PS Software Engineering
December 19, 2001

ImmoWarrior

Personal Report by Michael-A. Pober

mpober@cosy.sbg.ac.at

Academic Supervisor Mag. DI Dr. Roland Schwaiger

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

Prelude

In the Proseminar Software Engineering we deal with the subject of object oriented software development.
In a "learning by doing" manner we explore and try out professional methods to efficiently design a software product according to the customers wishes and requirements.
For this purpose we simulate a real life situation:

Project ImmoWarrior
A company that auctions real estate wants us to develop a software-solution for its process of collecting and representing information. (Code: "ImmoWarrior").
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 by using XML
- ImmoWarrior displays the information for a person - what requires us to design an appropriate user-interface.

The client requests distinctively, 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 first phase of the project, we ought to concentrate on the basic functions of the software product, but we should keep the possibility in mind to add expansion, developed in cooperation with our team.
The deadline for this project is the end of January 2002.

In our simulation the client is represented by the Proseminar-Leader. To find out about his wishes and requirements regarding the software product and to present our proposals and results, a meeting is held on a weekly basis.

One word on this paper: I have included just some of the diagrams, we created for the lessons serve as examples for the different diagram types. If you want to see the complete set, please refer to the group-paper.

Chapter 2

The 1^{st} Lesson

Basic information about the organization and the goals of the Proseminar Software Engineering is given. We form teams and our Proseminar-Leader presents us two possible Projects to choose from, "InfoWarrior" and "ImmoWarrior".
We, TheTeam (a.k.a. Team 5) select the latter.

Chapter 3

The 2^{nd} Lesson

3.1 The Assignment

Plan a CRC Card Session. For that purpose the team has to act out all the necessary processes involved in a CRC Session.
Following questions are to be dealt with:

- What is a CRC Card Session?
- What is its goal?
- How is the session organized?
- Who has what function?
- Who protocols the results of the session?

3.2 What are CRC Cards?

CRC cards where introduced by Kent Beck and Ward Cunningham in their paper "A Laboratory for Teaching Object-Oriented Thinking" released in OOPLSA '89. CRC stands for "Class, Responsibility, Collaborator". It is an informal technique for object oriented analysis and design, used for visualizing the structure and behaviour of object oriented systems. Typically a CRC card is a standard index card that has been divided into three sections.

Class A class represents a collection of similar objects. An object is a person, place, thing, event, concept, screen, or report that is relevant to the system at hand. The name of a class appears across the top of the card.

Responsibility A responsibility is anything that a class knows or does. For example, customers have names, customer numbers, and phone numbers. These are the things that a customer knows. Customers also order products, cancel orders, and make payments. These are the things that a customer does. Responsibilities are shown on the left hand column of a CRC card.

Collaborator Sometimes a class will have a responsibility to fulfill, but will not have enough information to do it. When this happens it has to collaborate with other classes to get the job done. The collaborators of a class are shown in the right-hand column of a CRC card.

This is an example for the design of a CRC card:

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

3.3 Using the CRC Cards

The cards are used to carry out a so called CRC card sessions. In this session a group tries to define possible Use Cases (actions and interactions, the software has to perform) and the software components required to be able to execute and support this use cases.

3.3.1 The Group

The ideal group size for a CRC card session is five or six people. This size generally allows everyone to productively participate. The group should be composed of developers, business domain experts, and an object-oriented technology facilitator.

3.3.2 The Session

Before starting a session there need to be some kinds of requirements for the systems. Weather they are implicitly or explicitly defined, the people in the group need to be familiar with them. A session also needs to focus on one part of the problem at a time. So one by one a subset of the problem needs to be chosen to be explored during the CRC card session.

Creating Classes The first step in modeling an system in the object-oriented paradigm is to identify the classes in the problem domain. One useful tool is to search all of the nouns in the problem statement. The nouns are a good key to what classes are in the system. This information is used as the basis for a brainstorming session that results in a list of possible classes. After that, the CRC cards are passed out to the members of the group and everyone is assigned at least one class. The group members are now the owner of that class during the session. Each person records the name of his/her class on the card. One class per card.

Finding Responsibilities Now the group has to determine the responsibilities. This time the verbs in the problem statement are a good clue to what the responsibilities are going to be. Typically responsibilities are added on the CRC cards that are obvious from the requirements or the name and purpose of the classes. It is not necessary to find every possible responsibility, as the lacking ones will be found later on in the scenario. (This fact also applies to the classes,...).

Use Case Scenario Testing Use case scenario testing is a task process pattern in which users are actively involved with ensuring that user requirements are accurate. The basic idea is that a group of business domain experts, with the aid of a facilitator, step through a series of defined use cases to verify that CRC model accurately reflects their requirements. The facilitator distributes the cards to the business domain experts, trying to ensure that two cards that collaborate are given to two different people . The facilitator then leads the group through acting out each scenario one at a time. There are six steps to acting out scenarios:

1. Call out a new scenario.
   Both the description of the scenario and the action(s) to be taken are called out and the group must decide if this scenario is reasonable.
2. Determine which card should handle the responsibility.
   When a scenario has been described, or when the need for collaboration has been identified, the group should decide what CRC card should handle the responsibility.
3. Update the cards whenever necessary.
   For example the responsibility needs to be added to an existing card, or a new card needs to be created with that responsibility.
4. Describe the processing logic.
   The business logic for the responsibilities should be described step by step and the rules for the system should be recorded.
5. Collaborate if necessary.
   When a group member describes the business logic of the responsibility, he/she will often describe a step where the card needs to collaborate with another card to complete its task.
6. When done go back to step 1.

