December 18, 2001

Seminar Software Engineering

ImmoWarrior - Personal Paper

Report

Ruhs Gunnar

gunnarruhs@aon.at

Academic Supervisor Mag. DI 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

Contents

Chapter 1

Introduction

1.1 About the Pro-Seminar Software Engineering

In this pro-seminar we deal with the cycles of software-development in the face of object-oriented methods. The goal is to get practise in using UML, object-oriented (oo) analysis and design, oo testing.
Remark: If i refer to users of ImmoWarrior with *he* it should be read as *he or she*.

1.2 The Project

There are two different given project. My group members and me decided to take the project called **ImmoWarrior**:

A company that auctions real estates wants us to develop a software-solution for their process of collecting and representing information. Potential users of this system are: the ones, who gather and enter the information and the ones, who want to read it. Those readers can either be persons or other systems. This leads to two different scenarios:

- Immowarrior sends the information to an other computer system using XML
- Immowarrior displays the information for a person - what requires us to design an appropriate user-interface.

Specially required by the client is, that the software is to be easily maintainable and that no one has to install client-software to be able to use Immowarrior.

In the face of our lack of experience in XML we decided to take the second choice - presentation of information in a human-readable form.

1.3 The Tasks

- **Task 1** (from 28.09 to 25.10) We should become acquainted with the supporting documents available in the secretariat.
- **Task 2** (from 25.10 to 30.10) We had to plan a CRC-Session with the client, represented by the seminar-leader. As the project was not very clear so far, we had to imagine what the client might want, prepare questions and some basic CRC-cards.
- **Task 3** (from 30.10 to 07.11) In this task we had to refine the problem definition and prepare another CRC-session. Especially Use-Cases should come to the fore. Furthermore we had to tax our brains with essential user-interface prototyping (but as a lession lasts only 30 minutes, we did not get so far)
- **Task 4** (from 07.11 to 14.11) The essential use cases and interfaces had to be worked out in detail. We also had to prepare a short presentation on them.
- **Task 5** (from 14.11 to 21.11) In this task we had to create class-diagrams using the results of the previous lessions. There was also another presentation to be prepared.
- **Task 6** (from 21.11 to 05.12) Prototypes of the user interfaces had to be made using HTML and WML. The goal of this task was to recover some new system requirements we did not find so far.
- **Task 7** (from 05.12 to 12.12) For the next lession we had to revise our UI-Prototypes particulary with regard to the last lession's results. We also had to specify more precisely the dynamic behavior of the system by using sequence- and activity-diagrams.
- **Task 8** (from 12.12 to 19.12) The personal paper (the one you are just reading) had to be completed and sent to DI Schwaiger Roland until Wednesday, December 19., 2001 - 12:00.

Chapter 2

The Workflow

2.1 Lession 1

2.1.1 Preparing for the 1^{st} Lession - Task 1

To be prepared for the first lession we had to get a copy of Mag. Dipl.-Ing. Schwaiger Roland's materials on software engineering available in the secretary. Reading through these documents was not very easy at the beginning because thinking in an object oriented manner was nothing we did so far. But later we recognized the parallels to the conceptual model of database design and from then on the whole thing became clearer.

2.1.2 The 1^{st} Meeting on 25.10.2001

In the first meeting with our academic supervisors we were split up into three main groups and further sub-groups. Mag. Dipl.-Ing. Schwaiger Roland (our main-group leader) then illustrated us the goals of the pro-seminar.

2.2 Lession 2

2.2.1 Preparing for the 2^{nd} Lession - Task 2

So what may be the procedure to prepare a crc-card-session? Fortunately there are many homepages on this topic in the internet. Planning the CRC-Session for the meeting with the client was not easy because the whole project's parameters were not clearly defined so far, therefore we had to imagine what the client might want. So we created an empty CRC-Card and copied it several times.
Later on we came through the following steps:

- The first step is to identify the classes in the problem domain. We have read that using the problem statement (find all of the nouns and verbs in the problem statement) is a good possibility to find out what classes are in the system. The verbs represent the responsibilities.
- We used this information for the basis of a brainstorming session to identify all the class we might need.
- As we got some labeled cards each person of our group got one of them and was responsible class.
- Then we acted out some use-cases (at that time we did not know that this is called "use case").

The result were the following cards:

Classname: **System**	
Responsibilities	Collaborators
interact with storage system forwarding information	Storage User

Classname: **USER**	
Responsibilities	Collaborators
read information write information	Real Estate System

Classname: **Storage**	
Responsibilities	Collaborators
store information read information	Real Estate System

Classname: **Real Estate**	
Responsibilities	Collaborators
keep information on re. read information	

2.2.2 The 2^{nd} Meeting on 31.10.2001

In this meeting (in fact the first real meeting) we put many questions on the project to get the whole problem definition a bit clearer. So some additional facts ImmoWarrior should can, are:

- Get the user input and present information through an HTML and WAP user-interface.
- Search for real estates
- There should also be the possibility for the offer-placing person to upload some multimedia-files as short audio and video sequences.
- Information on new placed real estates can also be forwarded to mail or handy (sms)
- Some free offers should be available through the HTML user interface fur unregistered users.
- Users should have different privileges:
  - Users who may just read the free offers (guest,unregistered users)
  - Users who may browse, search the placed offers (user)
  - Users who may place new offers (editors)
  - Users who may manage accounts, real estates,... (admin)
- There is not the possibility to buy a real estate through this system, ist is just for information.
- If an object is sold (ImmoWarrior should have access to other databases), the entry for the real estate shold be removed automatically.

