November 14, 2001

Proseminar Software Engineering

ImmoWarrior

Report

Fimberger Lucia
redwing@utanet.at

Nidetzky Marion
mnidetzk@cosy.sbg.ac.at

Rescher Robert
rrescher@cosy.sbg.sc.at

Strobl Elisabeth
estrobl@cosy.sbg.ac.at

Schobesberger Oskar
oschobes@cosy.sbg.ac.at

Academic Supervisor Mag. Dipl.-Ing. Schwaiger Roland

Department of Computer Science
University of Salzburg

Correspondence to:

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

Chapter 1

Introduction

1.1 Terms of reference

A real-estate auction firm $e - Immo$ wants a new software for processing the collection of information and of the representation of information. This processing should be simple. The users of this system are the collector of information and the inquirer, who can be a person or another system. One demand is, that the system is simple to service, by meaning, it should not be necessary to install the collection- and representation-components local. Furtheron, $e - Immo$ should have the possibility to influence the processing of information for individuals.

1.2 Development evolution

1. Demands on the system
2. CRC cards

Chapter 2

Problem-description

2.1 Demands to the system

- Collecting information (in our case real-estate).
- Possibility to enquiry information.
- Real-estate collection.
- No local installation of components.
- User has to be identified.
- Different access rights.
- Processing of representation.
- Auction is online.
- User's administration.

Chapter 3

CRC cards

3.1 Short introduction into CRC

Founder is Beck Cunningham (ca. 1989).
A *Class – Responsibility – Collaboration – Session* concentrates on modelling a CRC-model. A CRC-model defines a class for each part of a problem that should be solved. A class consists of its responsibilities, which are called the *properties* and the *methods*. A property is a non-operatable attribut, it is always an instance of another data type, which can also be a complex type or a class. The class-to-class relation is called collaboration. A method is an operating routine that can be called.
CRC-models are made for abstracting the reality by grouping functional units. These functional units communicate with each other. Usually, CRC-models have hierachical structures. It's not usual that two classes communicate circular with each other.
The members of a CRC-session are a project leader, some collaborators and perhaps a supervisor. The CRC concept seperates a problem into several classes, which has the advantage that developers do not need to know the functional structure of the whole, but of only one class. Finally, the classes commuicate with each other via defined interfaces, and it's equal to the calling class, how the rest of the called class is structured and implemented.

3.2 Each individual CRC card

1. <u>real-estate</u>

   responsibilities — collaboration

   - knows its characteristics

2. <u>real-estate collection</u>

   responsibilities — collaboration

   - knows the real-estates .... real-estate
   - edits real-estates .... recorder
   - returns real-estates .... enquirer

3. <u>user</u>

   responsibilities | collaboration

   - knows its characteristics

4. <u>user collection</u>

   responsibilities | collaboration

   - knows users and their access rights .... user, access rights
   - edits users and their access rights
   - passes information about user and his rights .... system interface

5. <u>access rights</u>

   responsibilities | collaboration

   - knows its characteristics

6. <u>system interface</u>

   responsibilities | collaboration

   - offers login .... representation processor
   - checks login data .... user collection
   - enables request or record of data .... representation processor
   - transmits data .... recorder, enquirer

7. <u>representation processor</u>

   responsibilities | collaboration

   - receives preprocessed data .... enquirer
   - passes query related real-estates .... terminal
   - displays masks for login, input and output

8. <u>terminal</u>

   responsibilities | collaboration

   - receives data .... representation processor
   - displays data

9. <u>enquirer</u>

   responsibilities | collaboration

   - requests .... real-estate collection

- gets required information about real-estates .... real-estate collection
- returns information .... representation processor

10. recorder

    responsibilities collaboration

    - makes possibility for input available .... represenation processor
    - returns completely collected real-estates .... real-estate collection
    - accepts information about real-estates

Appendix A

Protocols

A.1 2001-31-October

1. Person responsible for protocol
   Rescher Robert Harald

2. Participating persons
   Mag. Dipl. Ing. Schwaiger Roland, Rescher Robert, Strobl Elisabeth, Nidetzky Marion, Fimberger Lucia, Schobesberger Oskar

3. Topics talked about
   What is a CRC-Session?
   Presentation of our CRC cards.

4. New assignments
   We need new classes to refine our application.

5. Open questions
   Is the auction online?
   Is there any restriction?
   Should the user be identified?

First of all, the application was built. Therefore, we declared a class named **managing system**, which was our main operating application unit. This unit encapsulates the login managing system as well as the read- and -write-methods to access the database class.
The **database** class is the highest level in the hierarchy. It consists of the data pool and the methods to post and enquiry data. The database encapsulates the $real - estates$, $users$, $login - role$ and $user - rights$. Here are the contents of the classes:

- **real-estates**: This class contains all the necessary entities of the real-estates to be stored.
- **users**: This class contains all the user information.
- **login-role**: This class contains the access rights.

- **user-rights**: This class joins the user class with the login-role class.

These classes are part of the database, so they do not need functional routines. For logging in, there will be responsible another class that contains the functional routines, and another one for the user interface.
Finally, we constructed a user interface for managing the classes of the database, since the managing system does not distinguish between a user or a further instance of itself when communicating. So, there is a class **user interface**, which calls the methods of the managing system, and is also responsible for the display and input/output to communicate with a human user. The user interface is best described with a *user interface prototype*.

A.2 2001-07-November

1. Person responsible for protocol
   Rescher Robert Harald

2. Participating persons
   Mag. Dipl. Ing. Schwaiger Roland, Rescher Robert, Strobl Elisabeth, Nidetzky Marion, Fimberger Lucia, Schobesberger Oskar

3. Topics talked about
   Presentation of refined CRC cards.

4. New assignments
   Auction is online.
   User interface prototype.
   Restriction on complete object via onion-model.
   Other companies can input an offer.
   Perhaps a special offer: Display offers close to logged in user's area.

The next step will be to refine the user interface.
Therefore, we renamed the managing system to **system interface**, since it is better to understand with this terminology.
We have to distinguish at least these levels in our problem: The database level containing the real-estates and the one with the entities for user logins. Both these class groups (four class) can be joined by one class which holds the methods for editing and inquirying the data, or one class for the real-estates and one for the classes that contain the users login information. The second possibility can be used when there is a second system installed in the company that does not distinguish between collaborators and other users.
In combination with the login rights are not only the access rights, but also a set of real-estates that are returned when inquirying. So, the class **real-estates** has a property level. The level assigned to the user defined in the user collection must be at least as great as the level defined in the single instances of the real-estates to make them visible to the user. This filtering is done by the **representation processor**, a class that passes the real-estates valuing them by the **user collection**.
Since another system can communicate with it, until this point there are only passed data, never a display instruction. A class called **representation processor** formulates the data to a user mask, and passes the inpots done by the user to the higher classes.

A further requirement was that there is no local terminal program to be installed. So, the **representation processor** formulates a common format, such as *hypertext*. In the model appears a class called **terminal** which is not part of our application; it is for example a web-browser that understands hypertext.