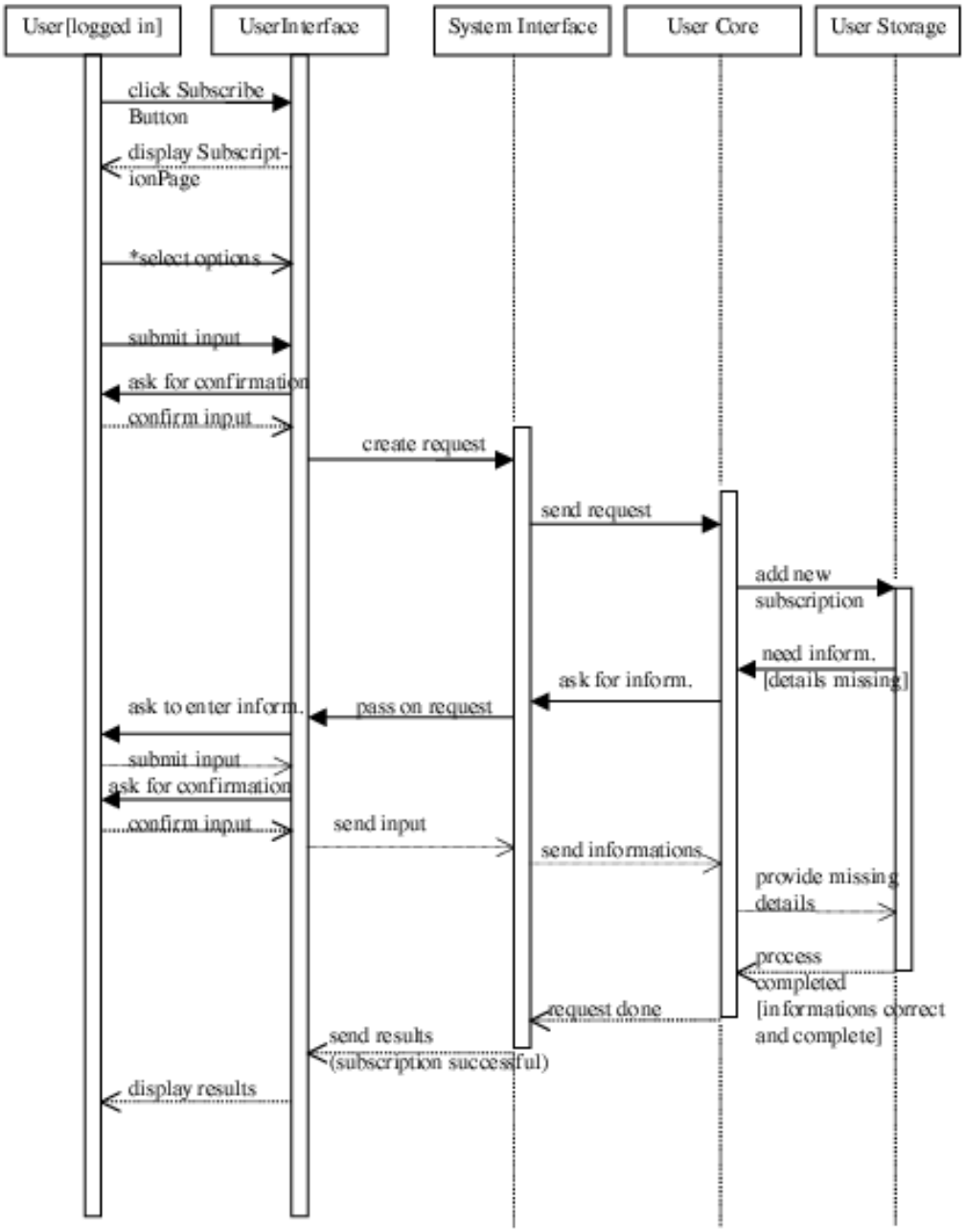

User
UI
System Interface
User Core
User Storage
Message Core
type login
type passwd
submitButton
send login&passwd
send login&passwd
getUser
[notfound]
[found]
errorCode
reqUserinfo
sendUserInfo
sendInfo
errorCod
SearchParameters
ErrorMessag
search
send MessageData
displayHomepage

2. Search


USECASE: USER SEARCH NEWS
UserInterface
SystemInterface
MessageCore
MessageStorage
Enters keywords
retrieveRequest()
ExecuteRequest()
new()
Request
FindMessage()
Response
new()
returnResponse()
showResultsPage()
sees

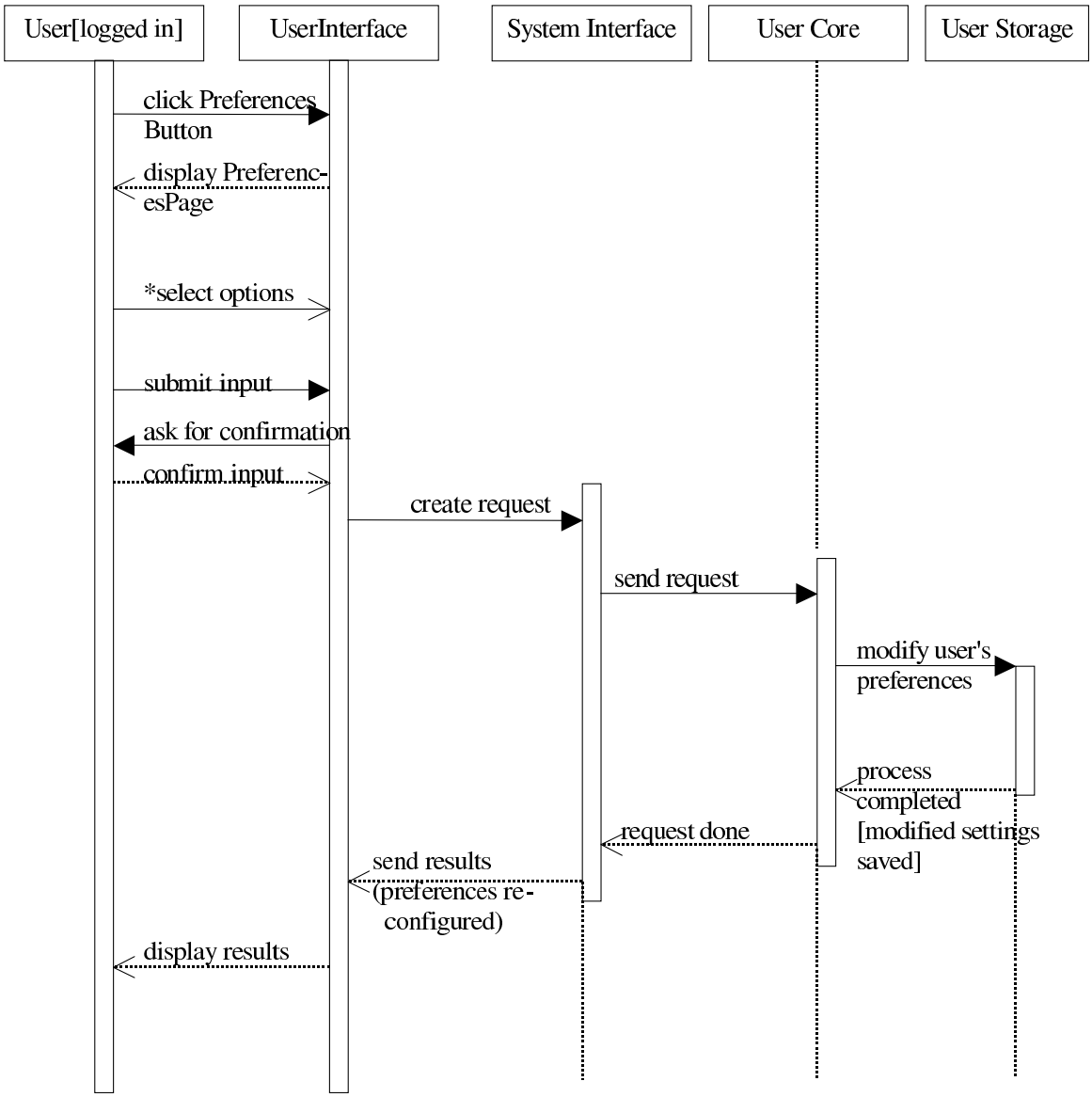
6.1.2 Customer Specific Sequence Diagrams