We also found out that there are many more CRC-Cards than we had so far. With the help of DI Schwaiger R. we created the following additional cards:
The Card "User" was split up into Users with different permissions:

Classname: **Admin**	
Responsibilities	Collaborators
access user interface view free offers log in configurate account create/modify account search for real estate view real estate info place/edit/delete real estate log off	user interface

Classname: **EditorAndUser**	
Responsibilities	Collaborators
access user interface view free offers log in configurate account search for real estate view real estate info place/edit/delete real estate log off	user interface

Classname: **Editor**	
Responsibilities	Collaborators
access user interface view free offers log in configurate account place/edit/delete real estate log off	user interface

Classname: **User**	
Responsibilities	Collaborators
access user interface view free offers log in configurate account search for real estate view real estate info log off	user interface

Classname: **Guest**	
Responsibilities	Collaborators
access user interface view free offers request new read account	user interface

Furthermore these cards were added too:

Classname: **User Interface**	
Responsibilities	Collaborators
read input forward data show info start session	guest user editor editoranduser admin session

Classname: **Session**	
Responsibilities	Collaborators
create/modify/delete account initiate sending of account info forwards data to user interface initiate search close session access info storage request account info	user interface account send account info search for info info storage

Classname: **Account**	
Responsibilities	Collaborators
access info storage modify account info	session info storage

Classname: **Information Storage**	
Responsibilities	Collaborators
store account info store real estate info forward data	session account search for info

Classname: **Search For Info**	
Responsibilities	Collaborators
perform search forward search results	info storage session

2.3 Lesson 3

2.3.1 Preparing for the 3^{rd} Lesson - Task 3

This time we should prepare a short presentation on the essential use-cases and user interfaces of ImmoWarrior.

Use Cases

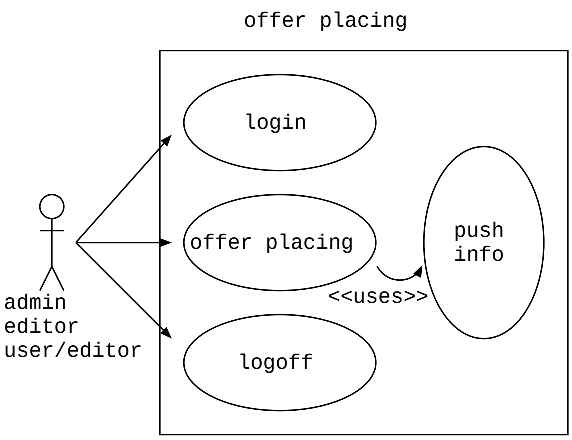
As **actors** play an important role in working with use cases,
here are our actors:

- guest (gu)
- admin (ad)
- editor (ed)
- user (us)
- user-editor (ue)

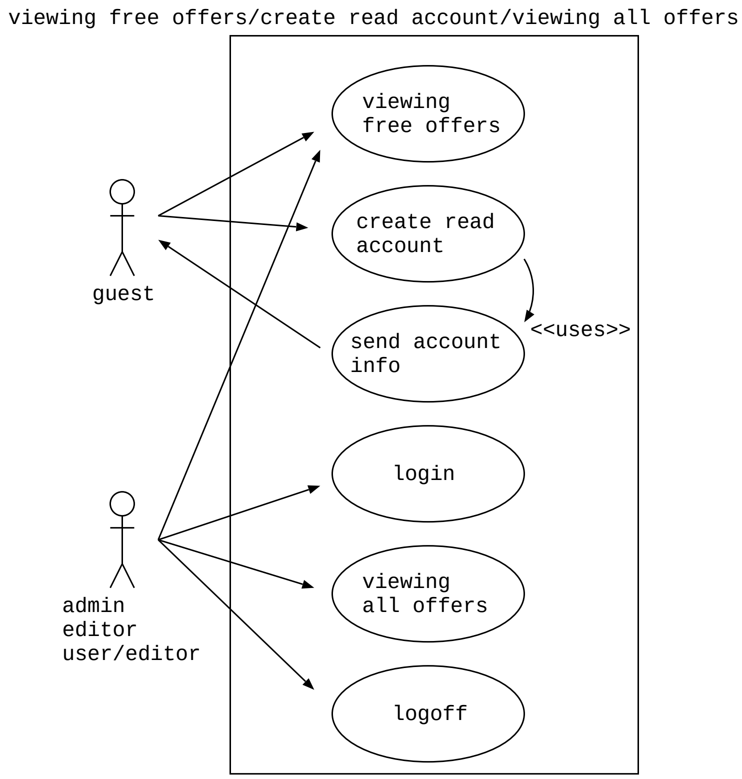
The differences between that different actors are the privileges they have: The **guest** may view the free offers and can request a read account. The **user** may view all offers. The **editor** may place real estate offers and has also the guest's privileges. The **user-editor** has the privileges of the user and the editor together. And the **admin** of course has all available privileges including the write-account privilege. But these actors were revised later on as there exist some kind of inheritance between them - but more on this later.

Now, we came across the following use-cases:

- **offer placing:**
  - The ad/ed/ue logs in. If successfull, he can do the next step:
  - The actor places his offer, which means he enters the data of a real estate, uploads files,...
  - Now if the user has submit the data the system stores it in the storage-system and pushes an info about the new entered real estate to all the users who have subscribed to get that info via SMS or mail.
  - Afterwards the actor logs off.

