

Program Finder

Project report

Rocco Santese

Tomislav Tvrđić

Contents

- 1. Description of the project.....3
- 2. Mobile user analysis.....4
- 3. Scenario analysis.....6
 - 3.1. Usage Analysis.....6
 - 3.2. Screen and Interaction Analysis.....7
 - 3.3. Environment Analysis.....8
- 4. Architectural design.....9
 - 4.1. Architecture.....9
 - 4.2. Subsystems.....10
 - 4.3. Challenges and benefits.....11
 - 4.4. Cost Analysis.....12
- 5. Navigation and user interface design.....13
 - 5.1. User interface analysis.....13
 - 5.2. Interaction optimization.....15
- 6. Testing.....16
- 7. Code structure.....18
- 8. Screenshots.....20

1. Description of the project

Our project, the program finder, is designed so that the user is able to search for a desired program in a program list. The user can browse detailed information about the program, and select the desired program. The selection is sent to a contact from the device's phonebook via SMS. The contact, upon receipt of the proposed program, can decide whether to accept or reject suggestion, or make a counter proposal.

The technical details of implementation will be explained in latter chapters.

2. Mobile user analysis

In this chapter of the report the target audience for the application is identified. We also need to identify the special requirements or functionalities that some of the users of our application may require.

In order to find the target audience for the TV program selector, we have to consider the usability of the system. Since the application is intended to be simple and have a user friendly interface, the target audience is very broad, since the application can be used by a variety of people, who do not have to have a technical background or not even a high level of education.

In order to collect some information about the target audience, the special requirements or functionalities they might require, and the opinion about some proposed functionalities, we have decided to make a survey amongst 10 persons, age ranging from 20 to 26, with sexes represented equally (5 males and 5 females).

The survey was divided in three parts:

First part regards the criteria of search of the desired TV program, and the parameters used for the search. Our proposals are: for the main selection - type of program (movie, documentary, sports, news and other) and the time span of the program and for the sub selections: genre of the program.

The second part regards the quantity and extent of information regarding a potential candidate program. Our proposal is displaying the time frame, basic plot and description of the program.

The third part regards the proposing procedure. Our proposal is that after the program selection procedure, the phonebook is used to mark the contact to which the proposal will be sent, and it will be sent by sending an automatic generated SMS. The way of notifying the user of incoming request is discussed as well. Our proposal is that the users are simply notified by an SMS message, on which they can respond with a simple click of the accept or reject button, upon which a SMS is sent to notify the proposing user that his proposal has been either rejected or accepted.

The results of the survey showed us that we mostly guessed the wishes of an average mobile application user. Our sample users approved our program selection proposal, with the remark that user can be able to select only one of the filters (time or type of program) in order to perform the search.

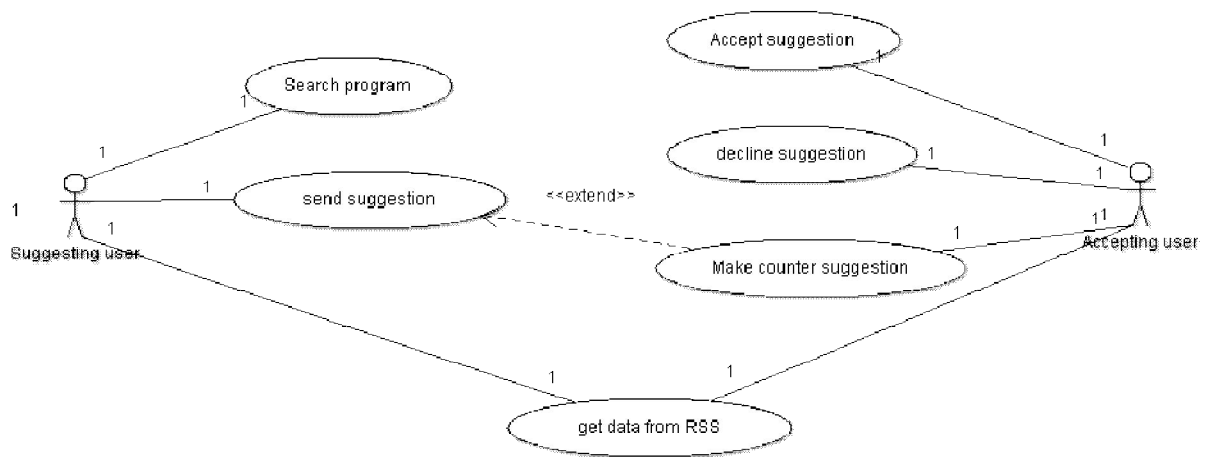
Quantity of information displayed on the screen was satisfactory to our sample users, and a suggestion that more information regarding the time and channel on which the program is displayed were adopted, and included in the program.