1. Subscribe to InfoWarrior Services



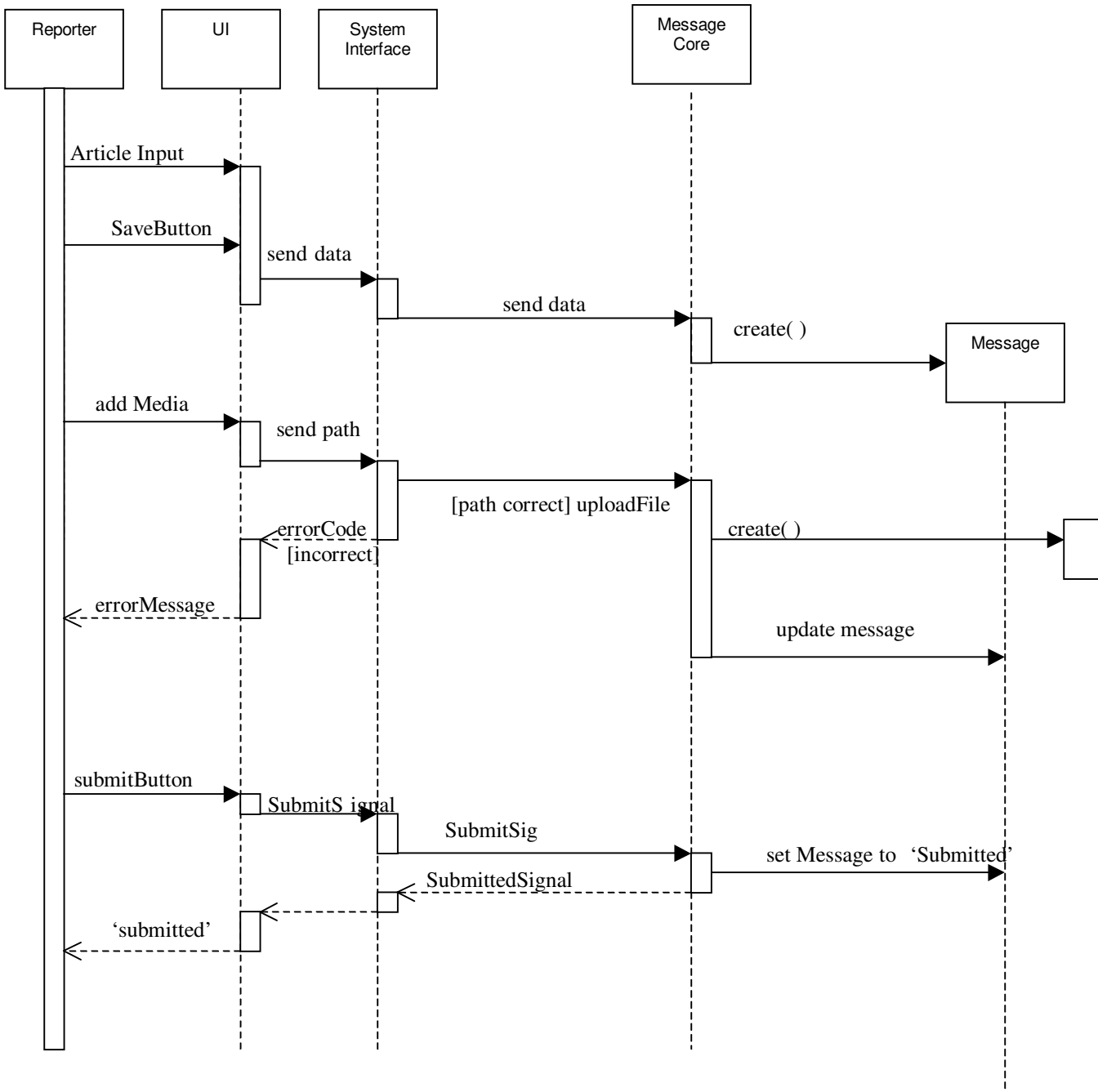
Remarks: The diagram focuses on the flow of data. Thus, it does not specify how the process "add new subscription" is actually implemented by the UserStorage-Unit.
It is assumed that users who want to subscribe for services need to have an account, as this condition seems to be sensible and may provides advantages concerning management of the system.
Of course, an user may be asked to enter additional information repeatedly, if necessary.

2. Edit Preferences



6.1.3 Reporter Specific Sequence Diagrams

1. Edit and Submit/Save Article

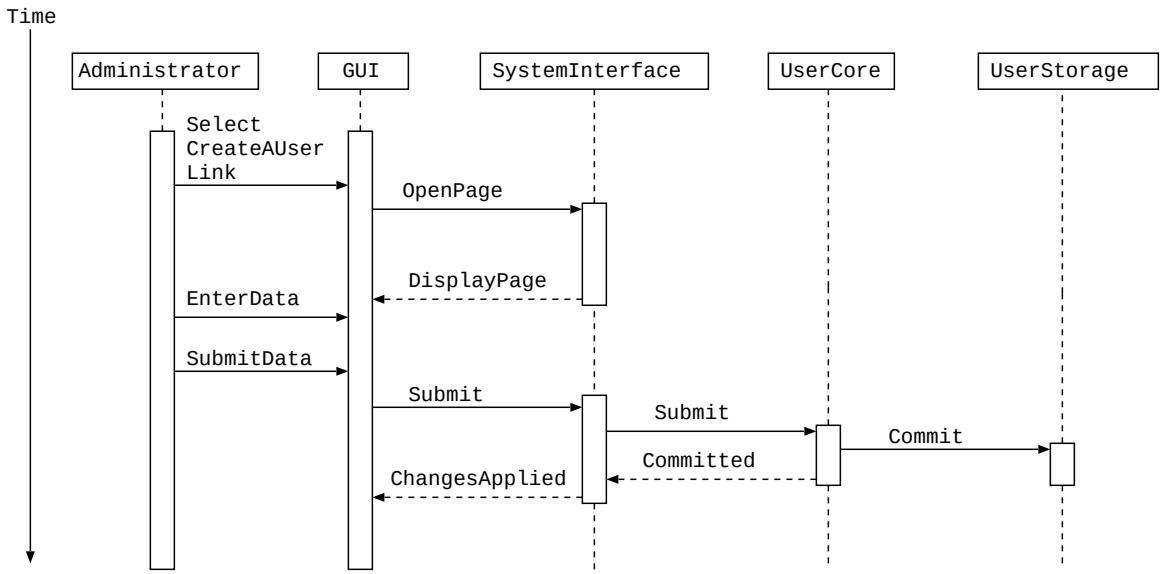

Reporter
UI
System Interface
Message Core
Message
Article Input
SaveButton
send data
send data
create( )
add Media
send path
[path correct] uploadFile
errorCode
[incorrect]
create( )
errorMessage
update message
submitButton
SubmitS ignal
SubmitSig
set Message to 'Submitted'
SubmittedSignal
'submitted'

6.1.4 Editor Specific Sequence Diagrams

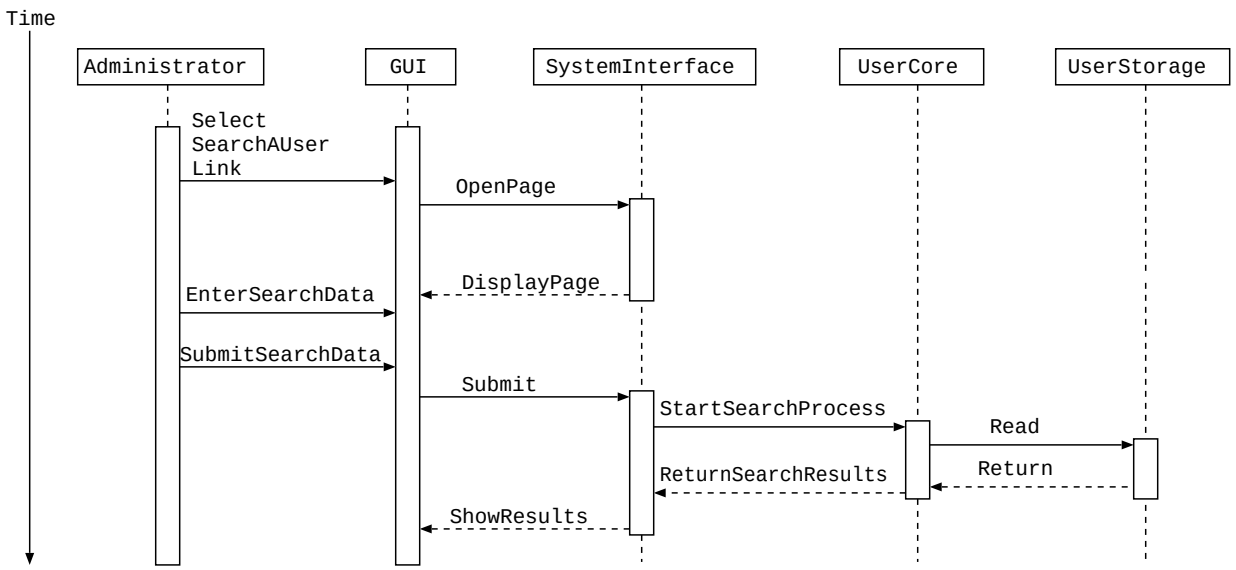
1. Edit and Submit/Save Article (see above)
2. Release Article