3.4 Advantages of CRC Modeling

- The analysis is done by people who understand the problem domain (experts).
- The future users are actively involved in defining the model for the program.
- CRC modeling can easily be explained to a group of people.
- A CRC session leads directly into continuing steps of software design, like drawing class diagrams.

3.5 Our Preparations and the 2^{nd} Lesson

- Team-leader: Christian Koidl
- Participating team-Members: Andrea Geierspichler, Gunnar Ruhs, Roman Gross, Michael Pober
- Topics talked about: Analysis of the problem and basic class definition
- Additional Information concerning the Problem Domain:
  As we know from the conceptual information about the project, the client has a firm, that auctions real estate. He wants us to create some kind of interactive database that provides a set of services concerning real estate offers (collecting offers, informing customers about new entries, help customers to find an appropriate offer,...).
  Here is a list of special requirements to the software system:
  - The system has to be accessible via an user interface that is open to everyone, a site on the Internet (including an WAP-site that can be visited using a cellular phone.)
  - There are various groups of users, each one with a different set of privileges:
    Private Person They visit the homepage and can only obtain information on a limited number of free offers. To be able to view more information, they have to become members. Therefore guests can create an account
    User Users are registered members. They have access to a special members-area in the user interface. From here they can search the entire database for real estate entries. A user may modify his account information.
    Editor Editors, like the users, are also registered and can access the members-area. They have the permission to enter a new real estate offer into the database and to modify the information on their entries.
    Admin An admin is some kind of an "omnipotent" user. Admins have access to the entire database. They can create and modify real estate entries and accounts. An admin sets the privileges for an account.
  - To ensure, that nobody has access to areas, where he is not allowed to go to, the software has to provide an access-control system.
  - After demanding an account, private persons receive their username and password in an e-mail.
  - One has not necessarily to be an employees of the client's firm to be able to get Editor-privileges.
  - If private persons want to place an real estate offer in our system, they have to pay an certain amount of provision.
  - Customers such as banks, authorities and private persons get only one account.
  - When placing an offer in our system, there should be the possibility to include different multimedia-elements auch as pictures, movies, sounds,...
  - Information on new offers should be sent to the customers either automatically by the system or on individual request.

- List of possible classes: Session, User Interface, Account, Permission, Information Storage, Real Estate, Editor, Private Person, Prospective Buyer
- Open questions:
  - Should permanent Editor-accounts be available for private persons?
  - Do private persons have to pay provision for their offer although the object is not sold?
  - Is it necessary to have an account to request information?

Chapter 4

The 3^{rd} Lesson

4.1 The Assignment

Refine the problem using the results of the first CRC Card Session and the according protocols. Plan another CRC Session to discuss open questions with the clients representative. In this second Session Use Cases will be an important topic. In addition think about Essential User Interface Prototyping.

4.2 Our Preparations and the 3^{rd} Lesson

- Team-leader: Andrea Geierspichler
- Participating team-members: Christian Koidl, Gunnar Ruhs, Roman Gross, Michael Pober
- Topics talked about: Use case definition, essential user interface modeling
- Results:
  - We renamed some of the CRC cards, we created last week. We changed Permission to Privileges (that is the correct term) and Perspective Buyer to User.
  - While talking about the system, we realized, that the process of sending the account information to a newly registered user has to be modeled as a separate element. Therefore we have added a new CRC card: Send Account Information.
  - We developed a number of use cases:
Viewing free Offers
Actor: All categories
Details: The user visits the homepage. The homepage starts a session, that queries the information storage for the free offers. The storage returns this data via the session to the homepage and the session is closed. The homepage displays the gathered information.
Create Account and Send Account Information
Actor: Private Persons
Details: The private persons comes to the homepage and views the free offers.

To be able to get information he has to become a member, so he enters the "create account" menu. Here a session is started and the private person is offered a form by the user interface, where he enters his personal information. After finishing, the private person clicks the submit button, the session gets the data and forwards it to the information storage. If the operation was successful, an acknowledgement is sent via the session to the user interface. Then the session submits the necessary data to Send Account Information, from where the user is send an e-mail with his username and password. Finally the session queries for the free offers, returns them to the homepage and closes. The homepage screen is displayed again.

Login, Offer Placing and Logout

Actors: Editors

Details: The editor contacts the user interface. The interface forwards the login inputs to the Account. Account checks in the information storage, if the user-account exists. When receiving an acknowledgement the Account sends the data to the Privilege. The Privilege gets the users permissions from the information storage and returns this data to the Account. From here the result is returned to the user interface. If the user is accepted, then the interface starts a session and displays the next screen, otherwise the user is informed, that his inputs were not acceptable.

