Part I
Analysis

Well begun is half done
-Aristotle

In this section, we will attempt to give the reader insight into the rationale behind the decisions made in the course of the project, by reflecting on challenges and surprises uncovered by analysing the problem area. In other words, the section will try to explain *why* the system has been designed as it has, why the process has been managed the way it has, and so on.

The section is structured by the various techniques and artefacts used in conducting the analysis. The primary topics covered in this section are:

- Business modelling
- Requirements analysis
- Data modelling
- Architectural analysis

We begin the section by investigating and discussing the activities aimed at uncovering requirements for the system.

1 Requirements Analysis

Generally speaking, requirements can be divided into two categories; functional and non-functional. The functional requirements for the RentIt server are expressed by use-cases, that each encapsulate some functionality the system must posses in order to be a useful solution. The non-functional requirements are captured by the factor-tabel. These artefacts will serve as the basis for the discussion of the requirements in this section.

1.1 Use Cases

As mentioned, the use-cases capture the functional requirements of the system. The list of use cases was composed by examining the business model and identifying user goals associated with uploading, viewing and buying videos, creating user accounts and so on.

When writing use cases for the RentIt system, we noted that they could be grouped into three categories of usage:

- **User management**
  Use cases concerning user profile creation, editing user profiles and authorizing users

- **Video management**
  Use cases concerning downloading, viewing and rating videos
- **Transaction management**
  Use cases concerning transactions between account balances and buying more RentIt credits.

In other words, the use cases reveal at least three major areas of responsibility within the required functionality of the system. These responsibilities imply a large scale organization of namespaces, by way of the GRASP principles.

In addition, the use cases provided some intuition about the control flow of some of the major operations of the system. Take for example user case no. 11, which describes paying for a video:

A user wishes to pay for a video

- Precondition: The user has an active account with sufficient money deposited
- Postcondition: The user has the appropriate amount withdrawn from his account
- Postcondition: The user is granted access to the video

  1. The user finds the video he wants to buy
  2. The user indicates that he wants to buy the video
  3. The user is granted access to the video

While short and simple, the pre- and postconditions as well as the steps involved in the use case, reveal a number responsibilities and changes in control flow to be considered. For example, the precondition indicates that each user has a balance associated with her account, to which deposits are made in a separate use case (no. 9). Furthermore, the following responsibilities need to be delegated to appropriate branches of the system

- withdrawing money from the users account
- granting users access to videos upon successful payment

This insight can be used as inspiration when designing the large scale organization of namespaces, and the control flow between these packages.

Moreover, the use cases proved an effective way of communicating ideas with the SMU group. By comparing our use cases with their use case model, it was possible to reconcile the two groups expectations to the semantic meaning of the different operations of the system. Take for example use case no. 4, dealing with purchasing videos:

A user wants to see a video and downloads it to his computer

- Precondition: User is logged in
- Precondition: User has enough credits to pay the video
- Postcondition: The video is saved on the users computer

- Postcondition: The appropriate amount is subtracted from the users account

    1. Available videos are presented to the user
    2. The user selects the video he wants to see
    3. The user is presented with options to rent and buy
    4. The user selects buy
    5. The user specifies where the file should be saved
    6. The download begins

This use case was utilized as the basis of a discussion between our own and the SMU group, about the semantic meaning of the term "buy", by the end of which it was decided that downloading a video constituted a buy, whereas renting would involve streaming the video, which is also expressed in the use case.

1.2 Factor Table

As opposed to the functional requirements, the non-functional requirements are not expressed by some functionality the system must offer, but rather some behaviour or quality the system must posses. These are captured by a factor table, as described in **Larman**. The list of factors (requirements) the factor table consists of, was devised by using the FURPS+ mnemonic, and brainstorming on predictable requirements based on the business model. This facilitated a fruitful discussion on topics such as security, reliability and performance requirements of the system. The factor table can be found below:

Factor	Measures and Quality Scenarios	Variability	Impact of factor	Priority	Difficulty
Security					
Secure communication with credit card services	Secure system against eavesdropping, man-in-the-middle and sniffing attacks. Tested by simulation.	Not likely to change	Customers are not likely to use a service that compromises their credit card information. Also, the service provider (us) could be made liable	H	M
user information security	Prevent password theft and phishing attacks by securing communication. Tested by simulation.	Not likely to change	Customers are not likely to use a service that compromises their credit card information. Also, the service provider could be made liable	H	M
protection of content	The business model is based on users' ability to sell access to their uploaded content. Thus,	Not likely to change	users dislike having their content stolen	H	M
Reliability					
user access to content in case of server failure	Provide access to purchased content when the server is down, or provide fault detection so users can have their rent period extended accordingly or be reimburged. Test by simulation	Videos may be cached locally	high priority to customers	M	L
Persistence of user data	All data should persist in case of server crash. Test by ensuring data persistence between runs	Not likely to change	crucial for business model	H	L
Performance					
limiting client video buffer wait time	No buffering time for a 300< Mb video on a client with 10MBit download bandwidth. (Perhaps smooth streaming?)	Not likely to change	Greatly impacts the user experience.	M	M
limiting server search time	Maximum of 5 sec round trip of search query to search result on the client side under normal(?) server load	Not likely to change	Greatly impacts the user experience.	M	M
Accuracy of search results	Test search results match specification. How is search performed with regards to tags. categories, descriptions and titles?	Not likely to change	Greatly impacts the user experience.	M	M
Supportability					
Adaptability towards credit card services	Interfacing with new credit card provider services should be easy to implement. Goal of 10 person days for implementation	Credit card services are fairly well established and do not change all that often	Of interest to developers only, but can enhance business sustainability	M	L
Adaptability towards web video streaming technologies	Service consumers are likely to implement clients using standard web technologies, and service should be able to adapt to changing web technologies. Adding web standard streaming capabilities to the serices interface should take 30 person days	Service will do fine to offer HTML5 streaming capabilites and perhaps IIS smooth streaming capabilities for now.	Of importance to client developers	L	M
Internationalization of currency	Secure adaptability towards displaying different currencies.	not likelily to change		L	M
Usability					
Documentation of web service interface	Provide human means for discovering service interface, both through metadata exchange and human readable documentation	not likely to change		H	L
Legal					
Who owns the videos???					

Some of the usefulness of the factor table is its recording of quality scenarios, because it encouraged the group to reflect on the quantifying the requirement, as to make it testable. Realistically, it would not be possible to satisfactory implement and test all of the factors, but for the sake of exercise, the factor

table was composed as if it was meant to be used in real world situation. As a consequence of the limited timeframe of the project, we decided to limit the factors that we would *actually* emphasize to:

- accuracy of search results
- documentation of web service interface
- persistence of user data

These factors were consensually understood as the most crucial and/or interesting requirements to pursue.

2 Data Modelling

Another major part of our problem analysis was devising a model of the data, partly in order to construct and refine a database design for persisting the necessary data, and in part to serve for inspiration for C# classes. This modelling was done using two artefacts, a domain to serve as a visual dictionary for objects and concepts in the problem domain, and an ER-diagram as an aid for reflecting on a useful database design. The main inspiration for these models were the use cases and the business model. These two artefacts serve the basis for discussions in this section.

2.1 Domain Model

The domain model was used to facilitate and document the results of a discussion of the problem domain. By modelling the objects and concepts in the problem domain, the project group came upon a number of questions, such as:

-

2.2 ER-Diagram

3 Architectural Analysis

In this section

3.1 Package Diagram

4 Project Management Plan