6.1.5 Administrator Specific Sequence Diagrams

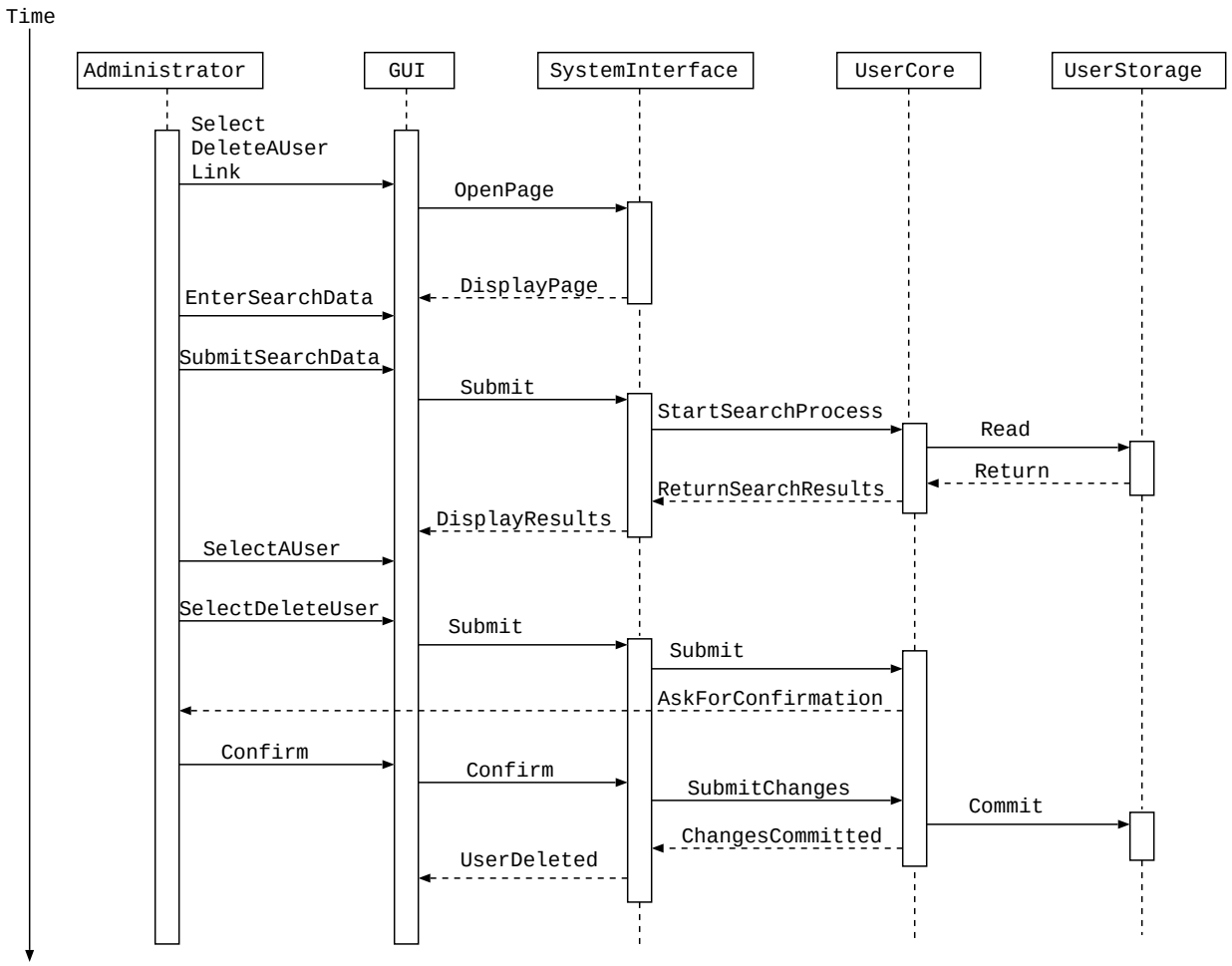
1. Create a User



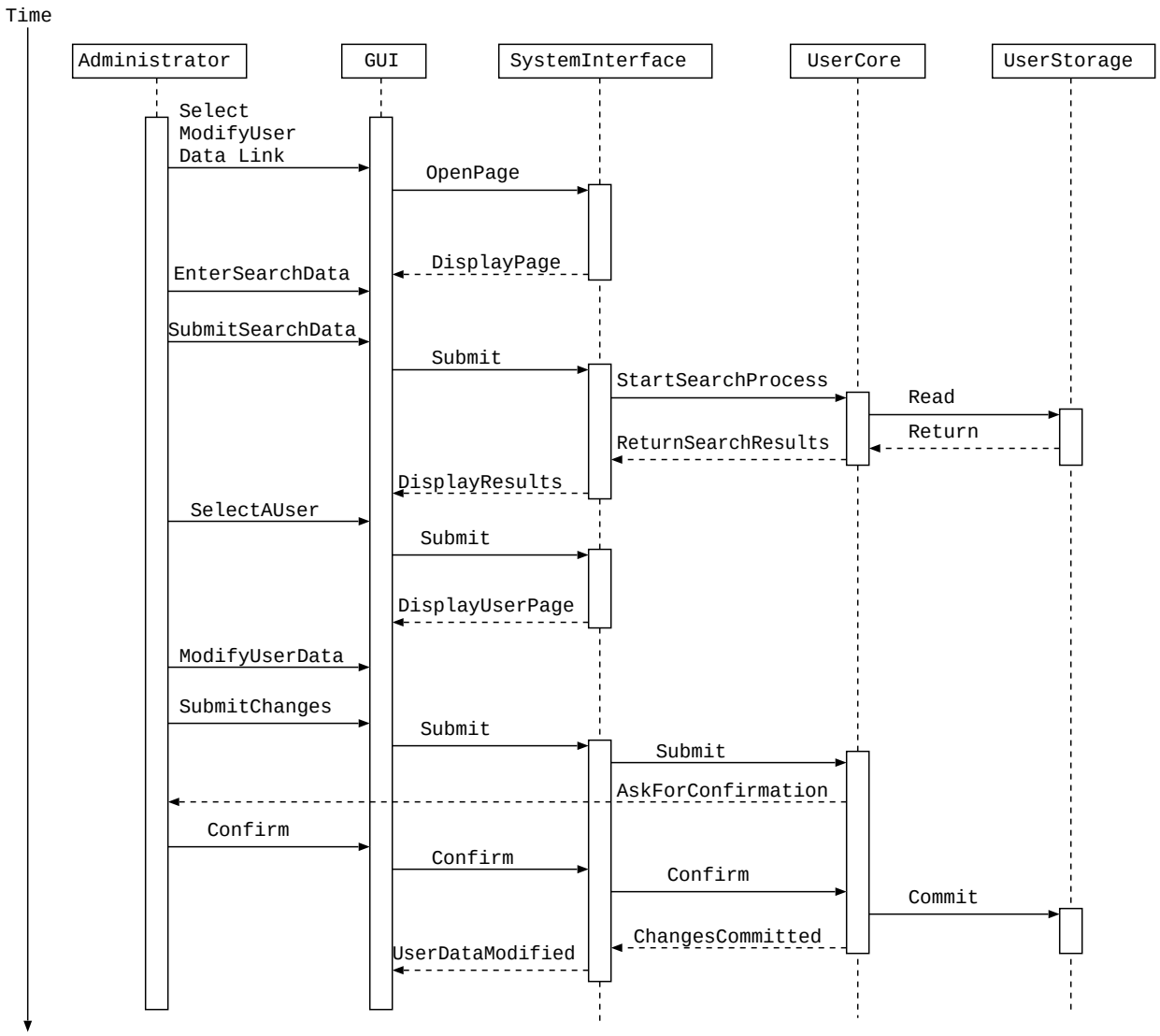
2. Search a User



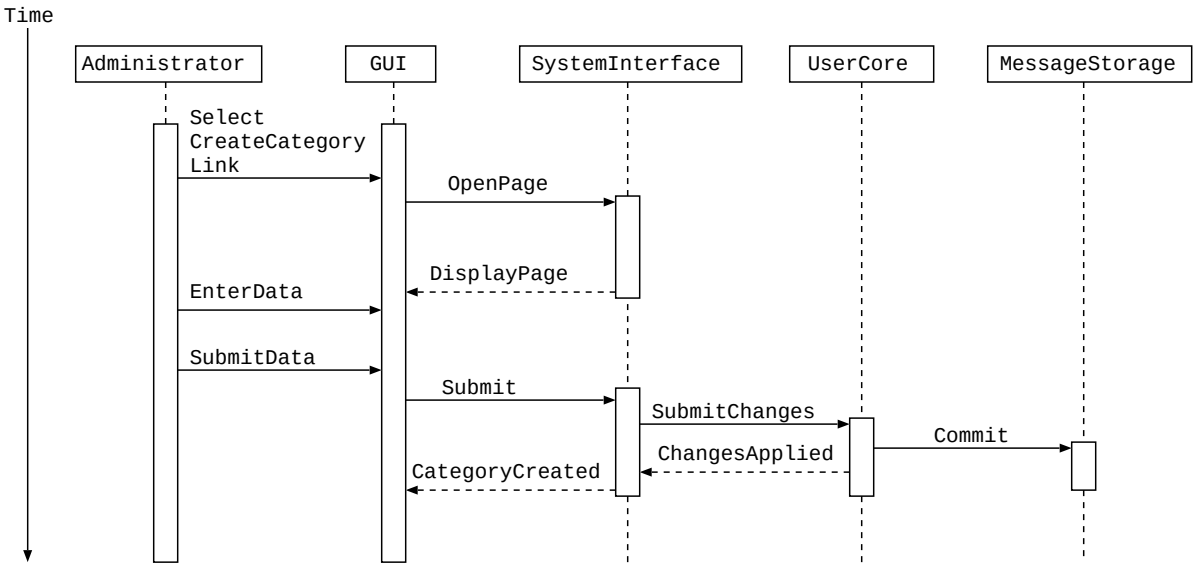
3. Delete a User



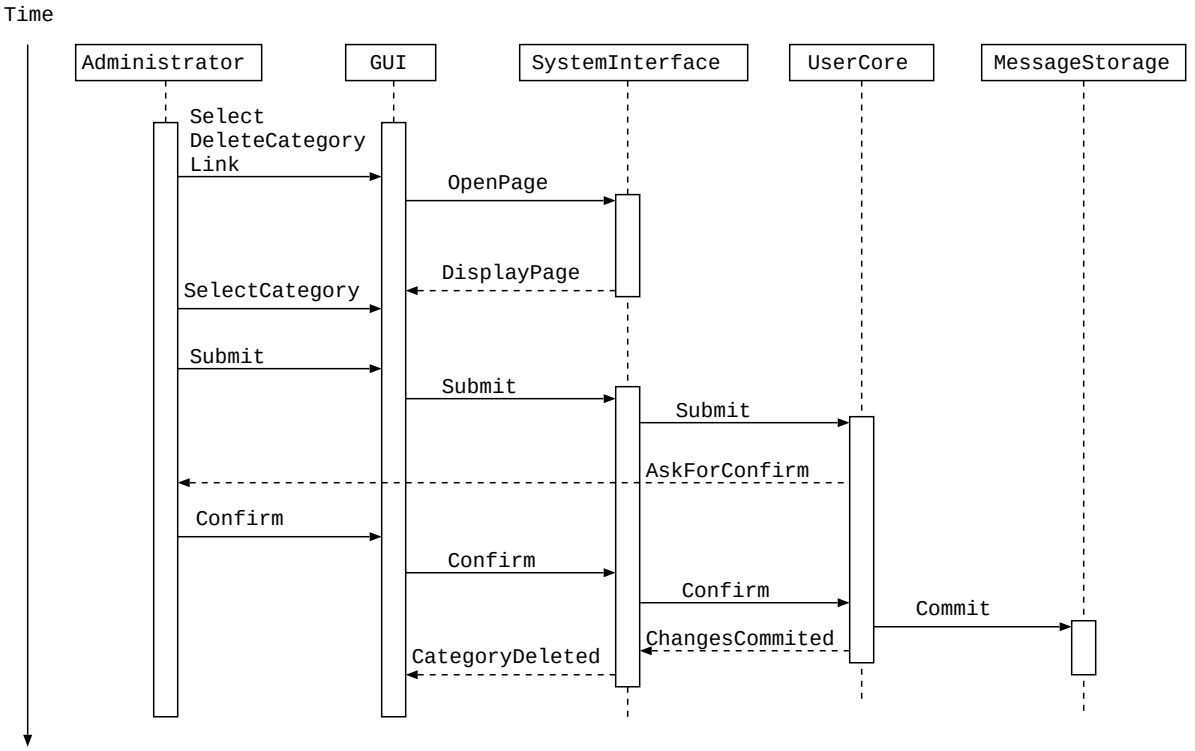
4. Modify User Data



5. Create a Category

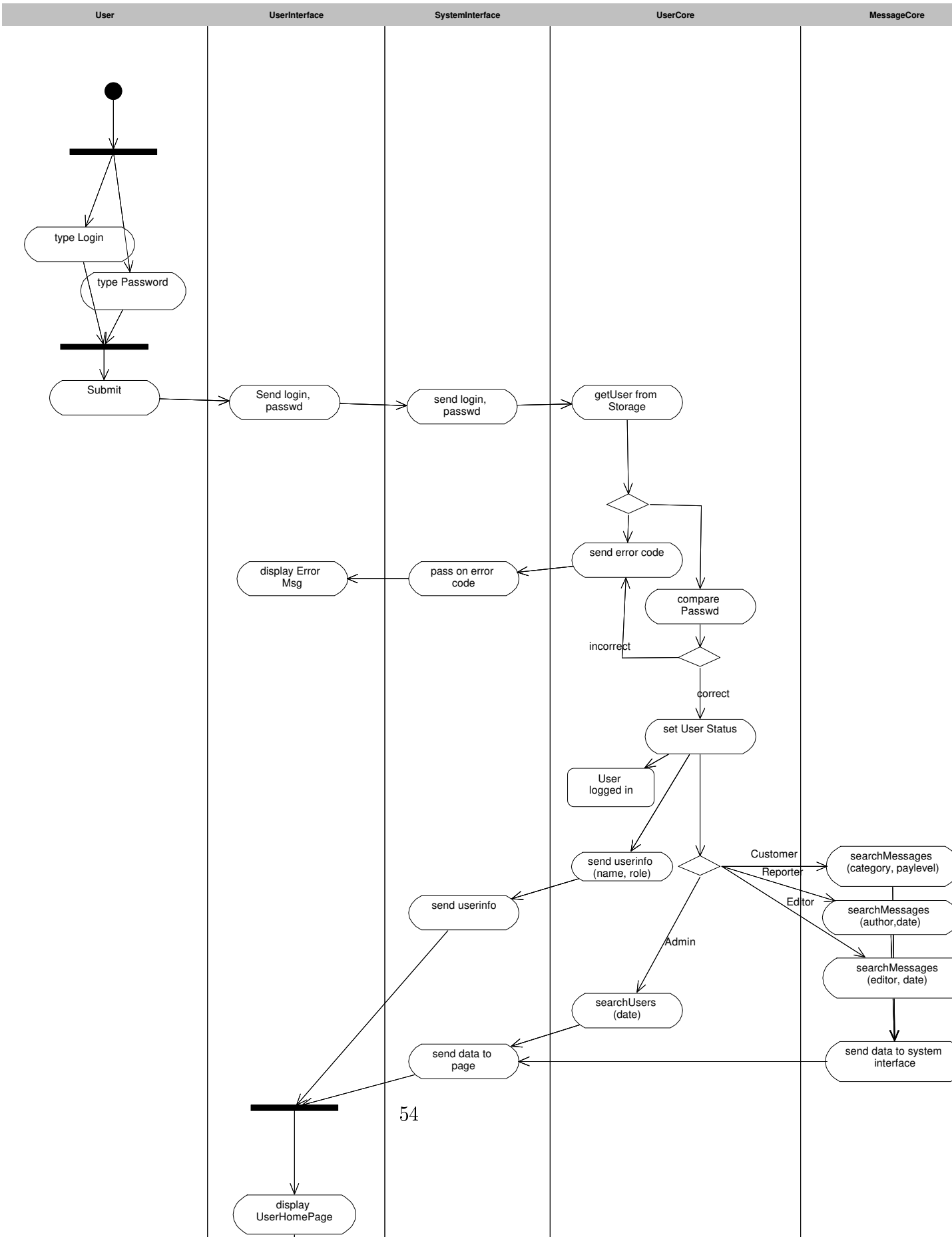


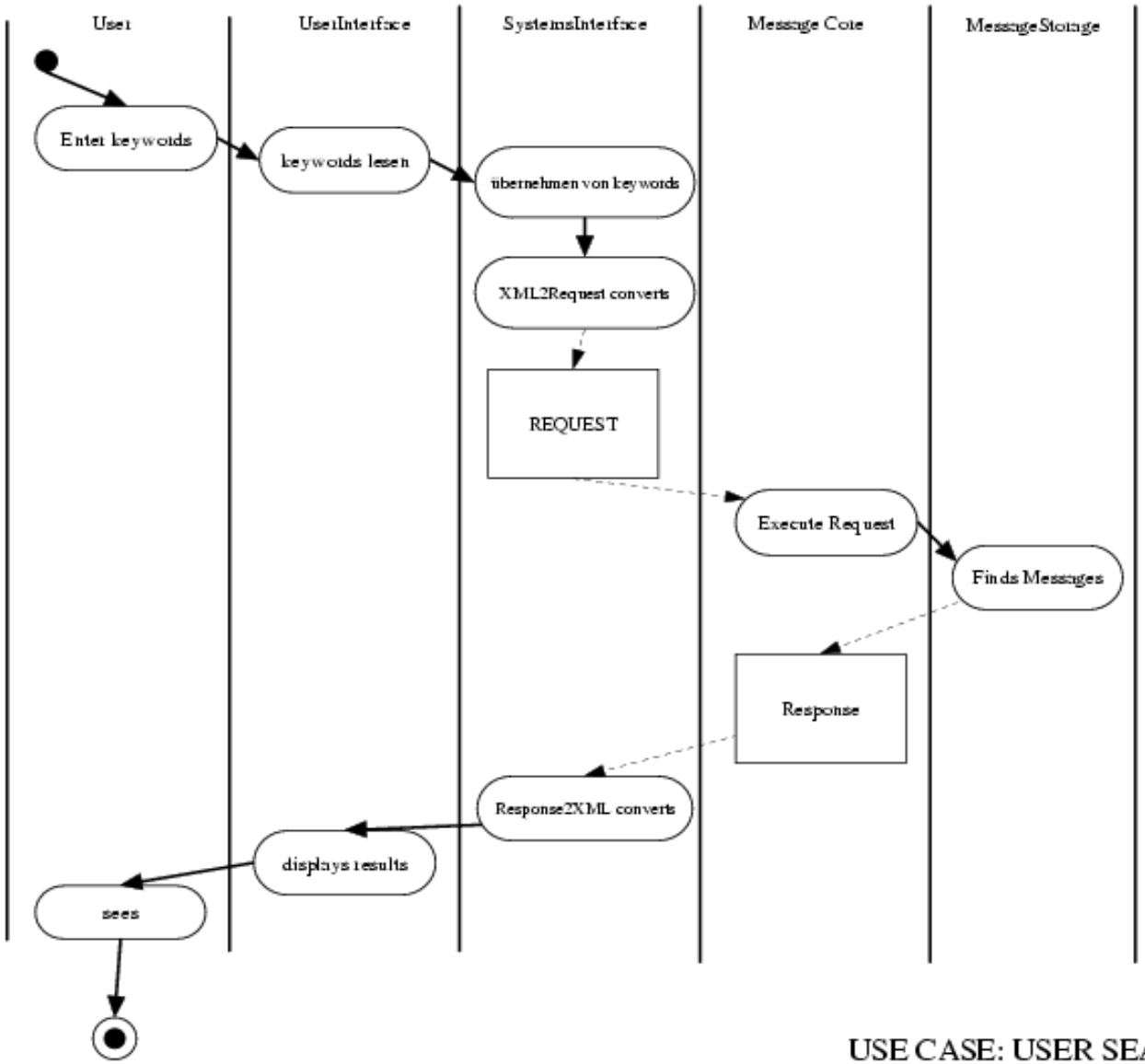
6. Delete a Category