In the members-area of the user interface, the user chooses the offer placing menu. Here he enters his real estate data into a form. The interface reads the information and forwards it to the session, which stores it in the information storage. The storage sends a success message to the user-interface via the session. The interface displays the success message and returns to the menu screen. The user clicks the logout-button and the interface closes the session and shows the homepage.

Manage Account

Actors: All categories except the private persons

Details: The user, who is already logged into the system selects the Manage Account menu in the members area. The session gathers the user information from the information storage and forwards it to the user interface, that displays it in a corresponding form. The user changes his details and submits the information again. The session gathers the data and forwards it to the information storage. The storage sends a success message to the user-interface via the session. If the password was changed, then the session activates the Send Account Information to send the username and the new password to the user (as acknowledgement).

Viewing all offers

Actors: Users, Editors, Admins

Details: Again the user is already logged in. He selects the View all Offers menu. The session receives this command and makes the user interface changes to the search screen. In this form the user can enter his search criteria. The session collects this data and submits it to the information storage. A list of matching offers is returned via the session to the user interface, that displays the results. Now the user chooses one of the offers. The session forwards the selection to the information storage, that returns more details to the according real estate entry. This data is once again transferred to the user interface to be displayed for the user.

  - During the CRC session we have also begun with a rudimentary design for the user interface. For now we have the following scheme: The homepage with a login-window. Then we need a screen for "Create Account". After the user entered his login information either an "Access Denied" window opens when the entry was incorrect, or the user is transferred to the main menu, where he can enter new real estate offers (using an appropriate form) or view the entries in the database. We are also going to need additional info-screens for "Operation Successful" and "Operation Failed".

- List of possible classes: Session, User Interface, Account, Privilege, Information Storage, Real Estate, Editor, Private Person, User, Send Account Information

- Open questions:
  - How is a user able to manage his account?
  - Is "Privileges" necessary?

Chapter 5

The 4^{th} Lesson

5.1 The Assignment

Describe the Essential Use Cases and the Essential User Interface. Prepare to present your results with slides.

5.2 What is a Use Case Diagram?

Use case diagrams model the functionality of a system using actors and use cases. Each use case has an actor that is allowed to execute it, that initializes its flow of events. Actors, are a representation of the outside world, and they can be people or other computer systems. One physical object may also play several roles and therefore several actors may model it.

5.3 Modeling a Use Case Diagram

This list shows the basic elements, that occur in an use case diagram:

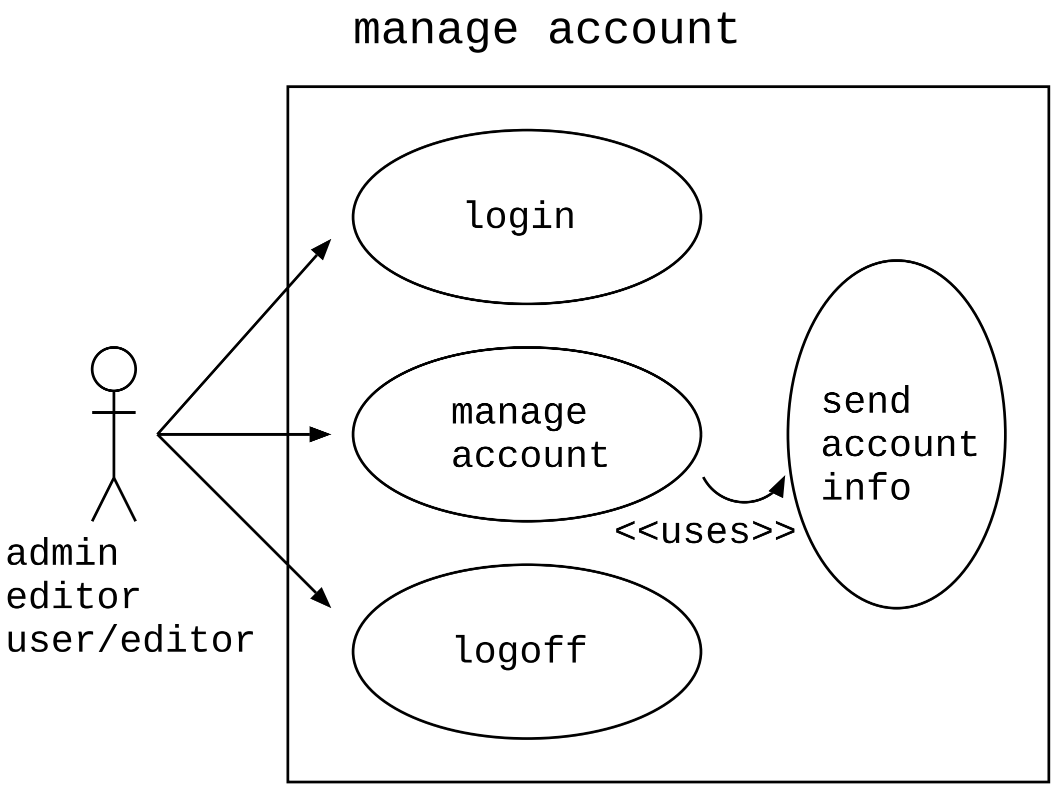
System The system's boundaries are represented using a rectangle that contains the different use cases. Actors are placed on the outside of the system's boundaries.

