Project Ivanhoe

Group 16
Rebecca Everett
Tobias Hassellöf
Henrik Törnvall
Johan Renner
Martin Waara-Grape

Table of Contents

1. Preface

1.1 Readership of document

The intended readers of this document are the group of people involved in designing, developing, testing and managing the release of this program. These also include the teachers and other students of this course (DD1363). This and all future appendices will be written in English.

1.2 Document history

- 2007-11-29 – The first draft of this document.

2. Introduction

2.1 Need of the system

In today's environment, there are many different ways to spend money. You can order via the Internet and receive an invoice, pay cash, debit your personal account or use a credit card. This makes it hard for ordinary people to keep track of their expenses and many struggle with debts that never get paid. In order to cope with unexpected outlays as well as saving for planned activities such as the family's vacation, it is essential to have a good overview of one's economy.

2.2 System functions

Project Ivanhoe is a software project for developing a Home Finance System (HFS) that lets the users input their daily expenses and expenditures. This in turn lets the users get a practical overview of his or her economical situation to make financial decisions. The users of the system are people with a home computer and a need for monitoring their personal finances. He/she is a person with income and expenses, who wants to be able to manage the money spent and earned in a more effective way. The system is designed for everyday use as well as special occasions. Some main functions of the program are:

- To create budgets
- To monitor income and expenses.
- To provide a graphical overview of the personal or family economy.
- To compare expenses with the budget.
- To monitor expenses, such as bills, with a reminder function

2.3 Other systems

The HFS will store all information on the computer and there will be no interaction with other services or systems. In order to use the system on a different computer, the information will need to be copied or re-entered. The system cannot manage the same profile on more than one computer since the program is not designed to synchronise with other systems and the information is stored on the computer only.

3. Glossary

Budget	A series of economic items planned for future use.
Due date	A date a bill is supposed to be paid.
Economic item	An income or expense that a user wants to record or has recorded.
HFS	Home Financing System – A generic name for a type of application that allows user's to manage home finances.
Java	A programming language developed by Sun Microsystems.
JRE	Java Runtime Environment. A piece of software that allows Java applications to run on a computer.
Minimal guarantees	What the system delivers no matter what happens in a use case.
PC	Personal Computer.
Pre-condition	A condition which must be fulfilled before the start of a use case.
Primary Actor	The actor carrying out the main operation in a use case.
Profile	A container for user information.
Stakeholder	A person with an interest in the outcome of a use case.
Success guarantee	What the system delivers if the use case is executed successfully.
Trigger	An action that starts a use case.
Use Case	A series of actions carried out by actors to reach a specific goal in the system.
User	The person who uses the system.
XML	The Extensible Markup Language (XML) is a general-purpose markup language. Its primary purpose is to facilitate the sharing of structured data across different information systems.
WindowsXP	An operating system used on a PC.

4. User requirements definition

4.1 Functional requirements

4.1.1. High priority requirements

4.1.1.1. Multiple users

The system shall provide support for several users. The system shall separate data for different users from each other.

Rationale: If several users use the system, they would not want their data mixed up with each other.

4.1.1.2. Create budget

The user shall be able to create budgets, where she can record planned income and expenses. The budgets shall be able to be constructed for an arbitrary time period.

Rationale: Budgeting is a corner-stone in managing finances. As for the arbitrary time period, the user might want to create yearly, monthly or even weekly budgets.

4.1.1.3. Manage budget

A budget shall be able to be updated by the user at any time. It shall be possible to add income/expense items and/or change the name of the budget.

Rationale: The user may not want to add all their bills at the same time; therefore it is important that the budget can be updated continuously.

Dependencies: "Create budget".

4.1.1.4. Record economic item

The user shall be able to record a single income/expense item at any time. This item shall be separate from the budget, meaning that it is an actual economic item/event (something that really happened) and not a planned one.

Rationale: The user probably wants to record her spending in order to see if it follows the budget.

4.1.1.5. Income/Expense categories

The user shall be able categorize income/expense items, either within a budget or as a separate entity. The system shall be able to provide a number of predefined categories as well as allow the user to define new categories.

Rationale: Categorizing items as for example "Entertainment" or "Car expenses" allows for an easier overview of one's finances.

The pre-defined categories will make the categorizing faster for the novice.

Dependencies: "Manage budget", "Record economic item".