6.2 Activity Diagrams in InfoWarrior

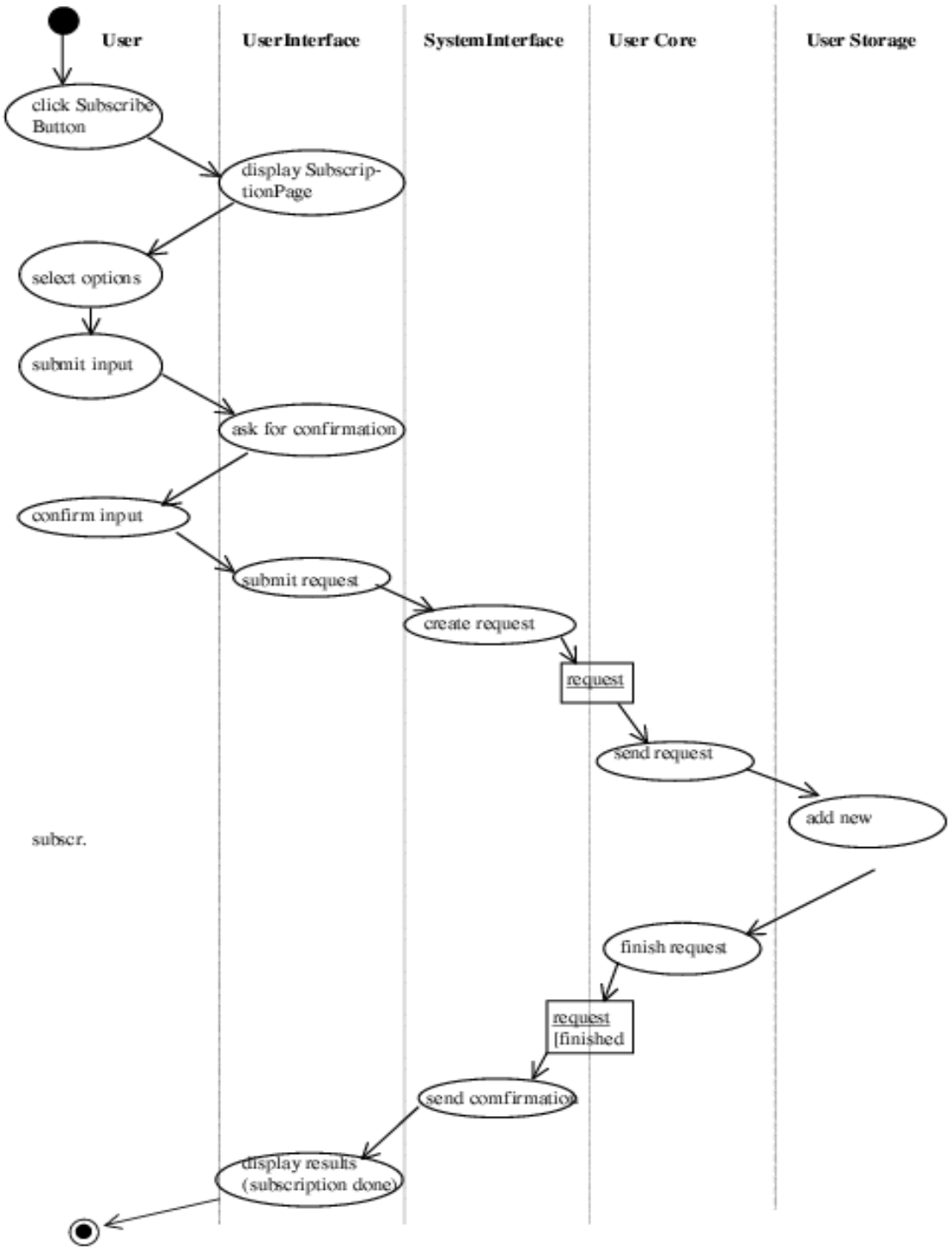
6.2.1 User Specific Activity Diagrams



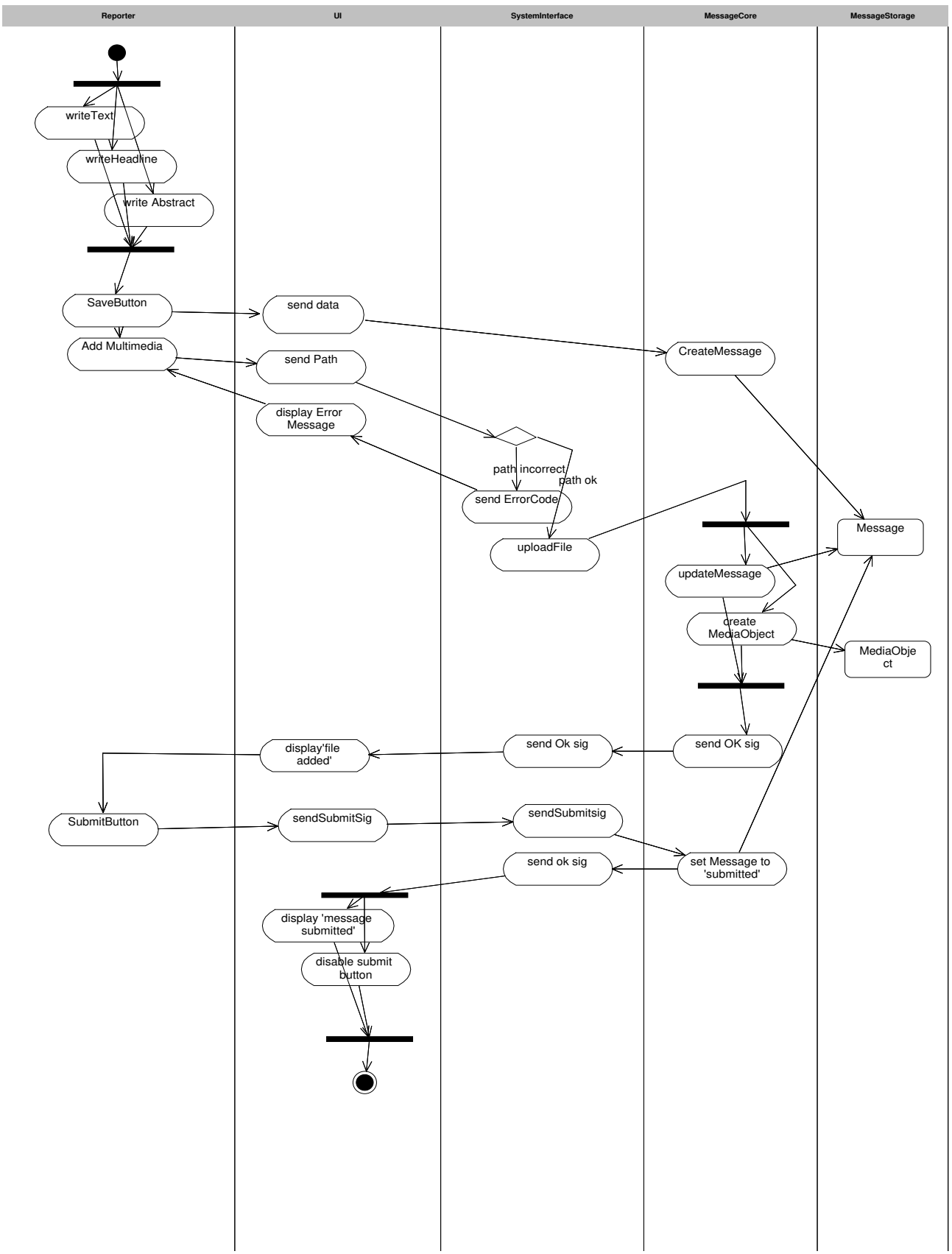


USE CASE: USER SEARCHES NEWS

6.2.2 Customer Specific Activity Diagrams

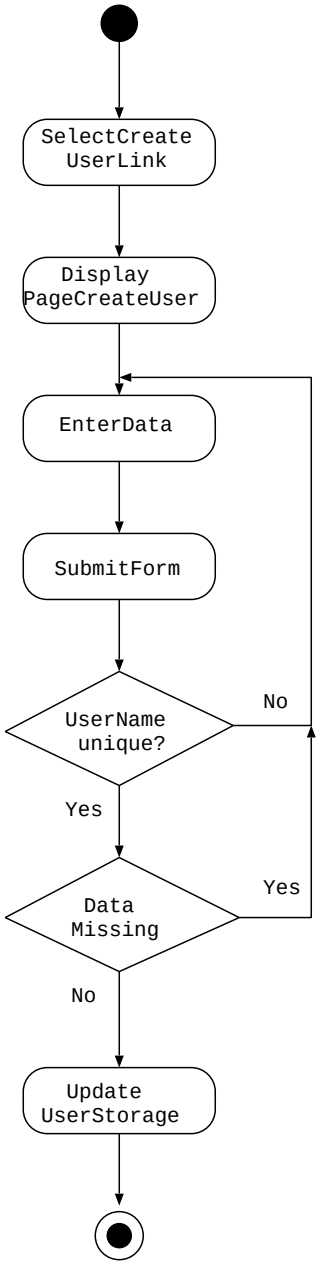

User
UserInterface
SystemInterface
User Core
User Storage
click Subscribe Button
display Subscrip-tionPage
select options