Use Case The symbol for a use case is an oval. The verb that represents the use cases functions is place in the center of the ellipse.

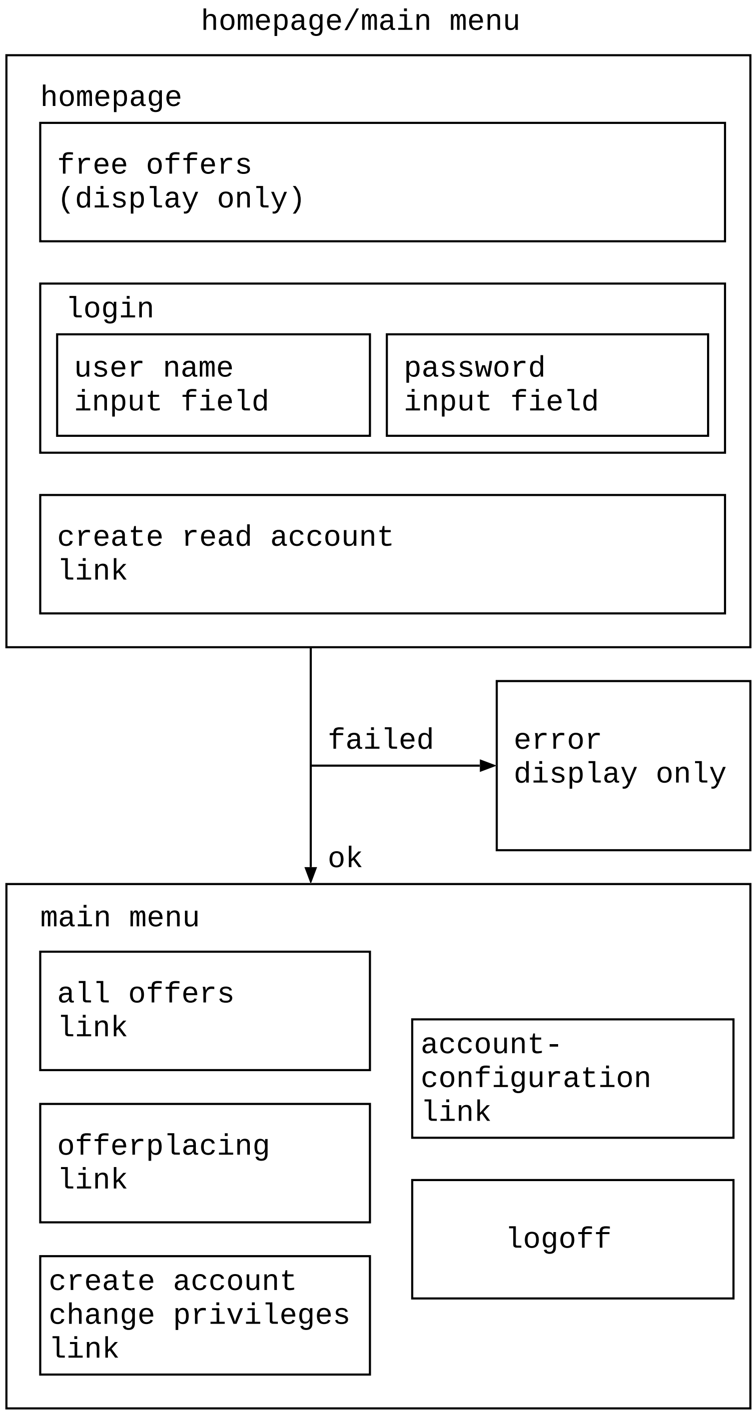
Actors They are the users of a system. An human actor is illustrated in the use case diagram by a match stick man with its lable on the bottom. When a system is the actor of another system, the actor system is labeled with the actor stereotype.

Relationship Relationships between an actor and a use case are represented with a simple line. For relationships among use cases, arrows labeled either "uses" or "extends" are used. A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.

On the following figure you see the use case diagram we designed for "Manage Account".



The next diagram is a rudimentary design for a part of our user interface. In this case you see the homepage including the login form and the main menu of the members area.



5.4 Our Preparations and the 4^{th} Lesson

- Team-leader: Gunnar Ruhs
- Participating team-members: Christian Koidl, Andrea Geierspichler, Roman Gross, Michael Pober
- Topics talked about: Presentation of the Essential Use Cases and a basic design for the user interface.
- Results:
  - We have renamed the CRC card Private Person to Guest. Also Viewing All Offers was changed to Search for Info, which is more appropriate.
  - As it is requested by the client, that the system has to be able to send information on new real estate offers to the customers via e-mail or to a mobile phone, we had to include a new class, "Push Real Estate Info", (not unlike "Send Account Info")that manages this task.
  - Another modification to the CRC cards: We restricted the permissions for the Editor. An editor has no longer read access to our database, because he only pays for entering information, not for viewing all our offers. As compensation we have created an new kind of user, the EditorAndUser. This category has combined privileges of Editors and Users. People in this class may enter new offers and view information on real estate in the entire database.
  - During the presentation of our propositions, the proseminar-leader explained to us an easier design for the different user groups (User, Editor,...). It is possible to model the various categories using inheritance. Guest is the superclass, and from there we can inherit the other privileges. By going through the line from Guest to Admin, we just have to add the new permissions.
  - Another point, that occurred in the presentation is the need for a new class "System". This represents a foreign system, that tries to interact with our software.
- List of possible classes: Session, User Interface, Account, Information Storage, Admin, Editor, EditorAndUser, User, Guest, Send Account Info, Search for Info, Push Real Estate Info, System