The third part was the most contradictory amongst sample users. Some of them were suggesting the creation of a custom SMS, while others were supporting our idea of automatically generated message. Since the majority, 7 out of 10 sample users supported the automatic generation procedure; we decided implementing our communication standard in that way.

3. Scenario analysis

3.1. Usage Analysis

We prepared a use case diagram, illustrating the use cases and interaction of 2 users on different mobile devices



Use case diagram

From the use case diagram, we can observe that there are two actors in the system. The first actor, labeled „Suggesting user” is the user that searches the program, using the criteria specified in the search options panel. When he finds a program that matches his wishes, he sends the suggestion of the program via SMS to the accepting user, which is his contact in phonebook.

Accepting user can be only one in a single interaction. Upon receiving the suggestion, the accepting user can accept the suggestion, decline the suggestion, or make a counter suggestion. If the accepting user

decides to make a counter suggestion, his role is switched to the one of the suggesting user and the suggesting user will become the accepting user upon receiving the SMS.

Both users retrieve program data for the current day from various RSS feeds, parse the information, and store them. The connection between RSS and user side is that first time that day the application is started; the client side downloads information on the mobile device. That way the database is up-to-date and lightweight.

3.2. Screen and Interaction Analysis

In order to analyze the interaction between the user and the mobile application in a clear way, the analysis will be performed for each of the possible screens of the application. More detailed information on the graphic user interface, with screenshots, will be provided in latter chapters.

In the *server connection* screen, the user is asked whether he wants to connect to the server, to check for a fresh daily database.

In the *search filters* screen, the user interacts with the system with use of prepared commands, and is not expected to enter custom information, except for entering time in the start and end time filter.

In the *search results* screen, the user is prompted the titles, time span and channel of the programs which match the search criteria. User can navigate to a program, and select whether to view additional information about the program, or invite one of his contacts to watch the program.

The *phonebook* screen is used to select one of the contacts to which the proposal will be sent. User can navigate through his contacts, and select to which the proposal will be sent.

The *message acceptance* screen is prompted when a proposal is received, and user can select whether to accept or reject the proposal, or to suggest another program, making a counter suggestion.

3.3 Environment Analysis

The user mobile device interacts with the other mobile device(s) via SMS. Such interaction is used when making proposals to other clients, or receiving proposals from a client.

The user mobile device interacts with the server to obtain information about the programs. The server side contains a simple xml database of pre-selected RSS feeds, which is updated daily, parses them, deletes the entries from the previous day, and stores the parsed information. The data is sent to the user application the first time user starts the application that day, and stored in the record store. That way the database is refreshed daily, and the size of the database is kept to a minimum.

4. Architectural design

4.1. Architecture

In this chapter we will analyze the functionalities of the system and the underlying architecture.

The first option displayed to the user is if he wants to connect to the server. If he does so, a thread is created which performs a date check, to make sure that the local database is up-to-date. If the date of the xml file on the server is newer than the date in the database, the parsing procedure is initiated. If that is not the case, the filter selection screen is displayed.

The parsing is done by first find a list of TV programs from the web via RSS feeds (XML format). The wireless internet connection is used, which will, using kxml, parse the xml database containing program information once a day. The parsed information is stored locally, in the record store of the device.