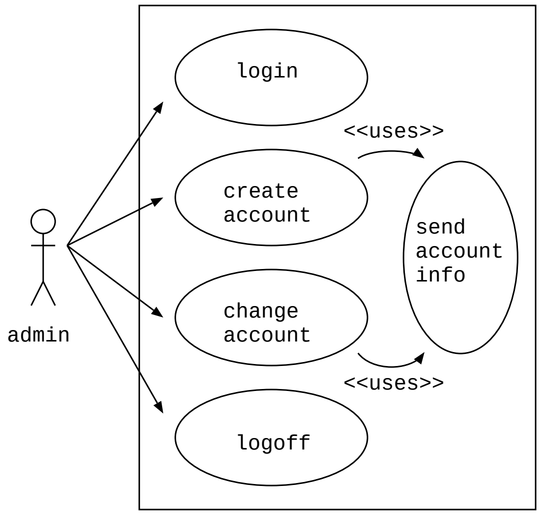


- **view free offers:**
  - The actor (anyone: gu/ad/ed/us/ue) can view the free offers (He does not need to log in).
- **create read account:**
  - The actor (gu) can request a read account by submitting his personal data including credit card information to the system.
  - After the system validates the submitted data it sends the account information back to the actor by mail. The actor's status changes from *guest* to *user*.
- **view all offers:**
  - The actor (ad/us/ue) logs in.
  - Afterwards he can view all offers (maybe by browsing through or searching for them).
  - When finished the actor logs off.



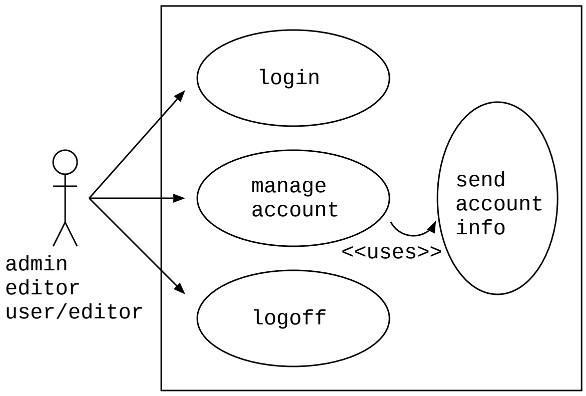
- **create account:**
  - The actor (only the ad) logs in.
  - Then he creates a new account by submitting the new user's personal information.
  - When done the system sends the account information to the new user.
  - The actor logs off.

- **modify account:**
  - The actor (only the ad) logs in.
  - Then he can edit the account information/privileges of any user.
  - When finished the actor logs off.



- **manage account:**
  - The actor (ad/ed/us/ue) logs in.
  - Afterwards he can modify some of his account informations (address, forwarding new real estate info to SMS/mail, cancel account).
  - When done the actor logs off.

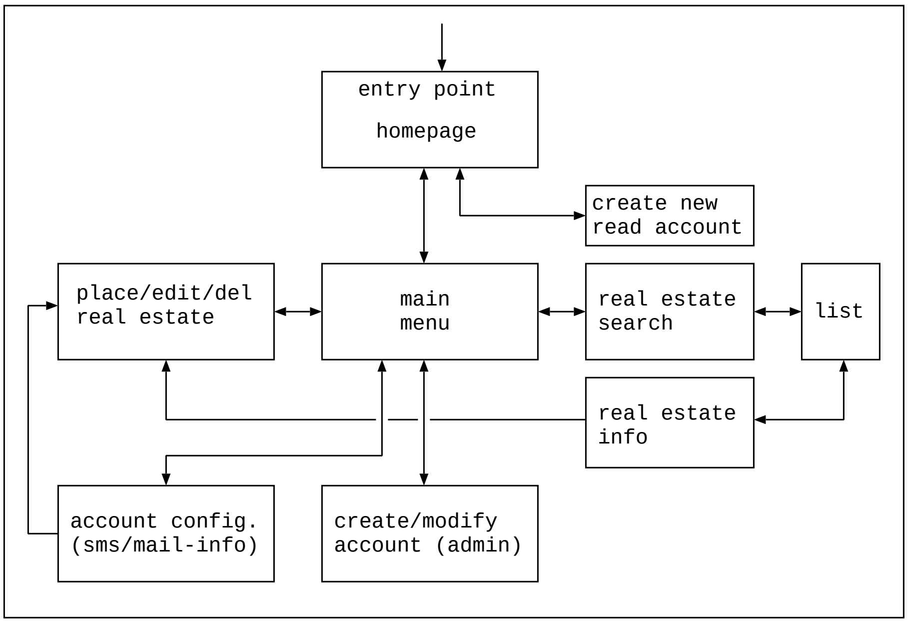
manage account



Essential User Interface - Overview

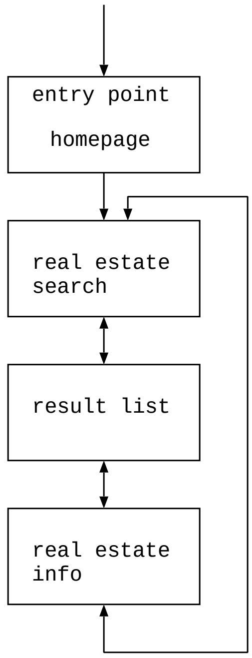
For the next meeting we also prepared the essential user interface of ImmoWarrior. As we expected the use cases to take almost the whole half hour of the meeting we designed just an overview this time.

- The HTML Interface:
  - The entry point of the user interface is of course the homepage (startpage).
  - From the startpage the user can view the free offers, create a read account or log in to get to the main menu.
  - Now at the main menu the user has the choice to seach for real estates, create/modify account informations and place new real estates.



- The WAP Interface:
  - The entry point of the WAP user interface is a login - screen.
  - After logging in the user gets to a search form, where he can enter some search criteria.
  - When he submits them he gets to the results form.
  - Now the user can choose one of the results and view the details of that selected real estate.

(see the draft on the next page)



2.3.2 The 3^{rd} Meeting on 07.11.2001

In the meeting we presented the results we worked out. As expected we discussed the use cases but we did not get to the essential user interface. So DI Schwaiger R. assigned us to work out the essential user interfaces in detail for the next meeting.