Chapter 6

The 5^{th} Lesson

6.1 The Assignment

Draw an Analysis Class Diagram basing on your CRC Cards, your Essential User Interface and your Use Cases. Present the results with slides.

6.2 What is a Class Diagram?

Now, that we have a conceptional CRC model of our system, we have to analyze this model in an object oriented manner. Therefore we have to create a class diagram. This diagram describes the types of objects in the system and various kinds of static relationships which exist between them. There are three principal kinds of relationships which are important: associations, generalization and aggregation.

6.3 Modeling a Class Diagram

Here is a list of basic elements that appear in a class diagram:

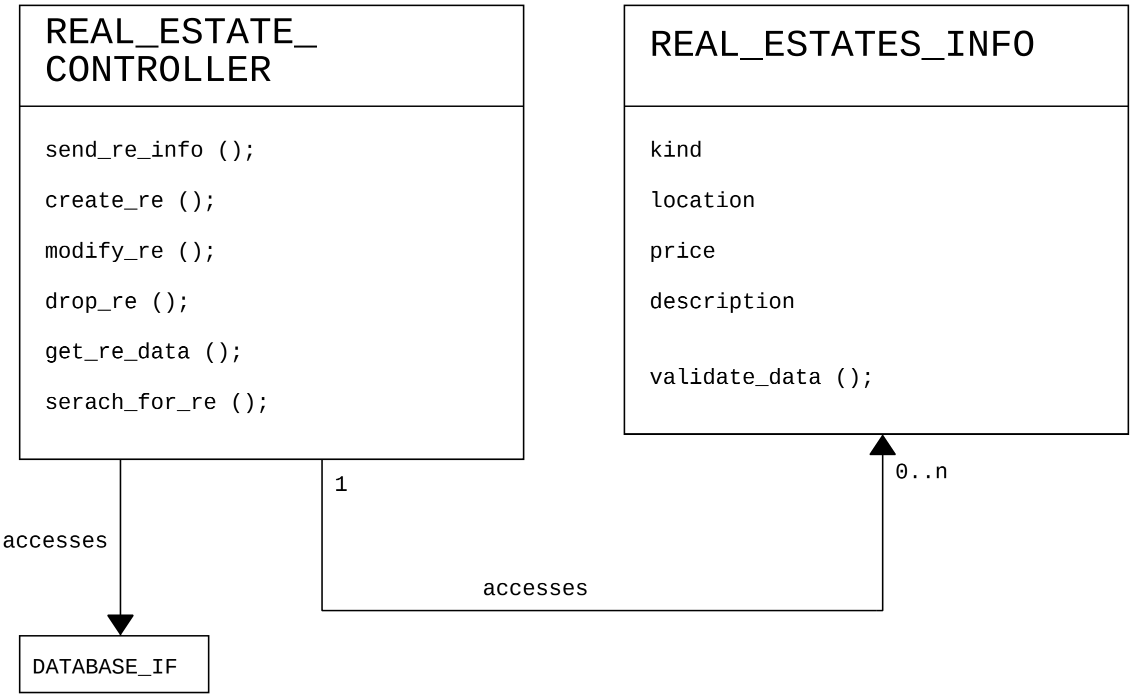
Classes Every CRC card in our model corresponds to a single class. We just have to transform them. In this diagram classes are illustrated with rectangles divided into compartments. The name of the class is placed in the first partition (centered, bolded, and capitalized). The attributes are listed in the second partition and the operations in the third.

Associations Associations represent relationships between instances of types. Typically they represent conceptual relationships between the types involved. An association is represented in the diagram by a line between two classes, the name is stated above or below it. Associations may be bidirectional or unidirectional. An arrow indicates the alignment.
One of the key aspects of an association is its cardinality. The cardinality notations are placed near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.

Generalization Generalization is another name for inheritance or an ”is a” relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, a coniferous tree is a type of tree. So the class cedar would have a generalization relationship with the class tree. In the Class Diagram generalization is shown by an arrow with an empty head from the subclass (coniferous tree) to the superclass (tree).

Composition and Aggregation Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. A composition is illustrated with a filled diamond. A hollow diamond is used to represent a simple aggregation relationship, in which the ”whole” class plays a more important role than the ”part” class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the ”whole” class or the aggregate.

In the next figure you will see an example for a class diagram. This one was created for the CRC card ”Real Estate”. You will find an explanation to this diagram in the following section.