submit input
ask for confirmation
confirm input
submit request
create request
request
send request
add new
subscr.
finish request
request
[finished
send comfirmation
display results
(subscription done)

6.2.3 Reporter Specific Activity Diagrams

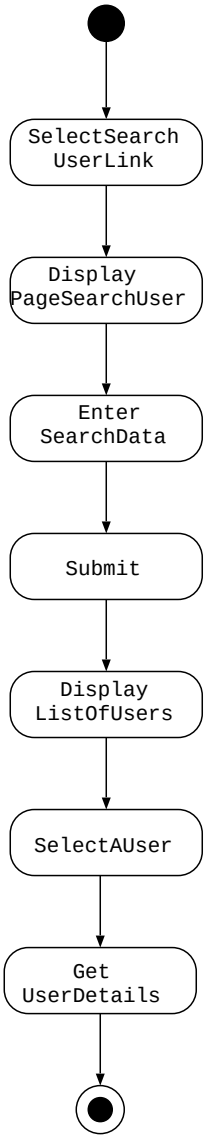


6.2.4 Editor Specific Activity Diagrams

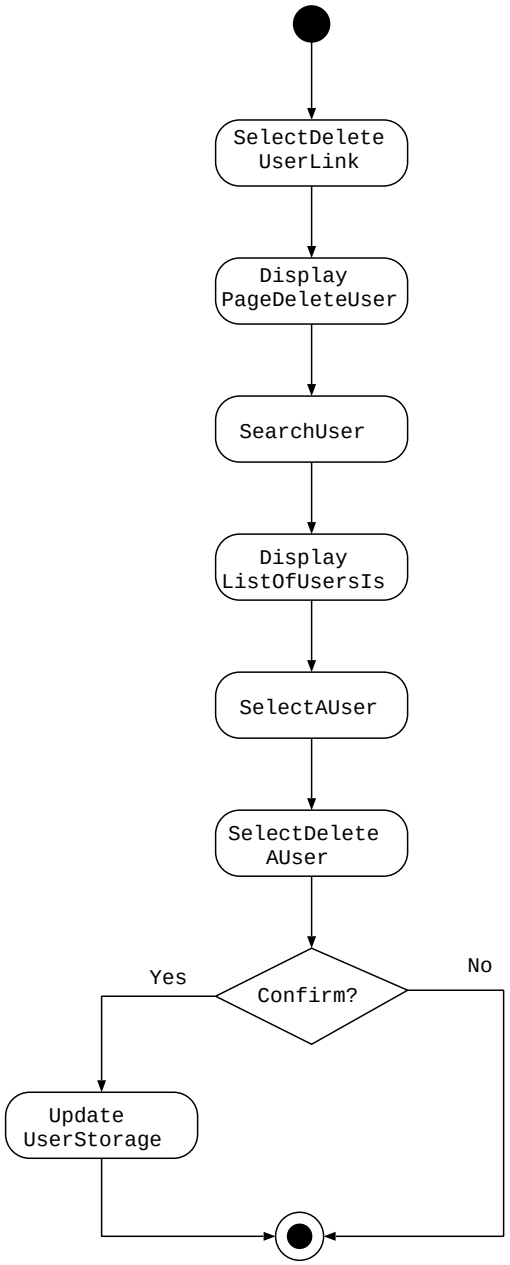
6.2.5 Administrator Specific Activity Diagrams