4.1.1.6. Comparing budget with actual expenses

The system shall provide a facility for comparing income/expenses in a certain budget with actual income/expenses recorded by the user for the same budget period. This shall be an active comparison, meaning it shall always be up to date.

Rationale: This is the central functionality of the system. This function will allow the user to see if she is on-track with the budget. Since it is central, making it an active comparison will reduce the effort needed by the user to use the program.

Dependencies: "Manage budget", "Record economic item".

4.1.1.7. Flexible comparison

The system shall provide a facility for comparing income/expenses for different time periods. This shall be a passive comparison, meaning that it is the user's responsibility to activate the function and to define which time periods and which income/expense items to compare. The income/expense items can be either budgeted or an actual recording.

Rationale: This function will help the user better understand how her expenses vary over time.

Dependencies: "Manage budget", "Record economic item".

4.1.1.7.1. The system shall provide a facility to present the comparison graphically in a diagram.

4.1.1.7.2. The system shall provide a facility to present the comparison in a tabular form.

4.1.2. Low priority requirements

4.1.2.1. Due-dates

The user should be able to assign a payment due-date to each expense item in a budget.

Rationale: This is to allow the user to get an overview of when bills need to be paid.

Dependencies: "Record economic item".

4.1.2.2. Due-date reminder

The system should provide a reminder function for due-dates of an expense item in a budget. This reminder shall be automatically activated each time the user starts the program.

Rationale: If a user has forgotten to pay a bill, the system will alert her of that.

Dependencies: "Due-dates".

4.1.2.3. To-do list

The user shall upon request be provided with a list of activities that need to be performed in the near future. This list shall contain activities that has a due-date within the specified budget period.

Rationale: People are forgetful and need to externalize data in order to not forget it. An example of a list item could be "Pay rent to XXX at the latest 2007-12-31".

Dependencies: "Record economic item", "Manage budget".

4.2 Non-functional requirements

4.2.1. Learning curve

The user shall be able to operate the system after 20 minutes of training.

4.2.2. Calculations

The user shall be able to trust the calculations of the program.

4.2.3. System start-up

The user shall be able to start the system within 10 seconds.

4.2.4. Simultaneous users

The user shall only be able to support one user at a time.

4.2.5. Environment

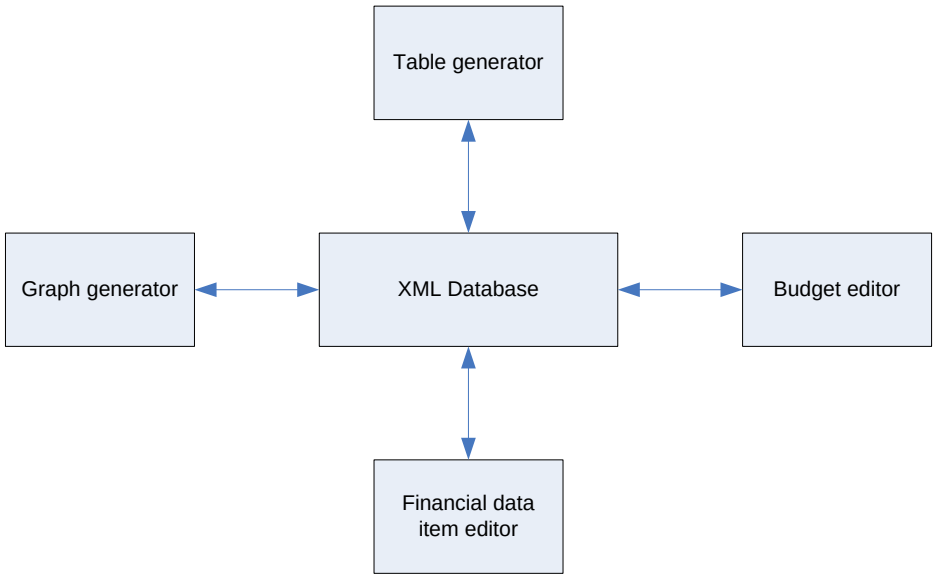
The user shall be able to run the application on Windows XP using Java Runtime Environment JRE 1.5.

4.2.6. Help

The system should provide a help file that the user can reference at any time. This help file should contain simple instructions for all the tasks a user can perform within the system.

5. System architecture

The system is based on the repository model, wherein all shared data is held in a central database that can be accessed by all appropriate sub-systems.