6.4 Our Preparations and the 5th Lesson

- Team-leader: Roman Gross
- Participating team-members: Christian Koidl, Andrea Geierspichler, Gunnar Ruhs, Michael Pober
- Topics talked about: Presentation of the class diagrams

- Results:
  - While we were planning the class diagrams, we realized, that we had to split up the class Account into an Account Controller and an Account Info class. The Account Info is basically a data record and the Account Controller provides the functions to work with this data records (modify account, create account,...). So for example when the Account Controller asks the information storage for an account entry, the database will return an object of Account Info.
    We have the same situation with the class Real Estate too. We likewise had to split it into Real Estate Controller and Real Estate Info.
  - One other thing, that we redesigned during the diagram drawing phase is the information storage. We split it up into two sections. One is the actual data storage in form of a database on a server. The other is the "Database Interface". This is our connection to the database. This interface has to transform the user's requests into a SQL statement and perform the query. This way our software doesn't have to provide an own data-management but can cooperate with any form of SQL database, which will surely increase its applicability.
  - During the presentation we noticed, how important it is to label the associations between the different classes, and to give them a direction (marked with arrows), otherwise it is quite difficult to recognize what class is doing what with who.
  - Another detail that simplifies understanding a class diagram (and presumably the project realization basing on these diagrams in the end) is to mark the cardinalities.
  - Attributes as "Privileges" and "Address" that consist of diverse elements (i.e. Street, City, State for Address) have to be modeled as own classes.
  - The proseminar leader showed us an professional design method concerning our user interface scheme: The layout for the user interface can be put into an own class. So we are able to model all the different sites as objects of the class "layout". Thereby it is not necessary to give every single sites its own layout properties, because it can easily inherit them.

Chapter 7

The 6^{th} Lesson

7.1 The Assignment

Develop an Essential User Interface Prototype using HTML and WML. Test, if you can act out all your Use Cases on the Prototype. Going out form your User Interface, draw an according Flow Diagram and try to discover new requirements to the system.

7.2 What is the Purpose of Prototyping?

User interface prototyping is done to show people the possible design(s) for the user interface of an application.
There are four basic steps in the prototyping process:

1. Determine the needs of the users (for example by using the information gathered during the CRC card sessions). The requirements drive the development of the prototype as they define the business objects that the system must support.

2. Building the prototype. Using a prototyping tool or high-level language, the screens and reports needed by the users are designed. The best advice during this stage of the process is to not invest a lot of time in making the code "good" because chances are high that a lot of changes have do be done anyway after step 3.

3. Evaluating the prototype. After a version of the prototype is built it needs to be evaluated. It helps, to address three basic issues during evaluation: Whats good about the prototype, whats bad about the prototype, and whats missing from the prototype. An important point is to verify all the sites are interlinked and no unexpected dead ends exist.

4. Determine if the process of prototyping is finished yet. The end of prototyping is reached when the evaluation process no longer generates any new requirements, or only a small number of not-so-important requirements.

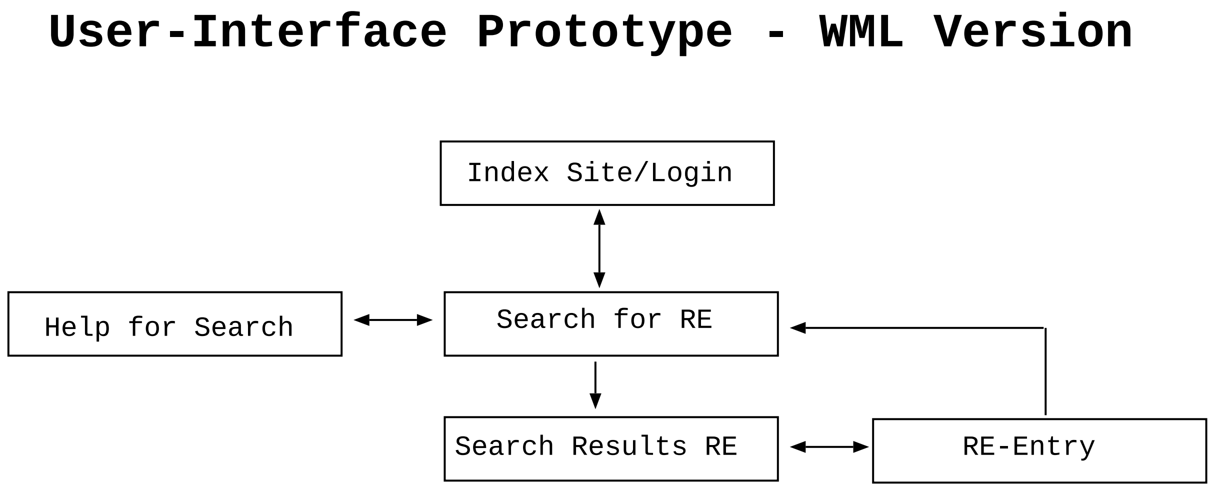
The prototypes of the ImmoWarrior user interface can be found at the following locations:

- HTML version: http://www.cosy.sbg.ac.at/~mpober/se1/se_html/
- WML version: http://immowarrior.wap3.de/

7.3 What is a Flow Diagram?

User interface flow diagrams show the relationships between the user interface components, screens and reports, that make up the application. The user interface objects (screens, reports, or forms) are represented by boxes and arrows stand for the possible flow between the screens. Because interface flow diagrams offer a high-level view of the interface of a system one can quickly gain an understanding of how the system is expected to work. That makes it easy to validate the overall flow of the applications user interface and its design.