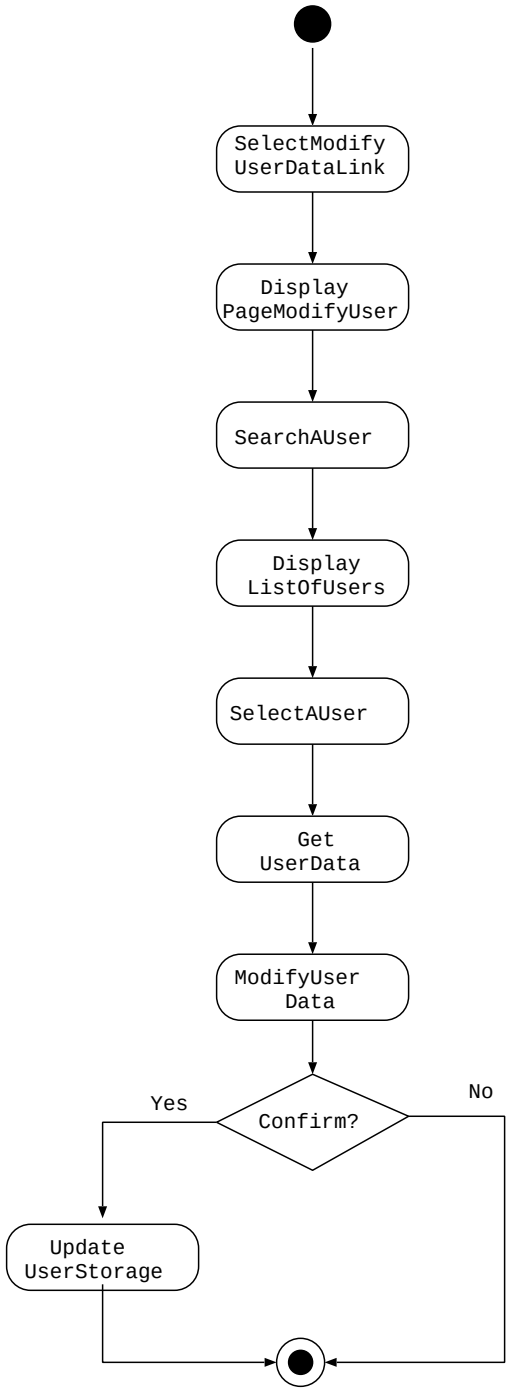
Activity Diagram: Create A User



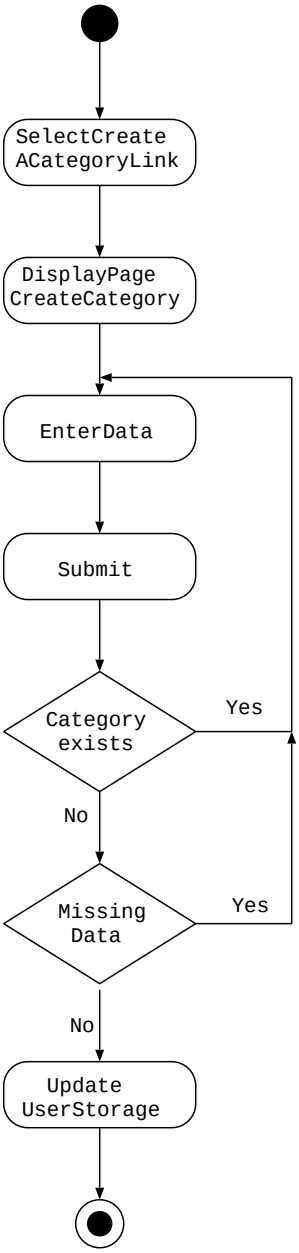
Activity Diagram: Search A User



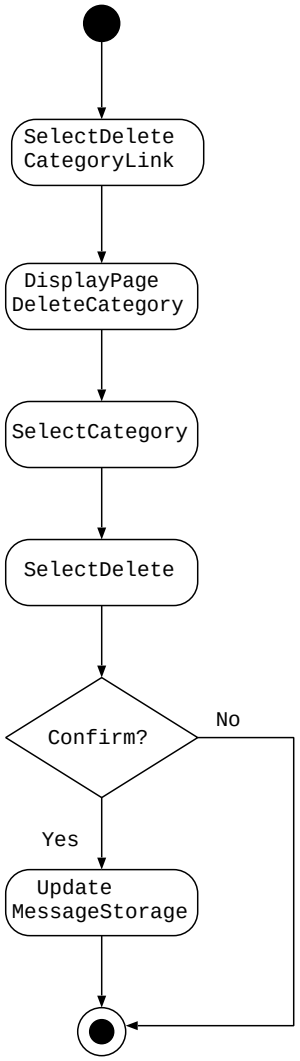
Activity Diagram: Search A User



Activity Diagram: Modify User Data



Activity Diagram: Create A Category



Activity Diagram: Delete Category

Chapter 7

Implementation

7.1 Introduction

The implementation will follow the classes described in the class diagrams very closely. However to allow for greater flexibility, all of these clases will be defined as Interfaces or Abstract Classes. This allows the actual implementation to be exchanged if necessary.

7.2 Implementation Technology

We suggest to use Java as implementation plattform. The following points were important for this decision:

- Plattform independence
- Stability
- Widely used and accepted standard
- Clean, Object Oriented Approach
- Wide range of libraries
- Same technology can be used for backend, web interface and graphical interfaces
- RMI allows distributed computation

The following implementation details are not specific to Java. However, they were written to be easily implmentable in Java.

7.2.1 Libraries/Existing Systems

Once the implementation plattform is decided, a comprehensive search on existing libraries and utilities with usable functionality will be done.

7.3 Organisation

7.3.1 Packages

Implementation packages are the same as the ones used in the class diagrams, however, the may contain sub packages for implementation classes.

7.3.2 Servers

The Core systems (User, Message, System Interface) run as independent systems and communicate through a message passing mechanism.
Accesses to the system from the outside are possible in two different ways. First an RPC (Remote Procedure Call) Service is exported, that can be used to directly make calls to the system interface. The technology actually used for RPC heavily depends on the language and plattform used for the implementation. Examples would be RMI, CORBA or DCOM. This methode should be used by all system that closely cooperate with the InfoWarrior, as it provides high performance and a standardized interface. It will also be used to connect the User Interface with the core system.
For systems that are unable to use the RPC mechanism, there will also be an interface, that supports communication using XML messages sent over a standard network connection. This interface will be used by application that loosely cooperate with the Infowarrior and that use technology, which is not compatible with the used RPC mechanism. The XML messages will have a format, that can represent the Request and Response classes used internally. The XML format allows easier communication with systems using other data formats, because it is a very flexible way to store structured data, that is widely used, throughout the industry. The XML standard also contains ways to transform XML data to other formats.

7.3.3 Classes

Data Container Classes

All classes that are mainly used for data storage have an interface with getters/setters defined for their public attributes and additionally a methode to store them to persistent storage. The implementation class for each container class depends on the storage type used. For the first implementation we plan to implement the classes using a SQL server.
Beside the interface implementation there will also be a factory class, used to create container classes.

Request/Response Classes

The interface for the request/response interface supports basic information on the content (command type) and exports functionality to write/read XML. There are multiple implementation classes for different request/response types to support specific attributes/methodes. Like the data container classes, the implementations are created in a factory class.

Security Classes

The security classes will be implemented similarly to data container classes. They define a common interface that can be used for different implmentations. The basic implementation will use database storage, but other implmentations are possible to integrate with existing security archictectures.

Core system

The classes of the core system (User Core, Message Core, System Interface) run as singletons. Though these classes will implement all methodes of the respective interfaces, the main logic will be capsuled in other classes, that are loaded dynamically and are easily exchangable.
For example: The message core provides a method for searching messages. This methode does not contain the actual search logic, it rather loads a specialized "searcher" class, that contains this functionality. This design makes it easier to exchange and update single functions in the system, if the underlying data structures change or new semantics are desired.

Factories

Factories are singletons. Their only purpose is to create object for a given interface. The do not only load the correct class for an interface, but also initialize the newly created object. This is of special importance for data container classes.
The use of factories allows to exchange interface implementation without touching the source code, but by just editing configuration files. Furthermore it allows the system to easily keep track of all objects created.
It also simplifies data consistency throughout the system, as two requests for the same data entity can return the same object, rather than two distinct objects filled with the same data.
Obviously factories add a performance penalty compared to the use of static object generation. However, one should keep in mind the infrequency of object creations combined with the fact, that the actual creation and initialization of an object takes a lot longer, than the factory overhead (at least an order of magnitude). This should make the performance penalty negligible.