The most important functionality is the search in a TV program list. The search of the program uses filtering by keywords or metadata (genre).

For the selection of type and genre filters of search, we will use a drop-down menu. The filters can be singularly applied or not, to achieve flexible search criteria. After the selection of filters has been performed, the local database is queried, and the results are displayed on the mobile screen.

When a candidate program is found, user is able to browse detailed information on the program (plot, channel, etc.). This information is provided by the database, and is displayed on the screen.

If the candidate program is selected as the suggested program, it is sent to the contacts of the user. A menu provides an „invite to watch“ command, which will take the user to the contacts selection screen. This is done by accessing the phonebook of the mobile device. The user can select one contact. Once the selection is completed, the user can send the proposal. To do so, an automatic SMS message will be generated, containing the title of the program.

The generated SMS message is recognized by the listener and displayed. The receiving user is notified of the suggestion, and can reply with 'accept', 'reject' or make a counter offer. In case of yes or no message is sent back to the sender, and in case of counter offer enter selection procedure is initiated, as described before.

After the choice has been accepted, it is stored in the record store of the proposing user.

4.2. Subsystems

- **Database (Record Store)** – used to store the information on the programs, is connected to the *Search* subsystem when it is queried, and the *connection manager* when it receives program information from RSS feeds. This subsystem includes the parser. Each user has a history database, in which all the accepted proposals are stored. That way the status of the interaction between the peers is stored.

- **Connection manager** – manages the retrieval of the information from the internet, and makes request for sending SMS messages with the suggestions and answers. This subsystem includes server with

XML database connection and SMS message generator and listener. Connected to the *Database* and *Search* subsystems.

- **Search** – handles the adjustment of search filters. Connected to the *GUI* in order to display the selections, connected to the *database subsystem* to retrieve the information and to *Connection manager* in order to notify that the selection has been made, and therefore the SMS message can be sent.

- **GUI** – user interface, used to display the information. Connected to the *search subsystem*, and the *connection manager*. Used to select the programs and send the proposal via SMS to selected contacts.

4.3. Challenges and benefits

Implementation of the subsystems was challenging, because the interaction between them should have been to minimum.

There was a big challenge facing us in implementation of the GUI, because the GUI should be user friendly and simple, but able to implement all the functionalities required. Another challenge was creating a modular system, meaning that the subsystems are interacting well, while keeping functional independence. That way we were able to re-use the subsystems and apply them in another context.

Since the application is simple, the performance is very good, and the robustness is achieved through the division of application on subsystems.

4.4. Cost analysis

The cost of using the service comprises in the brief usage of wireless internet, which is charged depending on the internet service provider and the tariff, and the usage of SMS. The wireless internet is used once a day, for the period of time which is required for the daily program database to be downloaded to the mobile phone. The SMS service is used depending on the user and the number of suggestions and counter suggestions made.

5. Navigation & User interface

In this chapter the design of the user interface will be discussed, and the analysis of usability of the system will be performed. The important issues that will be addressed in the design of the interface are clarity, simplicity, and context.

5.1. User interface analysis

The first screen visible to the user is the *server connection query*. The user is asked whether or not he wants to connect to the server in order to retrieve data (Screenshot 1). The commands available are “yes” and “no”.

The second part of our application is the *search filters selection*. (Screenshot 2) In order to perform a search of the desired program, the user is able to select between content and time filters, whose logical relationship is AND.

The first filter is the time filter. Time filter has two fields, one for the start time, and one for the end time. It is possible to leave one or both fields empty as well, since their default values are set to minimum (start time) and maximum (end time).

The content filters are used to select the type and the genre of program to be searched. It is divided in two parts, the type of program selector and the genre selector. The type of program selector is implemented as a prepared list, shown in a drop down menu. The user is be able to select the desired type of program with

the confirmation click on the filter, scroll through the list with arrows, and another confirmation click on the desired type of program.

The genre selector follows the same principle. The list in the dropdown menu depends on the type of program selected; therefore it is not possible to select the genre of the program before the type of program has been selected.