On the following illustration you will see an example of a flow diagram.



This diagram was made to demonstrate the function of of the WML user interface prototype. You start on the "Index Site" that includes the fields for entering the username and password. From there you get to the "Search for Real Estate" Site (bidirectional), where you can enter search criteria. If you want additional information on the function of this interface you can click the "Help"-Button, that transfers you to the "Help for Search"-Site (bidirectional). If you have entered your keys for finding useful information, you click on "Find It" and get to the "Search Results" site. There a list with all real estate entries corresponding to your search criteria is presented. After choosing one, you reach the "Real Estate Entry" site, where you can obtain a more detailed information about the offer. From this site you can return to the Results page or go back to the "Search for Real Estate" to start another query.

7.4 Our Preparations and the 6^{th} Lesson

- Team-leader: Michael Pober
- Participating Team-members: Christian Koidl, Andrea Geierspichler, Gunnar Ruhs, Roman Gross
- Topics talked about: Presentation of the User-Interface-Prototype (WML and HTML)

- Results:
  - During the prototype development we noticed, that in some areas the software must provide special functions for an admin, that enable him to manage the system. Therefore we added an extra link in the main menu, that leads to a separate administration sub-menu. From this menu an admin can get to special Create Account and Manage Account screens, where for example he has the possibility to assign the different privileges to the users. Another thing that is different to the screens of the "normal" users: Once an user has entered his personal data on account creation, he may no longer change his credit card information. If he wants to do so, he has to contact the admin, who is the only one, who has an corresponding field in his manage account screen. With this limitation we want to prevent, that once we have certified, that the payment-information is correct, the user changes it to avoid charging.
    Something that has to be mentioned about the Manage Account link in the administration menu: By clicking this link, the admin is not directly transferred to his manage account form, but to a newly added Search for Members screen. According to the clients information, the firm has thousands of customers, that will use our software, so it is necessary, that the system provides a function Search for Members (pretty similar to Search for Real Estate). With this service the admin can easily locate the account he wants to manage.
  - We constructed the user interface for a screen resolution of 1024x768 (which we thought was nowadays standard). The computer on which we had to present our work was adjusted to 800x600. This way we very clearly saw, how "unappealing" our design can look an a lower resolution than that it was designed for. So the client wants us to add a function to the user interface, that checks the users screen settings and accordingly loads the appropriate interface version.
  - While talking about the administration screens, the proseminar leader noticed, that our design was to fixed on a predefined set of user privileges. In our case it would be very complicated to add a new class of permission, if the situation required to. So we have to rethink this design.
  - Another problem, the proseminar leader found during the presentation: Till then we restricted, that an Editor has the permission to enter new real estate offers and to modify them, but he may not view all offers. So something, that we forgot about was, that although the Editor may not see all entries in our database, he must still be able to view the offers he has created. Otherwise it would be impossible to modify or update them. So something, that we have to add to our system is a possibility to determine, which entries a user has created and to access them.
  - A point concerning the WML interface: as already mentioned, this interface provides a function to search the database for real estate entries. In our design, the user is able to select a search criteria, submit it and than choose another. Something, that you did not have in our initial design is a list that shows the selected criteria. The client wants us to add this feature.

Chapter 8

The 7^{th} Lesson

8.1 The Assignment

Refine the Essential User Interface Prototype according to the insights you gained in the last session. Design Sequence Diagrams and Activity Diagrams to point out the dynamic structure of the system and the interaction of the single objects/classes. Present your results with slides.

8.2 Sequence Diagram

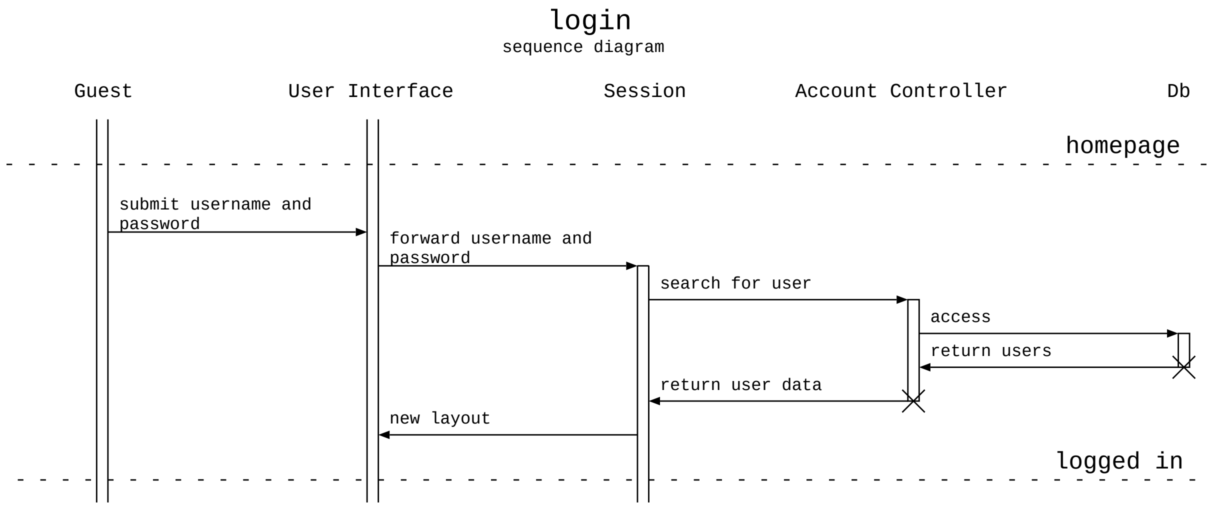
8.2.1 What is a Sequence Diagram?