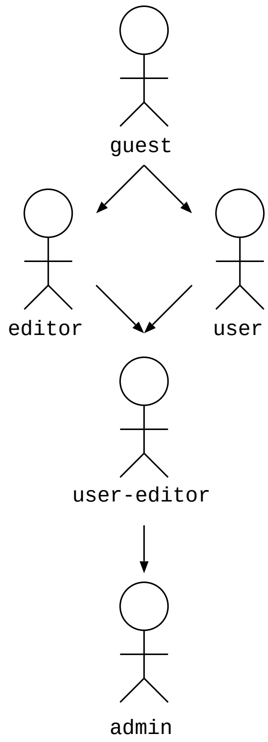
We acted out the following use cases:

1. **offer placing:**
The actor (ed/ue/ad) contacts the user interface, enters his login-data and submits it. The user interface forwards the retrieved data to the new started session which validates the username and the password by contacting the account. Again the account has to ask the storage system for the account information and the privileges of the actor. The result weather the login was successful or not is sent back to the user interface by the session. Now, after successfully logged on, the actor can place his offer by submitting the data of the new real estate to the user interface. The user interface takes the just entered data and forwards it to the session. The session again stores the real estate by sending the data to the storage system. When storing was successful a confirmation message is sent to the user via the session and the user interface. Afterwards the user logs off and the user interface quits the session.

2. **create read account:**
The actor (gu) contacts the user interface that starts as a consequence a new session. Now the actor requests a new read account by sending his personal data to the user interface. The user interface forwards it to the session. The session validates the new data (credit card information,...) and - if everything is correct - forwards the new account data to the account class. The account again stores the new user by contacting the storage system and afterwards calls send-account-information. So the username and the password is sent to the actor via e-mail. After getting a message that the new account creation was successful the user interface closes the session.

Things we had to revise in our concept were:

- The different actors are to be designed hierachically. Going out from the general actor (the guest) all the other actors can be constructed by using inheritance.



- As it is possible that foreign systems will communicate with ours, we needed to add a new class called *fsystem*.

2.4 Lession 4

2.4.1 Preparing for the 4^{th} Lession - Task 4

The essential use cases and interfaces had to be worked out in detail. And again we also had to prepare a short presentation on them.
So we took our user interface - overview, split it up and formed detailed user interfaces out of it.

We came across the following interfaces:

The HTML Interface

- **homepage/startpage**
  - free offers (display only)
  - login
    - user name (input field)
    - password (input field)
  - create read account (link)
- **main menu** - from homepage/startpage
  - all offers (link)
  - offer placing (link)
  - account configuration (link)
  - account administration (link)
  - logoff (link)
- **create read account** - from homepage/startpage
  - title (input field)
  - name (input field)
  - address (input field)
  - phone (input field)
  - mail (input field)
  - credit card number (input field)
  - expire date (input field)
  - business terms (display only)
- **account administration** - from main menu
  - existing user (input field + link)
    - user name (display only)
    - password (input field) X 2
    - privileges (option field)
    - delete account (link)
  - new user (link)

    * user name (input field)
    * password (input field) X 2
    * name (input field)
    * address (input fields)
      ...
    * phone, mail (input fields)
      ...
    * privileges (option field)

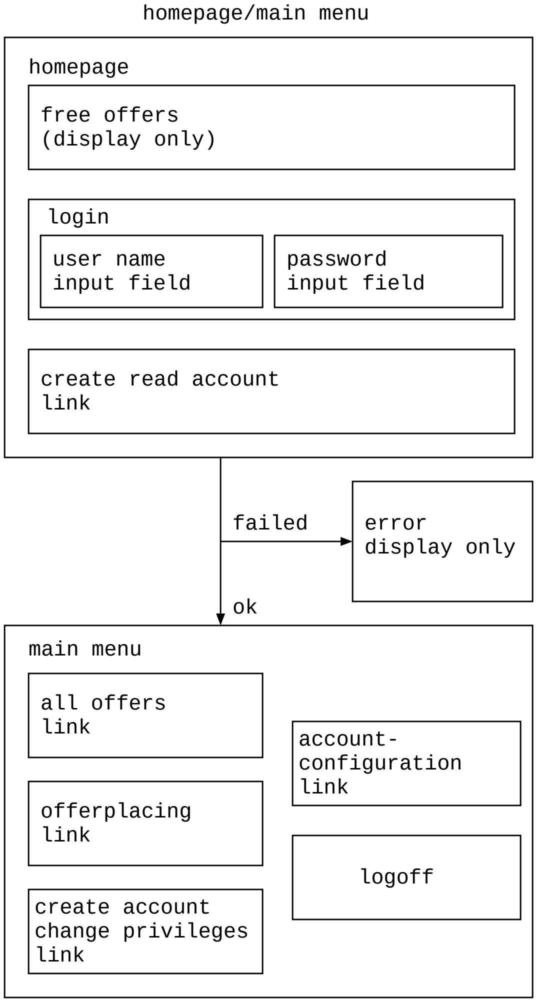
- **all offers** - from main menu
  - search criteria (display only)
  - keyword (input field)
  - search criteria (display only)
  - keyword (input field)
  - ...
  - submit search (link)
- **search results** - from all offers
  - list of real estates (list of links)
  - no matches (displayed only when no search results)
- **account configuration** - from main menu
  - title (input field)
  - name (input field)
  - address (input fields)
  - phone number (input field)
  - cell phone number (input field)
  - e-mail
  - send info on new real estates via SMS (check box)
  - send info on new real estates via e-mail (check box)
  - real estates placed by that user (link list)
- **offer placing/editing** - from main menu
  - kind of real estate (option field)
  - location (input field)
  - price (input field)
  - description (input field)
  - multimedia file upload (link)
  - cancel/delete (link)
- **real estate info** - from search results/account configuration
  - real estate info (display only)
  - edit real estate (link, only displayed if the editor of that r.e. views it)

The WML Interface

As there is by far not enough place on a handy-display, we cropped the whole user interface for HTML to fit on a handy-display. Therefore the users can only search for real estates, browse the results and view some basic infos of the real estates (without multimedia of course).