7.3.4 Data Consistency

Except the mechanism described above, where factories return references to the same object for identical data identities, no precautions are taken to keep data consistent throughout the runtime system.
Data consistency is only guaranteed within the storage system. These means a program creating a user object for some user can not be sure, that this users data isn't modified during the objects existence and the object data will not necessarily be updated accordingly.
This choice was made because the system is not depend on absolute consistency. Furthermore we expect concurrent updates to happen rarely if ever. The reasoning is, that a huge

majority of system accesses are reads and competencies for writes are very clear as most objects have one distinct owner.
However, there will be a possibility to prevent accesses to a loaded object. This allows higher levels of the system to achive absolute consistency. Furthermore the factories be able to create a unique copy of an object, rather than a shared reference, if requested.
This design choice makes the system fast for the common case (reads), because it does not have the overhead of creating proxy classes for all data objects.
As all classes are created and initialized in factories, they can ideally be used for data caching. This saves an enormous amount of time compared to reaccessing the underlying data provider every time an object is requested. Cache consistency should be trivial as factories are informed about all objects in the system and also about every object, that is written back to storage. The caches will act as write-through caches to ensure consistency in case of a crash. The big overweight of read accesses justifies write-through performancewise.

7.3.5 Case Study SQL

To clarify design issues and test the choices made, the implementation of the system using an SQL database as storage system will be described.

Class Implementations

- Data Container Classes The data container classes depend heavily on the underlying storage technology as their content must be stored to and restored from the storage system. In the SQL modell all data entities can be identified by the table that stores them and an unique id.

  Each data object should be able to initialize itself from stored data and write modified data back to persistent storage. The database objects used variables to store all their attributes. The actual data retrieval is done by singleton, which handles database connectivity. This class exists as a singleton, it knows about each classes attributes and how to retrieve them. This done either by using configuration files or some other class information (idl, java reflection, ...). This class also handles data storage. All logic implemented in the database classes is a call to the singleton and the getters/setters. Except for classes that support special logic for data manipulation (e.g. collections with add/remove methodes).

- Request/Response Classes Independent from the storage system.

- Security Classes Independent from the storage system, though most security classes can be treated like data containers.

- Core system Mainly independent from the storage system, except the handling of the functionality classes. These classes have to map InfoWarrior calls to semantically identical operations on the database.

  The searcher class for example must be able to convert a search request to a select statement, that returns the desired objects.

- Factories The factories have to be able to create the special database implementation classes and provide them with the information required to initialize themselves (database id).

7.4 Design Patterns

The implementation described above is not modelled strictly following one design pattern. It incorporates ideas and structures from at least 3 design pattern.

- **Factory Pattern** This is clearly the pattern, which had the strongest influence on the implementation. The use of factories to create classes is essential to the way the whole system works.
- **Singleton Pattern** Some of the most important objects of the projects act as singletons, namely the Cores and the Factories. However, this decision was destined by the use of factories mor than other considerations.
- **Facade Pattern** The system interface and the cores both act as facade for other objects. This decision was already made during the object design process. So facades are used, though the facade pattern was not followed in depth.

7.5 Evaluation

A lot of factors determined every single decision made during the process of implementation design. Often there is little hard evidence for or against a decision, nevertheless it has to be made. This section will explain some of the advantages and disadvantages, that can be seen in the design at this moment. Certainly, a final conclusion about pros and cons of the decisions taken will only be possible after the actual implementation process is progressed further.

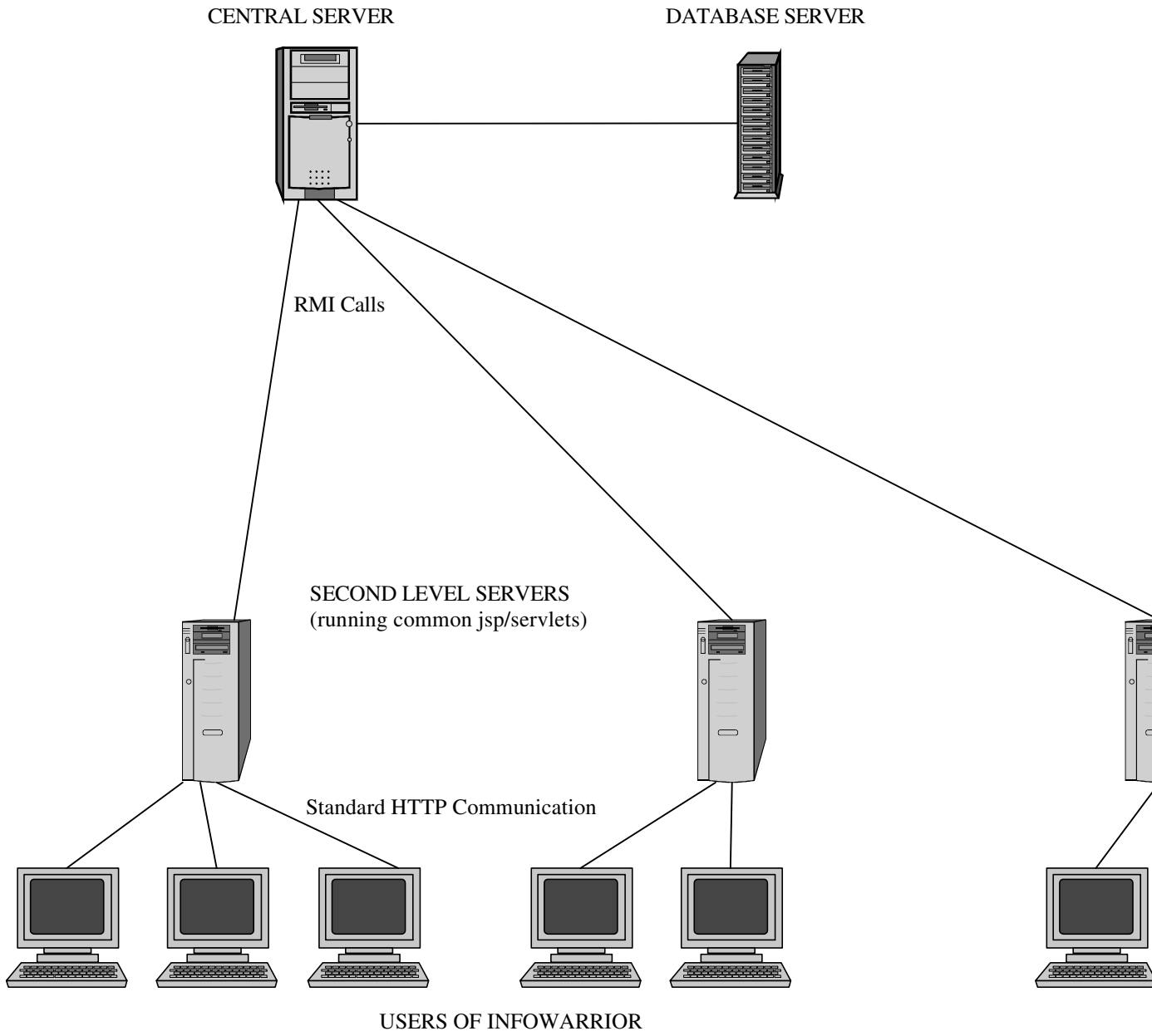
7.5.1 Advantages

- The plattform we decided on (Java) should allow both quick development and a stable, powerful product. The existence of extensive libraries should reduce the amount of code, that has to be written, to a minimum, while leveraging flexibility.
- The separation into different independent servers makes the system stable, scalable and easy to integrate into existing environments.
- The factory approach makes upgrading and maintenance of the server extremely easy.
- RPC and XML server communication make the integration of InfoWarrior with other products rather simple, especially for other Java applications.

7.5.2 Disadvantages

- Decisions were not made with performance as a primary goal in mind. Examples are the use of Java, Factories, RPC etc. This raises the hardware requirements for the system.
- Data consistency was solved in a very lenient way. Though this will be sufficient for this system, it might be problematic in other scenarios with high update rates, especially if updates are done by different users.
- Access to the system is not solved in an ideal way. Providing two interfaces (XML, RPC) increases development time and might lead to confusion/inconsistencies.

7.5.3 Network Topology



Appendix A

Protocols

A.1 25.10.2001

1. Person responsible for protocoll: Huskic Zenina
2. Participating persons: Schweiger Roland, all students visiting Software Engineering PS course in WS 2001/02
3. Topics talked about: introduction, general information about PS (tasks, marks, presentation)
4. New assignments:
   - CRC Session
   - planning
   - role distribution
   - setting the goal
   - open questions

A.2 31.10.2001

1. Person responsible for protocoll: Huskic Zenina
2. Participating persons: Schweiger Roland, Theresa Krichhammer, Peter Gruber, Albert Meixner, Michael Brachtl and Huskic Zenina.
3. Topics talked about: Open questions
4. New assignments:
   - Aufgabe 3 (07.11): Verfeinern Sie die Problembeschreibung mithilfe der ersten CRC Session und der darin entstandenen Protokolle. Planen Sie eine weitere CRC Session um aufgetretene Fragen nochmals mit dem Kundenverteter zu besprechen. In dieser zweiten Session werden Use Cases eine Rolle spielen. Desweiteren sollten Sie sich Gedanken zum Essential User Interface Prototyping

machen. Das Verständins für die Problemdomäne ist bereits vertieft. In den Einzel- und Gruppendokumenten werden bis zum 07. 11 die Protokolle der ersten Session mitaufgenommen (Gruppenprotokoll in Gruppendokument und Einzelprotokolle in die Einzeldokumente). Dokumentieren Sie den Ausgang der ersten Session.