The commands available on this screen are “history” and “search”.

By selecting the “history” button, user can view his previous choices that were accepted by another party in the *selection history screen* (screenshot 3).

By selecting “search” button, the user starts the search. The search is performed depending on the number of filters selected. The results of the search are displayed in the *results screen* (Screenshot 4).

Search results can be navigated with the arrows, and each entry is selectable.

The commands available are “Select” and “Invite to watch”.

By selecting the “Select” option, some additional information about the program is displayed on the *program additional information screen* (Screenshot 5).

Choosing the “Invite to watch...” option displays the next screen which is the *phonebook* (Screenshot 6), and on which it is possible to select one contact to which the SMS containing the proposal will be sent, by clicking on the “Send...” command.

The *incoming message screen* (Screenshot 7) is used to display information on the proposal, rejection, or acceptance of a program.

Commands available are “Reject”, “Accept” and “Suggest another”.

By selecting the “Suggest another” option, the user is prompted the filter selection screen, which is used to select the counter proposal.

5.2. Interaction optimization

The simplicity of our GUI is proven by the fact that a simple search by using one content filter might be done by 5 clicks. 2 clicks to select the program type, 2 to select the genre, and 1 to start the search. By doing additional 2 clicks, the proposal will be sent to a contact in the phonebook.

The time filter is the only filter in which the search criteria are inputted by the user. The user must type the start and/or end time using the mobile device keyboard, according to the format “hh”. If the format is not respected, an exception is thrown when trying to perform the search and the user is prompted to insert the correct time.

6. Testing

Testing has been performed by an independent tester, i.e. the person that performed the tests was not one of the programmers.

Use case 1: The use case that has been performed in order to test the application was using all the filters to perform a search on the database. One of the titles was selected, and sent to a different phone, chosen from the phonebook. The receiving phone received the notification, and performed a counter suggestion, using only one content filter. The counter proposal was then sent, and the original sender, now a receiver, accepted the proposal.

Results: The observed behavior of the application was satisfactory, since correct queries have been performed, and message interaction was successfully achieved. After the acceptance of the proposal, the acceptance message was received, and the choice of the proposal was stored in the history database.

Conclusion: The application behaved as expected, and no crashes or unexpected behavior occurred. We observed the memory monitor for this use case, since it is the most common use case of our application and we found out that the maximum memory usage of our application was at the parsing time, and it totaled 1,826,284 bytes. The memory monitor activity can be observed in screenshot 8.

Use case 2: Application with a clean database, trying to connect to a server which is not running.

Results: The observed behavior of the application was satisfactory, since the user was warned that the connection could not be established.

Conclusion: The application behaved as expected, and no crashes or unexpected behavior occurred.

Use case 3: Browsing though all programs, without entering any search criteria, and sending it to a peer.

Results: The observed behavior of the application was satisfactory, since all of the programs were displayed, so user could freely browse without applying any filters.

Conclusion: The application behaved as expected, and no crashes or unexpected behavior occurred.

7. Code structure

Class ProgramFinder

- used for displaying and processing information. Contains following methods:

dispParsingWindow() - Displays the warning when XML file is being parsed

displayFilters() - Displays the main Form where the user applies filters, searches for results or looks at the History

displayHistory() - This method displays the History of previously accepted movie watch-invitations

listenSMS() - This method creates a Thread that Listens to incoming SMSs while the application runs

messReceived(TextMessage mess) - Handles incoming SMSs messages and shows properly their meaning to the user

sendSms(String number, String message, int type) - Sends SMSs with properly encoded message

notifyServerMissing() - This method generates an Alert message when the server is not Running or the XML source file is missing

openPhoneBook() - Handles the PhoneBook showing the contacs contained in it or creating an Alert when the PhoneBook is empty

bSearch() - Runs the search of TV programs based on selected filters

displayInfo() - Displays details about the selected TV program

run() - Runs the various Threads required by Networking APIs to avoid deadlocks

Class DocParser

Used to parse information from an external XML database and perform queries and existence checks. Contains following methods:

parse() - Parses the XML containing all data about TV programs

store(String a) - Stores data in RecordStore

search(String g,String t,int sTime, int eTime) - Searches TV programs based on genre,type,Start time and end time

checkDate(String date) - Based on the date of creation of our RecordStore, and on the date of TV guide available in the server side, this methods decides whether to renew the record store or to keep it unchanged

storeChoice(String choice) - Stores the history in the record store

getChoices() - returns the history from record store

8. Screenshots

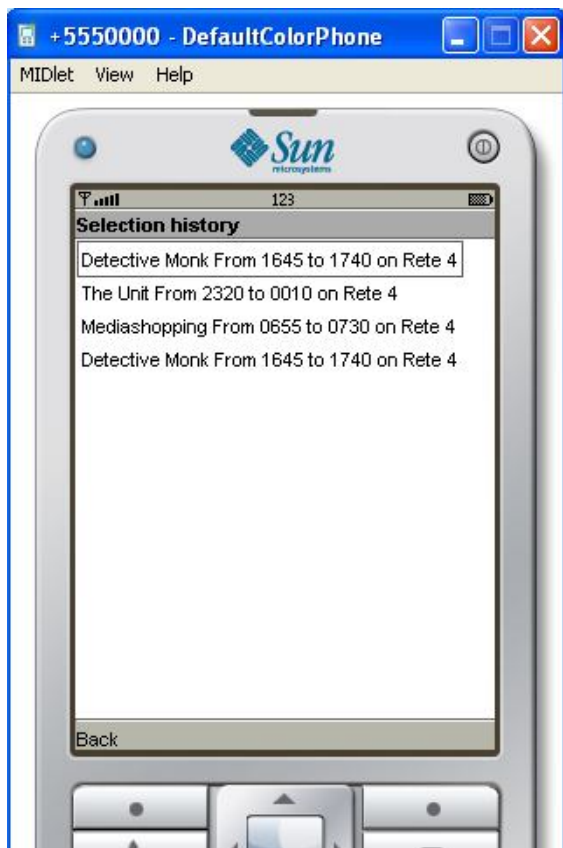
The following screenshots are used to illustrate the different stages of our program.



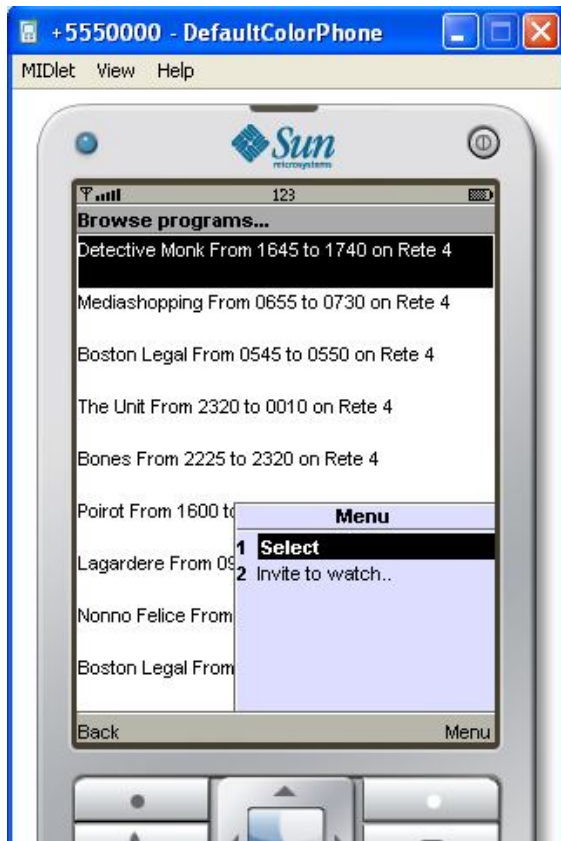
Screenshot 1: the server connection query