A sequence diagram shows an interaction arranged in time sequence. In particular, it shows the objects participating in the interaction by their "lifelines". It does not show the associations among the objects. The sequence diagram represents an Interaction, which is a set of messages exchanged among objects within a collaboration to effect a desired operation or result.
A sequence diagram has two dimensions: the vertical dimension represents time, the horizontal dimension represents different objects. Normally time proceeds down the page. Usually only time sequences are important but in real-time applications, the time axis could be an actual metric. There is no significance to the horizontal ordering of the objects.

8.2.2 Modeling a Sequence Diagram

An object in a sequence diagram is represented by a rectangle. The objects role is shown as a vertical dashed line called the "lifeline". This lifeline represents the existence of the object at a particular time. If the object is created or destroyed during the period of time shown on the diagram, then its lifeline starts or stops at the appropriate point; otherwise it goes from the top to the bottom of the diagram.
A message is a communication between objects that conveys information with the expectation that an action will follow. The receipt of a message is a kind of event. A message is shown as a horizontal solid arrow from the lifeline of one object to the lifeline of another object. In case of a message from an object to itself, the arrow may start and finish on the

same object symbol. The arrow is labeled with the name of the message and its argument values. The arrow may also be labeled with a sequence number to show the sequence of the message in the overall interaction. Sequence numbers are useful on the diagrams for identifying concurrent threads of control.
On the following figure you see an example of a sequence diagram. This one shows the process of logging into our system.



8.2.3 When to use a Sequence Diagram

A good design can have lots of small methods in different classes. Because of this it can be difficult to figure out the overall sequence of behavior. This diagram is simple and visually logical, so it is easy to see the sequence of the flow of control. The diagram also clearly shows concurrent processes and activation.

8.3 Activity Diagram

8.3.1 What is an Activity Diagram?

Activity diagrams describe the workflow behavior of a system. The diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. An activity is trigged by one or more events and activity may result in one or more events that may trigger other activity or processes.
Typically an activity diagram is attached to the implementation of an use case. The purpose of this diagram is to focus on flows driven by internal processing (as opposed to external events)

8.3.2 Modeling an Activity Diagram

Basic elements for creating an activity diagram:

Action State Action states represent the non interruptible actions of objects. Typically an action state is represented by an rectangle with convex arcs as sides.

Action Flow Action flow arrows illustrate the relationships among action states.

Object Flow Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.

Initial State A filled circle followed by an arrow represents the initial action state.

Final State An arrow pointing to a filled circle nested inside another circle represents the final action state.

Branching A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."

Swimlanes: The activity diagram may be divided visually into "swimlanes" each separated from neighboring swimlanes by vertical solid lines on both sides. Swimlanes are used to group related activities into one column.

8.3.3 When to use an Activity Diagram

Activity diagrams are used for documenting existing process, analyzing new Process Concepts, identifying and finding re-engineering opportunities. They are mostly used to show parallel behavior between the different events and activities. Activity diagrams are also used to document decisions and iterations. From the software engineering perspective they can be used to analyze requirements of the use cases before assigning methods.

Advantages

- The activity diagram focuses on activities, in this sense it is like a flow chart supporting compound decisions. However, it differs from a flow chart by explicitly supporting parallel activities and their synchronization.
- The starting point eliminates the need to traverse through the entire process model for identifying the process start or triggering event that is of user interest.
- The back flows of activities can be represented.

8.4 Our Preparations and the 7^{th} Lesson

- Team-leader: Christian Koidl
- Participating Team-members: Andrea Geierspichler, Gunnar Ruhs, Roman Gross, Michael Pober
- Topics talked about: Presentation of the activity diagrams and sequence diagrams

- Results:
  - In an activity diagram it is important to also show the actions of an user. In our designs, we only gave attention to the processes that were happening inside our system, we did not mention any actions of an actor on the outside. So for example in the activity diagram for viewing free offers, we have to insert, after the system shows a set of appropriate real estate offers, an action state "user selects offer".
  - Concerning the sequence diagram: In our design, we have to intensify the representation of the messages between the different objects.
  - Another thing: We should think about the addition of a display controller, because the user interface has no information on the properties of the display device. A display controller would guarantee the correct representation of our interface on diverse systems.

Chapter 9

The 8^{th} Lesson

9.1 The Assignment

Complete the personal papers and send them in until Wednesday, December 19^{th}, 12am.

This is hereby done.