A.2.1 Answers to the open questions

1. What does "einfach = simple" mean, that is what is for customer simple enough?
   Simple means: data collecting, data presentation and meaningfull data input.

2. Who are Erfasser, "e-News" agency employees or we? How many data collectors are there?
   Reporters will prepare the news. The director controlls and issues the news. There are about 5 directors. Each director is responsible for approximately 50 reporters. Each reporter prepares at least one, and at most 5 news per day.

3. What is going to be prapared, that is, what is information content (text, pictures, multimedia)?
   Information content: text, pictures, multimedia...

4. Who are Abrufer, that is who can access the infomation, that is, are there different access levels?
   There are different access levels based on the price model:

   - free information available to the whole world
   - very important time dependent information will be paid

5. What kind of foreign systems?
   Foreign systems are: other pages (e.g. CNN page), SMS,...

6. Are there already other systems existing or not? If yes, does DTD for XML exist or not? That is, how is information delivered?
   There is no predefined DTD for XML.

7. How is information delivered to the end-users(per e-mail?)?
   Information to the end-users will be delivered in two ways:

   - push: via SMS, or via e-mail
   - pull: login

8. Does it mean that this a web-based application , or what?
   Yes, this is a web-based application.

9. What is for customer "wesentlich = important"?
   Important: collect and distribute important information
   Not important: fees accounting, customer accounting, etc.

10. Access time?
    The new information (not older than a month) should be faster available than the old one.

A.3 07.11.2001

1. Person responsible for protocol: Peter Gruber.
2. Participating persons: Roland Schweiger, Theresa Krichhammer, Peter Gruber, Albert Meixner, Michael Brachtl and Zenina Huskic.
3. Topics talked about: Open Questions, CRC Sessison, Use Cases and Essential User Interface Prototyping in InfoWarrior.
4. New assignments (21. 11):
   - In dieser Aufgabe beschreiben Sie die Essential Use Cases und das Essential User Interface. Damit ist das Fundament fr die Entwicklung eines Klassendiagrammes gelegt. Sie werden Ihre Ergebnisse einerseits in Ihren Papers dokumentieren, andererseits auf Overhead Folien eine Prsentation ber diese Themen vorbereiten.

A.3.1 Notices

As a result of this second session, several more details of system requirements and administrative business were discussed and clarified:

- *Payment-Methods:*

  Invoice for payment concerning services that do not come for free will be done on a monthly basis. This should give a possibility to check whether users are still paying or willing to pay in time. Usually, invoices will be sent to users' postal adresses.

- The system should be developed as platform independent. The main aspect should be an easy maintenance of the system.

- *Actors:*

  The following items were agreed upon:

  - *human actors:* editors, reporters, users of the system (clients)
  - *external systems*
  - *archives:* containing news that reach further back in time
  - *storage units:* subdivided into a message storage (containing high priority information) and user data (adaministrating access control etc.)

- *entry page - user interface:*

  The entry page should present at least some accurate and on time information from all "over the world". Informations should refer to certain categories. Access to informations will be granted for different levels of payment. The user interface may also present an option to subscribe for services.

- *Sign-in-Procedures:*

  Sign-in is required for push-methods. Therefore, it will be necessary to extract informations from articles for different purposes. This should enable e.g. an SMS-News-Service, that is a minimum information in a given format.

- *Technical Requirements:*

  Editing of articles is done only once for different services. The input form of articles will thus provide seperate areas for entries such as headline, abstract (up to ... characters), categorization, payment-level of information, ... As a last instance, the editor will supervise all entries.
  Editors may compose news, too. Anyway, the two-level-process of releasement has to be secured, so any time an editor acts as "reporter", his article has to be approved by another editor.

- *Technical Aspects of Administration:*

  New subscriptions to services should be processed automatically. Thus, the system has to meet any requirements necessary. However, anytime an action concerning users of the system is taken, the administrator or people of the administration office should be informed personally. Frequent actions may be summarized as reports. Persons, however, should not be required to take actions in order to complete a standardized process.

In this second session, a first concept of User-Interface was presented and approved in general.

A.4 14.11.2001

1. Person responsible for protocol: Huskic Zenina

2. Participating persons: Roland Schweiger, Theresa Krichhammer, Peter Gruber, Albert Meixner, Michael Brachtl and Zenina Huskic.

3. Group Document presented by: Peter Gruber.

4. Topics talked about: Use Cases and Essential User Interface Prototyping in InfoWarrior.

5. New assignments (21. 11):

   - In dieser Aufgabe erstellen Sie ein Analysis Klassendiagrammes. Mit dem Essential User Interface, den CRC Cards und den Use Cases haben wir dafür das Fundament gelegt. Sie werden Ihre Ergebnisse einerseits in Ihren Papers dokumentieren, andererseits auf Overhead Folien eine Präsentation über diese Themen vorbereiten.

6. Project Leader (14. 11 - 21. 11) : Michael Brachtl.

A.4.1 Notices

- Reporter has to have a possibility to interrupt submitting of news ("submitted news" back to "not submitted news".)
- What kind of search will be available?
  There are two possibilities: keyword search or full text search.
  Team has to discuss this point.
- There is no deleting of data from database.
- We have to take care of naming convention (e.g., click vs. select).

A.5 21.11.2001

1. Person responsible for protocol: Zenina Huskic.
2. Participating persons: Roland Schweiger, Theresa Krichhammer, Peter Gruber, Albert Meixner, Michael Brachtl and Zenina Huskic.
3. Group Document presented by: Michael Brachtl.
4. Topics talked about: Class Diagram in InfoWarrior.
5. New assignments (05. 12):
   - In dieser Aufgabe erstellen Sie Prototypen für User Interfaces. Ihr Essential User Interface mit Use Cases und dem ausgearbeiteten Klassendiagramm sollte bzgl. Anwendbarkeit geprüft werden für je einen HTML und einen WML UI Prototypen. Aus diesen Prototypen werden Sie Rückschlüsse auf Essential User Interface Flow Diagrams ziehen und weitere Anforderungen an das System entdecken. Sie werden Ihre Ergebnisse einerseits in Ihren Papers dokumentieren, andererseits mithilfe von Programmen eine Präsentation über diese Themen vorbereiten.
6. Project Leader (21. 11 - 05. 12) : Albert Meixner.

A.5.1 Notices

- Our class diagram is to detailed, and the goal of a class diagram is to show basic classes and relationships. The implementation details are not important in this phase (Problem Domain Modeling.)
- UML Notation should be used.

A.6 05.12.2001

1. Person responsible for protocol: Peter Gruber.
2. Participating persons: Roland Schweiger, Theresa Krichhammer, Peter Gruber, Albert Meixner, Michael Brachtl and Zenina Huskic.
3. Topics talked about: HTML and WML User Interface Prototypes in InfoWarrior.
4. New assignments (12. 12):
   - Überarbeiten Sie die Aufgabe 6 lt. Besprechung bei der Präsentation. In dieser Aufgabe erstellen Sie Sequenzdiagramme und Aktivitätsdiagramme um das dynamische Verhalten des Systems und die Interaktion der Objekte/Klassen genauer zu spezifizieren. Halten Sie sich an das Skriptum im Sekretariat und vergessen Sie nicht, dass wir uns immer noch in der Analysephase befinden. Sie werden Ihre Ergebnisse einerseits in Ihren Papers dokumentieren, andererseits auf Overhead Folien eine Präsentation über diese Themen vorbereiten.
5. Project Leader (05. 12 - 12. 12): Zenina Huskic.

A.6.1 Notices

In this session, prototypes for HTML-based and WML-based User-Interfaces were presented. During development, one important guideline was that all pages should have a consistent look and feel (logo, title bar). The first page (Start Page) should present the most important news and a log-in option.
During presentation and discussion of the HTML-based version, some more demands to the system emerged:

- *Search-Link:* A search can be restricted to categories. In addition to the basic search page, more sophisticated pages may be developed, too. Repeated search might be optimized by restricting the search to previous search results.
- *Subscriptions:* Payment levels can be selected seperately for each category. Therefore, we will have to discuss one more usecase: reconfiguring information about users.
- *Articles:* A reporter has the possibility to clear, save and submit articles (email them to editor). The editor will finally decide about payment level and categorization.
  Articles that are submitted to an editor should be linked to the names of reporters rather than their identification numbers.
  As related articles should be linked together, additional meta-information on articles is required.

Discussing the WML-based version, the following important aspects were mentioned:
Different layouts of these two prototypes demand different flow charts. Thus, one has to look for elegant ways to handle different technologies. As the logic background is the same, however, it might be enough only to change the view and keep the rest the same.

A.7 12.12.2001

1. Person responsible for protocol: Zenina Huskic.
2. Participating persons: Roland Schweiger, Theresa Krichhammer, Peter Gruber, Albert Meixner, Michael Brachtl and Zenina Huskic.
3. Group Document presented by: Zenina Huskic.
4. Topics talked about: Sequence and Activity Diagrams in InfoWarrior.
5. New assignments (12. 12):

   Hiermit leiten wir die nächste Phase ein, nämlich die Design Phase. War unser Ziel in der Analyse Phase zu verstehen WAS das Problem ist, werden wir nun bestimmen WIE das System zu bauen ist. Das Design Modell entsteht aus dem Analyse Modell mit Informationen aus dem Entwicklungskontext, Design Konzepten und Anwendungen von Design Pattern. Entwickeln Sie in dieser Aufgabe ein Design Klassenmodell, machen Sie sich Gedanken über mögliche Design Patterns und die geplante Architektur für Ihr System. Sie werden Ihre Ergebnisse einerseits in Ihren Papers dokumentieren, andererseits auf Overhead Folien eine Präsentation über diese Themen vorbereiten.
6. Project Leader (12. 12 - 09. 01): Theresa Krichhammer.

A.7.1 Notices

- Sequence Diagrams are used to modell the exchanging of messages between associated objects, and therefore they should not contain activities, which are specific to activity diagrams (e.g. enter data.)