Explanations:

1. XML database: All data is held in this project repository as XML data.
2. Budget editor: This is the function that will manage budget data stored in the project repository.
3. Table generator: This is the function that will generate tables from the financial data in the repository.
4. Graph generator: This is the function that will generate graphs from the financial data in the repository.
5. Financial data item editor: This is the function that will manage the financial data items stored in the repository.

6. System requirements specification

6.1 Functional requirements

6.1.1. High priority requirements

6.1.1.1. Multiple users

Different user's data shall be separated from each other and grouped in one place that is unique to a specific user.

6.1.1.2. Create budget

The budget shall be a collection of data items, clustered together under a heading. The heading shall be a unique name chosen by the user.

6.1.1.2.1. The system shall allow the user to record data items in the budget in a structured manner. The data items should be separate entities that represent income and expenses.

6.1.1.2.1.1. Each data item shall consist of a tagged line of information. The line of information shall consist of a unique (within that budget) identifier (ITEM_ID), a text field ("category"), a real number ("amount") and a free-text field ("notes").

Rationale: The tag will categorize an information line as either an income or an expense, making it easy for the user and the system to separate the two. The "category" field refers to the user requirement "Income/expense categories". The "notes" field will allow the user to write any additional information associated with the data item.

6.1.1.2.1.2. The system shall tag the data items recorded by the user as either "income" or "expense" items. It is the user's responsibility to classify the data as either income or expense.

6.1.1.2.1.3. The system shall control the validity of the "amount" field. It is the system's responsibility to assign and validate the sign of the numbers in this field. Income-tagged lines must have a positive real number in the "amount" field and expense-tagged lines items must have negative real numbers.

6.1.1.3. Manage budget

It shall be possible to delete data items in a budget, as well as add data items and change the budget name. The adding of data

items shall function in the same way as described in the “Create budget” points.

6.1.1.3.1. The deletion of items and changing of the budget name shall be “one-click” operations by the user. The deleted items shall be identifiable by their ITEM_ID and erasable from under the correct “budget”.

Dependencies: ”Create budget”.

6.1.1.4. Record economic item

Each single economic item shall be identifiable by a unique identifier (ITEM_ID). The item shall be constructed in the same way as the items described under the “Create budget” points.

6.1.1.4.1. The item shall be marked with a date. The user shall be responsible for setting the date.

Rationale: In order to satisfy the “Comparing budget with actual expenses” and “Flexible comparison” requirements, the program must know which items belong to which time-periods.

6.1.1.5. Income/expense categories

The system shall provide a number of pre-defined categories for data items, as well as allow new categories to be defined by the user. The categories shall be a text-field in the information line described under the “Create budget” points.

Dependencies: “Create budget”.

6.1.1.6. Comparing budget with actual expenses

The system shall update the comparison (difference) each time the user records another data item. The comparison data shall not be stored, but automatically recalculated each time the user starts the program or records another data item.

Dependencies: “Create budget”, “Record economic item”.

6.1.1.6.1. The comparison *should* be visible to the user in the main view of the program.

6.1.1.6.2. The comparison shall be made for the defined time-period of an active budget and actual recorded items that have dates coinciding with the budget time-period.

6.1.1.6.2.1. Comparisons should be possible to make for whole categories of items as well as individual items. The system should make the comparison based on the “category” information in the information line for a data

item. It should also identify if two items have the same ITEM_ID and compare those two items.

Rationale: If the user has entered for example an item like entertainment expenses in a budget, and later recorded the actual expenditure for the entertainment, he has probably entered the same ITEM_ID for those two different items. This makes for an easy comparison between the two.

6.1.1.7. Flexible comparison

The system shall be able to compare items based on criteria specified by the user. This shall be done upon request from the user.

Dependencies: "Record economic item".

6.1.1.7.1. The system shall use the date-marks of the items to identify the correct items to compare for the correct time-period.

The user shall be responsible for specifying which time period she wants to use for the comparison.

6.1.1.7.2. The system shall identify different data items and different categories of items and make a comparison between them.

The data items shall be identified by their ITEM_IDs and the categories shall be identified by their "category" tag.

6.1.1.7.3. The system shall be able to generate graphs and diagrams based on the comparison criteria specified by the user.

The data to be presented in the graph shall be obtained as per the previous points.