- **homepage/startpage**
  - user name (input field)
  - password (input field)
- **searching for real estates** - from homepage/startpage
  - search criteria (display only)
  - keyword (input field)
  - search criteria (display only)
  - keyword (input field)
  - ...
  - submit search (link)
- **search results** - from searching for real estates
  - list of real estates (list of links)
  - no matches (displayed only if there are no search results)
- **real estate info** - from search results/account configuration
  - real estate info (display only)

After we got this listing of the use cases we drew the whole thing in TGif for our group paper. The results looked something like this:



By the way: For our presentation of the essential user interfaces we made hand-drawn slides.

2.4.2 The 4^{th} Meeting on 14.11.2001

And again we made a fabulous presentation of our essential user interface. The following corrections had to be made: First we often used *name* as an input field. This has to be split up into *family name* and *first name* as they are entered and stored seperaltely. Also *address* now consists of many more fields: *Country, State, City, ZIP, Street.*
A major revision has to be made with the interface *account administration*:
In our design the administrator had to know the username of any user to be able to modify the accounts. When the admin selects *account administration* he got to a new window where he had the choice to create a new user or to modify an existing one. As creating a new user did not need any modifications we just replaced the input field for the username of existing users with a link to a new user interface:

- **search for members** - from account administration
  - title (input field)
  - family name (input field)
  - first name (input field)

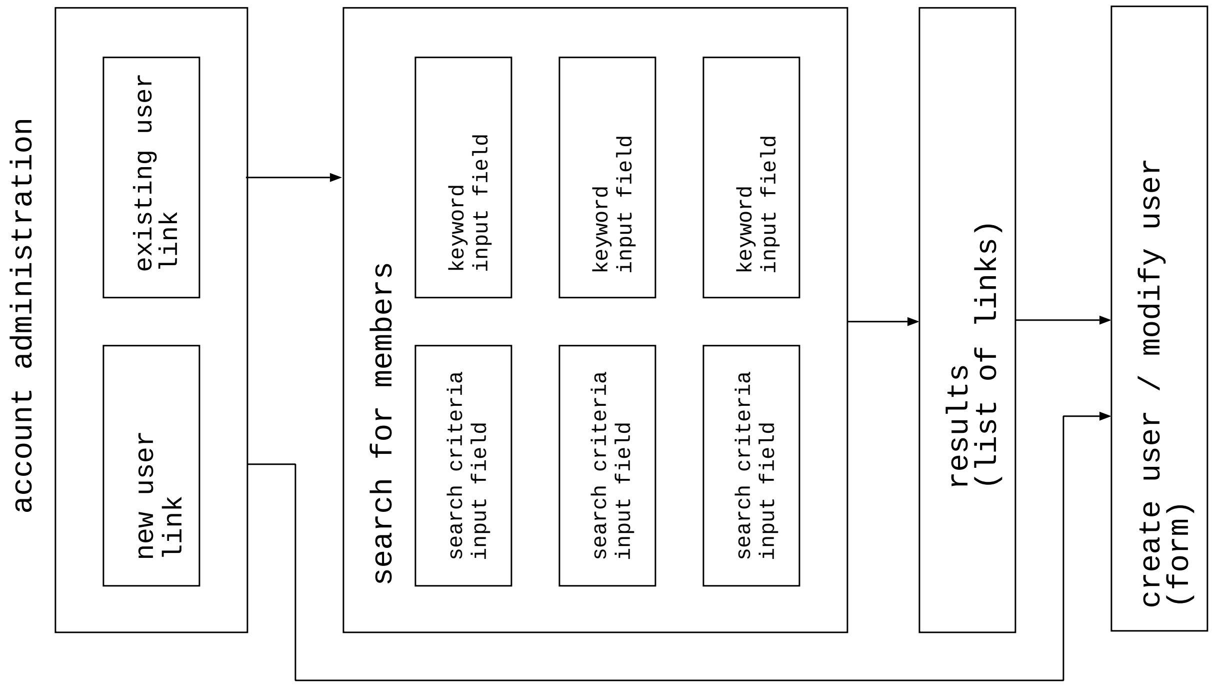
  - country (input field)
  - state (input field)
  - city (input field)
  - zip (input field)
  - street (input field)
  - submit search criteria (link)

- **members search results** - from search for members
  - members (link list)

Now, if an admin wants to modify an existing account he has to go through the following steps:

1. log in,
2. select *account configuration* form the main menu,
3. select *search for members*,
4. enter some personal data of a user (usually the name),
5. submit them,
6. choose the right user from the *search results*.

After performing these steps he can modify the account informations of the selectes user.
The modified user interface looks something like this:

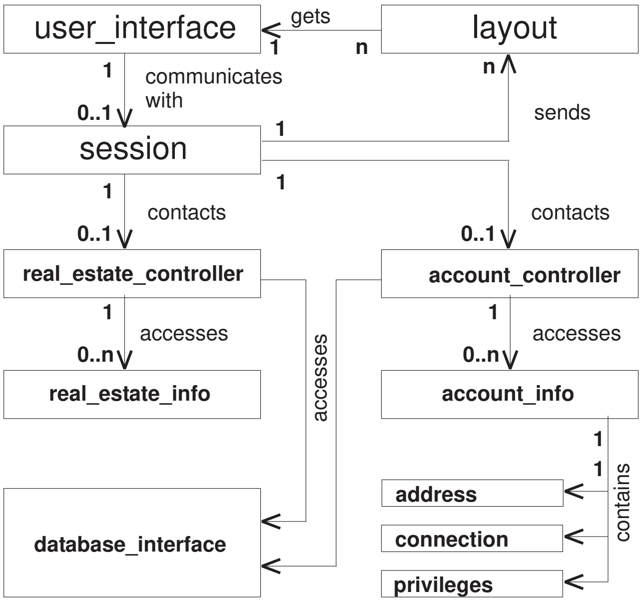


2.5 Lession 5

2.5.1 Preparing for the 5^{th} Lession - Task 5

In that week we had to create class-diagrams and a presentation of them using the results of the previous lessions. Now, this is the crossover from the conceptual model of our system to the object oriented analysis.
First i am going to show you the classes that occurs in our project and their relations among themselves and then the interface (methods) they offer including the attributes:

1. **classes and relations**



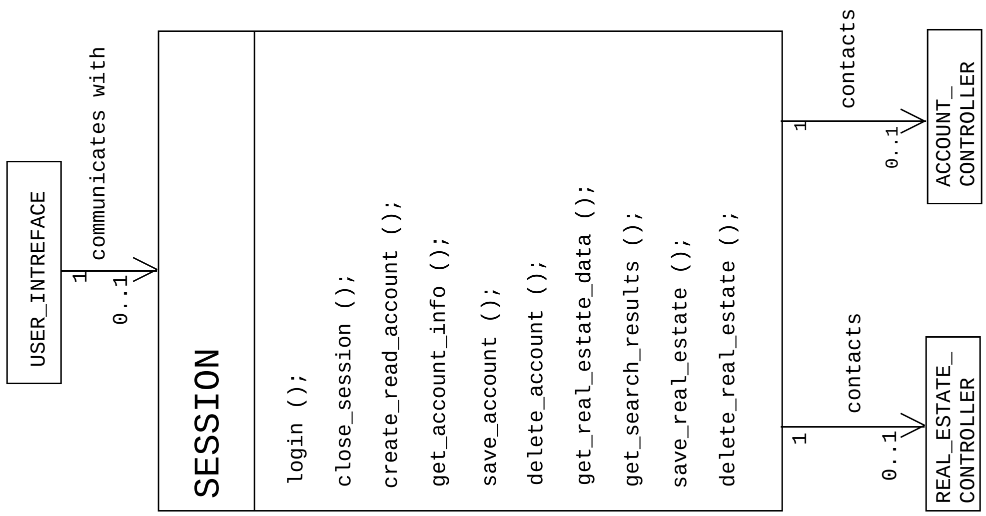
2. **class methods**

user_interface	**layout**
link_pressed() : external	size
button_pressed() : external	color
	widgets
	...

account_controller	**account_info**	**real_estate_controler**	**real_estate_info**
send_account_info()	username	send_re_info()	kind
create_account_info()	password	create_re()	location
modify account()	family_name	modify_re()	price
drop_account()	first_name	drop_re()	description
get_account_data()	country	get_re_data()	files
	state	search_for_re()	validate_data()
	...		
	credit_card_number		
	expiration_date		
	phone		
	cellular_phone		
	e-mail		
	privilege		
	validate_data()		

session	**database_interface**
login()	connect()
close_session()	execute_sql()
create_read_account()	disconnect()
get_account_info()	
save_account()	
delete_account()	
get_real_estate_data()	
get_search_results()	
save_real_estate()	
delete_real_estate()	

Here is an example of the slides we have prepared for the presentation:



2.5.2 The 5^{th} Meeting on 21.11.2001

As the presentation preceded very fast, we had time to discuss some facts:
DI Schwaiger R. told us it is not good to put that many attributes in the class *account_info*, therefore we created some new classes:

address	**connection_info**	**privileges**
family_name	phone number	(us)
first_name	cellular phone number	(ed)
country	e-mail	(ue)
state		(ad)
zip		
street		

We also talked about all the classes we had at that time and which we could drop.
So the following classes were thrown away:

- interrested_party
- homepage
- private_person
- search_for_info

As we now have the class *privilege* we do not need the classes *editor, admin,...* anymore but one single class *user* because this new class in combination with *privileges* contains all the older classes.
So let me give at this time an overview over all classes we have got so far:

user_interface	layout	session	account_conroller	account_info
address	connection_info	user	privileges	real_estate_controller
real_estate_info	database_interface	storage_system	foreign_system	

2.6 Lesson 6

2.6.1 Preparing for the 6^{th} Lesson - Task 6

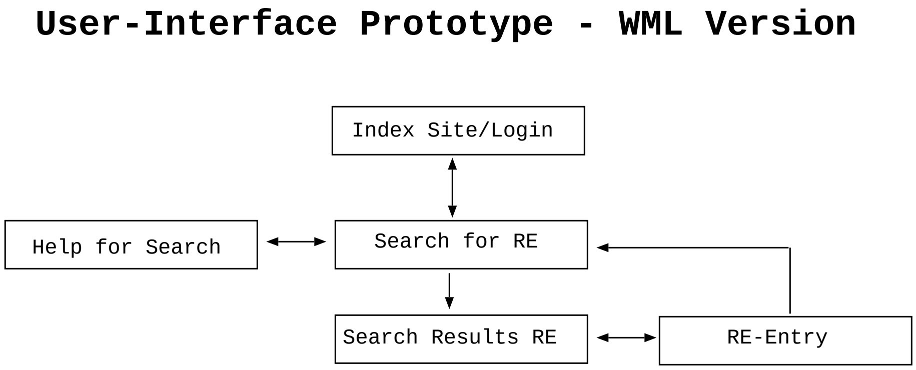
This time we had to transpose the concept of our user-interface we had developed last lession into a real one using HTML and WML. The user interface did not have to work but should be linked together as in the real program.
Therefore we took our conceptional model and wraped it's UI-part into HTML. As there exist many tools for designing HTML-Pages that part of our task was not that difficult. Nevertheless it took us several hours to complete that task. When finished, we put all the HTML files to:

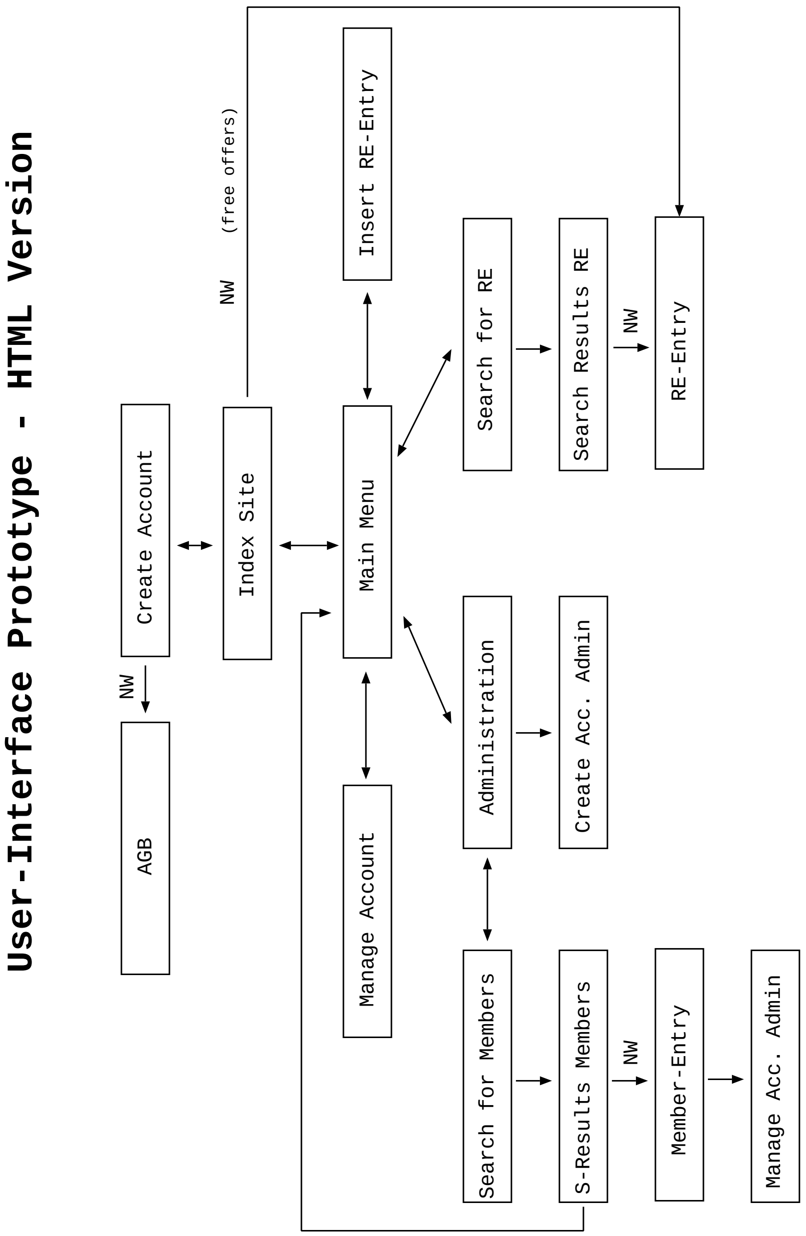
- http://www.cosy.sbg.ac.at/˜mpober/se1/se_html/index1.html

Well, developing an user interface prototype using WML seemed to be more difficult as no one of us had experiences in making WML Pages. But soon we found some tutorials in the Internet (as for example at http://www.wap.org). As the WML user interface just has to be able to perform a search for real estates and no graphics are needed we soon had that prototype ready.
But we came across a problem: When using a local emulator to view the WML files there was everything ok but after transfering the files to our account at cosy we did not have access to them. Ok, we had access to them but all the emulators and handys we tried could not interpret them in the correct way. We got error-messages as for example "Wrong MIME-Type" or "This is not an WML file". After browsing through some forums for WML-programming we found the hint that the server must support WML (we did not expect that beacause we thought that only the syntax of the tags in the file is important). So we applied for a free WML-account at www.wap3.de that was granted imediately. So we uploaded our files and - IT WORKED!
Now, the WML Interface can be found at:

- http://immowarrior.wap3.de

To get an impression how our two different interfaces are built up, here are the flow diagrams:





So far we were well prepared and ready for our next meeting:

2.6.2 The 6^{th} Meeting on 5.12.2001

We presented our HTML Interface by using the beamer of T01. But there occured a problem: Our Homepage was created for screen resolutions 1024*768+ but the beamer only has a resolution of 800*600. So DI Schwaiger R. assigned us to solve that problem for the next meeting. Also the representation of the user's privileges was not optimal as we used a drop-down box for all possible combinations of privileges. So that was another thing we had to think about.

To present our WML-Interface we used a WAP emulator (www.wap.at). The only thing that was not ok with that interface was:
When the user filled in a search keyword into the search form and wanted to enter a second one (what could be done by selecting the link *enter another search keyword*) the previous keywords were not displayed again. So this was another thing to be revised for the next meeting.

2.7 Lesson 7

2.7.1 Preparing for the 7^{th} Lesson - Task 7

Task 7 was to revise our UI-Prototypes particulary with regard to the last lession's results. We also had to specify more precisely the dynamic behavior of the system by using sequence and activity diagrams.