Screenshot 2: the search filters selection



Screenshot 3: Selection history screen



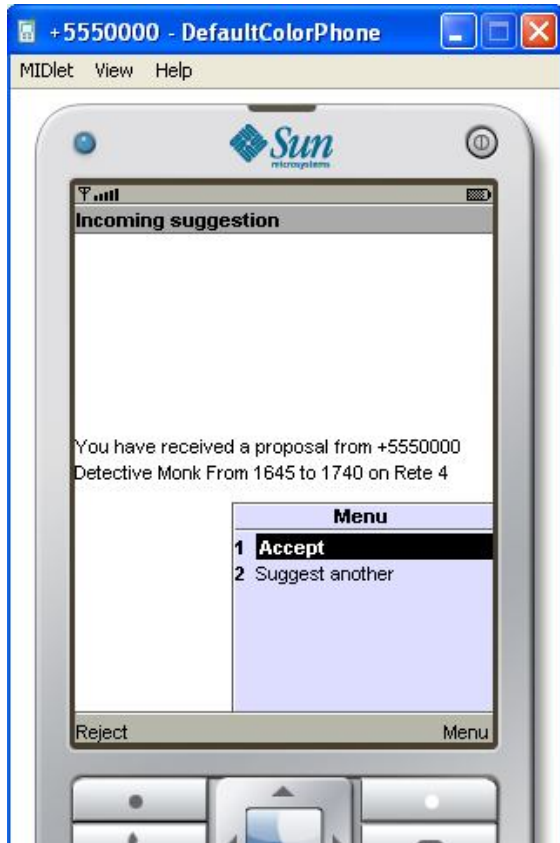
Screenshot 4: Results screen



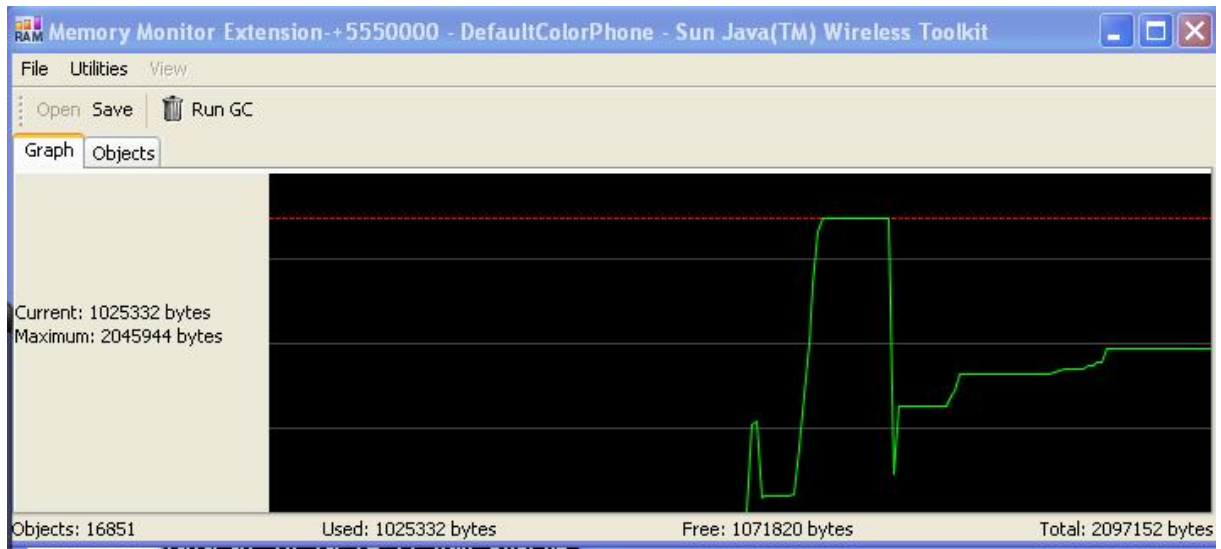
Screenshot 5: Program additional information screen



Screenshot 6: The phonebook



Screenshot 7: Incoming message screen



Screenshot 8: Memory monitor for use case 1