6.1.1.7.4. The system shall be able to generate a table based on the comparison criteria specified by the user.

The data to be presented in the table shall be obtained as per the previous points.

6.1.2. Low priority requirements.

6.1.2.1. Due-dates

The due-date should be a field in the before mentioned information line of a data item. Setting a due-date should be the user's responsibility.

Dependencies: "Record economic item".

6.1.2.2. Due-date reminder

The system should automatically be able to produce a reminder.

Dependencies: "Due-dates".

6.1.2.2.1. The system should automatically upon start-up of the program go through the active user's data items and find any due-dates that are active. The system should then produce reminder that will be automatically visible to the user.

6.1.2.2.2. An expense item should be able to be marked if handled. This should be the user's responsibility to perform.

Rationale: This will help the system determine which due-dates need to be reminded of.

6.1.2.3. To-do list

The system should go through the user's data and find any due-dates that are active. This shall be done upon request from the user.

Dependencies: "Record economic item", "Manage budget".

6.1.2.3.1. The system should then produce a list containing all items found in the search. This list shall be automatically visible to the user.

6.2 Non-functional requirements

6.2.1. Learning curve

The system shall be easy to learn and provide a help document to make it possible to operate the system after 20 minutes of training

6.2.2. Calculations

6.2.2.1. The system shall calculate correctly.

6.2.2.2. The system shall detect irregularities such as negative amounts or budgets deficits.

6.2.3. System start-up

The system shall give interface control to the user within 10 seconds of the program being started.

6.2.4. Simultaneous users

The system shall only be able to run with one active user.

6.2.5. Environment

The system programming language shall be Java.

7. System evolution

7.1 Overall assumptions

The system is dependent on a PC platform running windows XP. The system does not communicate with any external components, and so the only change we have to fear is that the PC platform will become obsolete, which is not probable.

7.2 Java evolution

The software is written in java. Java is a portable format and minor changes can be made to fit other platforms if required. Java is also upwards compatible with newer versions, and so the code will not have to be changed to fit newer versions of the JRE.

7.3 Changing user needs

The reason this program is being made is that people have a hard time managing their money. This problem is not likely to go away with time, but rather escalate as the way of spending and acquiring money will probably increase. The user might then need more monitoring functions, which can be added to the program either in a new version or as external plug-ins.

The user might want to use an automation tool for paying bills online, an also synchronise their spending plans with their bank statement. The program should in time be able to do this, but more extensive knowledge from the project members is required to implement functions like these.

8. Appendices

8.1 Use Case UC1 – Record expense

Primary Actor:
Student
Stakeholders and Interests:
- Student: Wants to be able to record an expense in his/her profile. The process should be intuitive and there should be a possibility to cancel it. The student wants to be able to add a note to the expense and categorise it as a certain expense category.
Preconditions:
The system has been started. The student has a profile.
Minimal Guarantee:
Only the information saved by the user will be saved to the profile. There should be no ambiguities concerning what has been saved or not.
Success Guarantee:
The expense is registered on the profile with a note and has been categorised.
Trigger:
The student sits down at his/her computer with an expense that he/she wants to record in his/her profile.
Main Success Scenario:

1. Student chooses his/her profile.
2. Student chooses to record the expense.
3. Student inputs the amount of the expense.
4. Student chooses a category for the expense.
5. Student inputs a short note explaining the expense.
6. Student inputs the date of the expense.

The student repeats 4-7 until all the expenses has been inputted.

7. Student chooses that he/she is done.
8. The expenses are saved by the System to the current profile.

Extensions:
*a. At any time, System fails:
If the system fails and the program shuts down then only the saved data will be available. If the student hasn't chosen that he/she is done, then the information is lost.
3a. Student inputs a negative amount of the expense.
    1. System rejects the amount and registers the absolute value.
4a. The category that the Student wants to use does not exist.
    1. Student adds another category.
    2. Student chooses the new category for the expense.
4b. Student doesn't know what category to use.
    1. Student chooses miscellaneous, which is a predefined category.
    2. Student decides to change the category at a later time.
5a. Student chooses not to enter a note for the expense.
7a. Student decides to cancel the input.
    1. The recorded data is discarded by the System.
8a. Student hasn't inputted an amount and /or chosen a category for one or more expenses.
    1. System informs the student that there is information missing.
        a. Student chooses cancel.

i. The information is discarded by the system.

b. Student inputs the missing information.

i. The student chooses that he/she is done.

ii. The expenses are automatically saved by the System to the current profile.