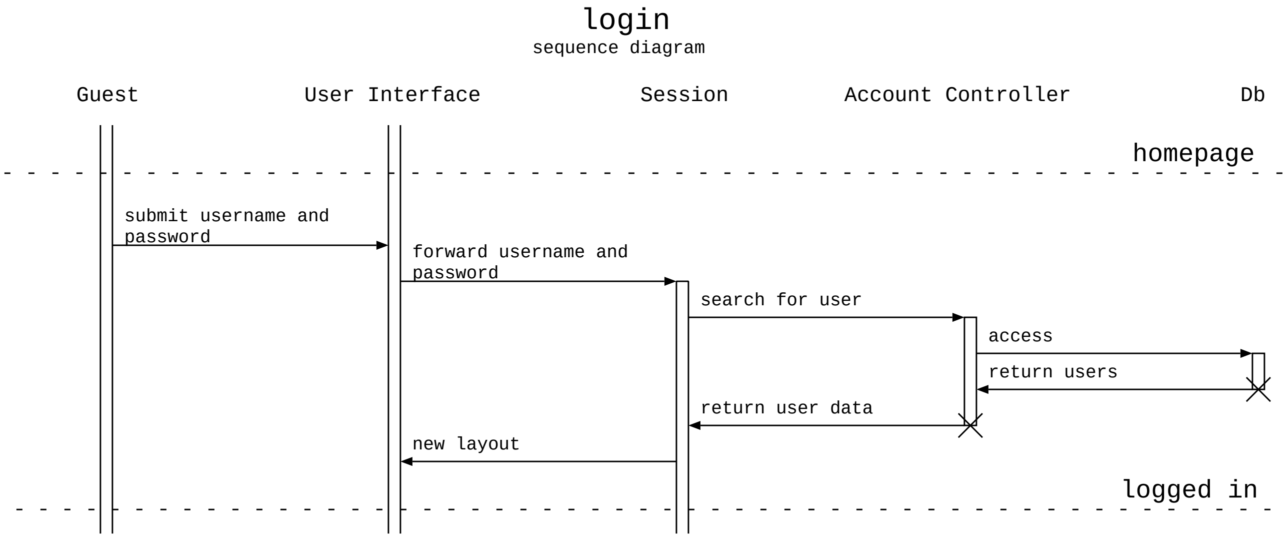
First of all we turned towards revising our UI prototypes. We solved the problems in the following way:

- **HTML Prototype:**
  - **Screen Resolution Problem:** We solved that problem by inserting a new startpage that checks the screen resolution by using JavaScript and directs the user to the correct main page. Therefore we created a copy of the old *start.html*, named it *start800.txt* and adapted it to lower screen resolutions. In the case JavaScript is disabled in the user's browser the user has to select the link corresponding to his screen resolution (800*600 and less, 1024*768 and above).
  - **Displaying Privileges:** As ImmoWarrior should be easily expandable and wrapping privileges into dropdown boxes does not meet that criterion, we decided to use checkboxes. If a new kind of privilege will be added in the future, only a new checkbox will have to be inserted (in a dropdown box there would have to be stored all possible combinations of privileges what trends to grow fast in amount).
- **WML Prototype:**
  - **More than one Search Criteria Problem:** To solve that problem we inserted all previously entered search keywords at the top of the search page.

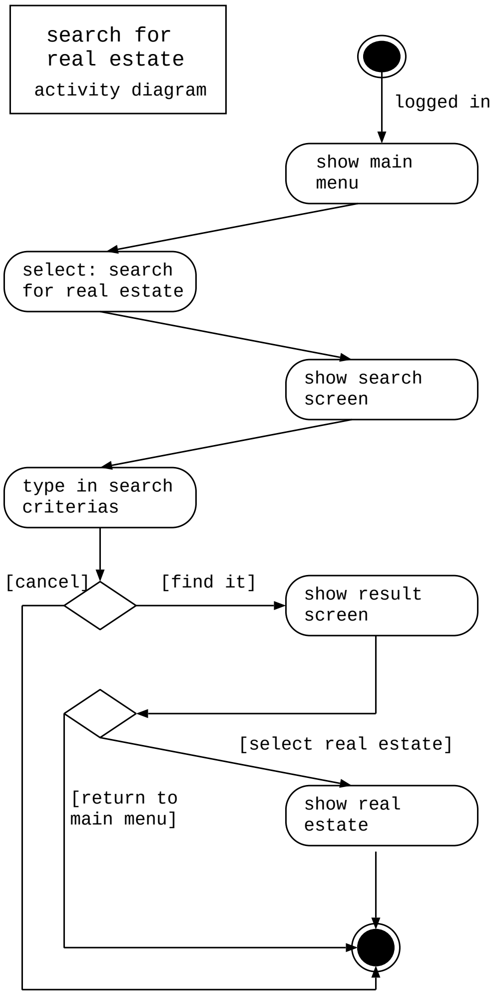
So far, the first part of the 7^{th} task was complete.
Now we had to create **sequence diagrams** and **activity diagrams**.

From the classes and their methods we had got so far we created the sequence diagrams of ImmoWarrior and printed them on slides.

Here are examples of a sequence diagramm and an activity diagram:



.



The complete collection of the sequence diagrams and activity diagrams can be found in the *Group Paper*.

2.7.2 The 7^{th} Meeting on 12.12.2001

Surprisingly we did not have to show our revised UI prototypes.

So the only thing we presented were our sequence and activity diagrams. We did not get trough all of them by far. Therefore there was not much time for discussion left.

Things that were to be revised:

- The flow diagram is to be created from the user's point of view only. We had also included some activities of the system.
- In the sequence diagrams the description of what is going on has to be written down on the left side of the diagram and not over the called methods.

2.8 Lesson 8

2.8.1 Finishing the Personal Paper - Task 8

Lesson 8 was not an actual meeting, we just had to send the current version of our personal paper to DI Schwaiger Roland.