8.2 Use Case UC2 – Creating a profile

Primary actor:
Parent in the family
Stakeholders and Interests:
Parent – wants to create a profile for the family in the system so that he/she is able to analyse the family's economy. If the profile already exists, he/she wants to be made aware of this with a possibility to cancel or overwrite the profile.
Members of the family – wants the parent to be able to analyse the family's economy.
Preconditions:
The system has been started.
Minimal Guarantees:
Only a profile saved by the parent will be available in the system. There should be no ambiguities concerning what has been saved or not.
Success Guarantees:
A user profile has been created and saved.
Trigger:
Parent sits down at the computer and wants to cerate a profile in the system.
Main Success Scenario:

1. Parent chooses to create a new profile.
2. Parent inputs the name of the new profile.
3. A new profile is created and saved by the System.

Extensions:
*a. At any time, System fails:
If the system fails and the program shuts down, only the saved data will be available. If the parent hasn't saved the profile, then the information is lost and a new profile has not been created.
2a. The name of the profile already exists.

1. Parent chooses not to overwrite the profile.
    a. Parent inputs a new name for the profile.
2. Parent chooses to overwrite the profile.
    a. The information saved on the previous profile is discarded by the System.
3. Parent chooses to cancel the process.

8.3 Use Case UC3 – Create budget

Primary actor:
Parent in the family
Stakeholder and Interests:
Parent - Wants to create a budget for his/her family in the family's profile.
Members of the family – wants the parent to be able to monitor the economy with a budget.
Preconditions:
The system has been started. There are one or more known incomes and expenses.
The family's profile has been selected.
Minimal Guarantee:
Only a budget saved by the parent will be available in the system. There should be no ambiguities concerning what has been saved or not.
Success Guarantee:
A budget has been created and saved to the family's profile.
Main Success Scenario:

9. Parent chooses to create a new budget.
10. Parent inputs a name for the budget.
11. Parent chooses a time period for the budget.
12. Parent inputs expected income or expense.
13. Parent chooses a category for the economic event.

Parent repeats step 4-5 until satisfied.

14. Parent chooses that he/she is done.
15. The budget is saved by the System to the family's profile.

Extensions:
*a. At any time, System fails:
If the system fails and the program shuts down, only the saved data will be available. If the parent hasn't saved the budget, then the information is lost and a new budget has not been created.
2a. The name of the budget already exists.

4. Parent chooses not to overwrite the budget.
    a. Parent inputs a new name for the budget.
5. Parent chooses to overwrite the budget.
    a. The information saved on the previous budget is discarded by the System.
6. Parent chooses to cancel the process

4a. Parent inputs a negative amount of the economic event.

1 System rejects the amount and registers the absolute value.

7a. The category that the parent wants to use does not exist.

1 Parent adds another category.
2 Parent chooses this category for the economic event.

6a. The sum of the incomes doesn't cover the budgeted expenses.

1 System alerts the user of the incorrect budget
    a. Parent chooses not to change the budget.
    b. Parent chooses to alter the budget.
        i. Parent changes the economic events.
        ii. Parent chooses that he/she is done.

8.4 Use Case UC4 – Compare budget with cash flow

Primary actor:
Student
Stakeholder and Interests:
Student - wants to compare a budget with the month's cash flow.
Preconditions:
The system has been started. A budget has already been created and the expenses and incomes of the month are inputted. All data exists in the student's profile.
Minimal guarantee:
The previously saved information should not be lost. There should be no ambiguities concerning what has been saved or not.
Success Guarantee:
Student has been able to compare the cash flow from the previous month with the budgeted amounts.
Trigger:
Student sits down at his/her computer and wants to compare his/her budget with the month's cash flow.
Main Success Scenario:

1. Student chooses his profile.
2. Student chooses to analyse the cash flow.
3. Student chooses a budget.
4. Student chooses the time period to analyse.
5. Student chooses how the information should be presented by the System.
6. System presents the information.
7. Student compares the budget with the cash flow and analyses his/her budget.

Extensions:
*a. At any time, System fails:
If the system fails and the program shuts down, the previously saved data will not be lost.
*b At any time, Parent chooses to cancel the process:
System ends the process.
4a. The father doesn't choose a time period.

1. The system uses the default value that is the previous month.

9. Index