Requirements and Analysis Document

(Note: This is part of the contract between the group and the customer (assistant). May not be changed without agreement form the customer.)

Table of Contents

1. Introduction

1.1 Purpose of System

The purpose of the system is to create a communication framework to enable the user to easily manage information and tasks between different units in the network (e.g. servers and desktop computers). An example is the task to start a computer on the network with your laptop, where you would use the framework to send a wake-on-LAN.

1.2 General Characteristics of System

A network based system that enables communication between a client and a server application which is modular and extendable via a plug-in architecture.

1.3 Scope of the System

Our scope is limited to the creation of a few simple but powerful plug-ins, to enable an interesting presentation of our framework.

Our primary scope does not include the creation of a mobile client nor do we intend to create any advanced plugins.

1.4 Objectives and criteria of the project

a. Our first and foremost objective is to create a server and a client application that can communicate using our framework to send information and executing plug-ins on another client/server.
b. The next objective is to develop a few plug-ins so that we can showcase the possibilities of our framework.

1.5 Definitions, acronyms and abbreviations

- GUI, Graphical user interface.
- Java, platform independent programming language.
- JRE, the Java Run time Environment. Additional software needed to run a Java application.

2. Proposed Application

2.1 Overview

We intend to create a way for computers or other devices in a network to send information to another client or server to perform a task on that client or server.

This would be implemented in a plugin based fashion which would mean that the system would be highly extendable and versatile.
Usage example: Starting a program on another computer, or starting another computer.

2.2 Functional requirements

2.2.1 Client application requirements

1. Connect/disconnect to a server from the client application.
2. Manage plugins.
    a. Install a plugin, both locally and from the server.
    b. Choose options for a given plugin.
    c. Remove a plugin.
    d. Use a plugin.
    e. Manage permissions for accepting commands from plugins on another client.
3. See a list of the plugins on the server thats available for installation.
4. Exit the application, disconnecting from the server

2.2.2 Server application requirements

1. Start and stop the server.
2. Select a port which it listens on.
3. Manage plugins:
    a. Install a plugin.
    b. Choose options for a given plugin.
    c. Remove a plugin.
    d. Manage permissions for accepting commands from plugins from a client.
4. Give a client a list of all the clients currently connected to the server.
5. Give a client a list of all the plugins currently available for installation.
6. Give a client a list of all the plugins currently installed on another client.
7. The server has to be able to connect to multiple clients.
8. Give client ability to send error messages to other clients. (Standard plugin, error message?)

2.3 Non-functional requirements

2.3.1 Usability

Usability is of very high priority both in the server and client applications, a user should practically be able to use the applications directly after installing them without any further education. In short, it needs to be very intuitive. The aim is also to make the construction of plugins as easy as possible but this is still a feature for the more advanced user.

2.3.2 Reliability

Our aim is to make the server as reliable and stable as possible, since this can create a lot of frustration. The main focus on reliability is on the connection part since this is the critical part of the systems functionality.

2.3.3 Performance

The time it takes to connect is more up to the network the user is currently on rater than the system.

2.3.4 Supportability

The main focus in this area is to make the construction of plugins simple. Implementation to support different types of both simple and complex plugins will be place in the API. This means that the plugins should not know about anything else but the most necessary information for it's task.

2.3.5 Implementation

The system will be implemented in the Java environment, this because it is platform independent. If a user want to send or receive information from a plugin this plugin needs to be installed on both the sending and receiving device. Downloads can be made from the connected server if the plugin is installed on it.

2.3.6 Verification

The connections is the most critical part to test and this will be done by checking that right information is sent and received. We will also test that the connection is establish right and that it is stable. Testing of standard plugins will be done but when our work is finished this will not be covered for new plugins.

2.3.7 Packaging and Installing

- Files needed to use the system will be sent in a .zip file, there will be one .zip for the client and on for the server. (containing .jar files)
- All the standard plugins will be place in the server .zip in a dedicated plugin folder.
- GUI will be included in both client and server .zip.
- A README-file will also be included for easy start-up.

2.3.8 Legal

NA

2.4 Application models

This section presents an analysis of the applications domain and it's functionality

2.4.1 Scenarios

The use-cases are relatively easy, no scenarios are necessary.

2.4.2 Use case model

See APPENDIX.

2.4.3 Static model

See APPENDIX.

2.4.4 Dynamic model

See APPENDIX.

2.4.5 User interface

See APPENDIX.

2.5 Possible future directions

- Manage users, creating a black list on the server side.
- Encryption of data that is sent over the network.
- Mobile client application.
- Exstendable plugins, so you can implement functionality from other plugins when creating a new plugin.

2.6 References

NA

3. APPENDIX

3.1 Use cases

3.1.1 Use case: *Open client main window (plugin-window)*

Description

The user wants to open the client main window

Priority
High, this is one of the critical aspects of the program.

Participating actors

- The user
- The client application

Normal flow of events

1. Right-clicks on the system tray icon for the client application and chooses “Main”. (user)
2. The main window is shown. (client)

3.1.2 Use case: *Open client settings window*

Description
The user wants to open the client settings window

Priority
High, this is one of the critical aspects of the program.

Participating actors

- The user
- The client application

Normal flow of events

1. Right-clicks on the system tray icon for the client application and chooses “Settings”. (user)
2. The settings window is shown. (client)

3.1.3 Use case: *Exit client*

Description
The user wants to exit the client

Priority
High, this is one of the critical aspects of the program.

Participating actors

- The user
- The client application

Normal flow of events

3. Right-clicks on the system tray icon for the client application and chooses “Exit”. (user)
4. The client exits. (client)

3.1.4 Use case: *Connecting to server*

Description

The user wants to start a new connection from the client application to the server.

Priority

High, this is one of the critical aspects of the program.

Participating actors

- The user
- The client application

Normal flow of events

1. Right-clicks on the system tray icon for the client application and chooses “Settings”. (user)
2. The settings window is shown. (client)
3. Enters the desired servers IP and port. (user)
4. Clicks the connect button. (user)
5. The text on the connect button changes from “connect” to “disconnect”, and the connection to the server is established. (client)

Exceptional flow - Connection failed

1. Right-clicks on the system tray icon for the client application and chooses “Settings”. (user)
2. The settings window is shown. (client)
3. Enters the desired servers IP and port. (user)
4. Clicks the connect button. (user)
5. A pop-up window appears indicating that the connection failed. (client)
6. Clicks “OK” to close the pop-up (user)

3.1.5 Use case: Send a message using the SendMessagePlugin

Description

The user wants to send a message to the server using the SendMessagePlugin

Priority
Medium

Participating actors
- The user
- The client application
- The server

Normal flow of events
1. Right-clicks on the system tray icon for the client application and chooses “Main”. (user)
2. The Main window is shown. (client)
3. Clicks the SendMessagePlugin tab. (user)
4. Enters a message in the textfield. (user)
5. Clicks send (user)
6. Sends a plugincall to the server with the message (client)
7. Shows a window with the message (server)

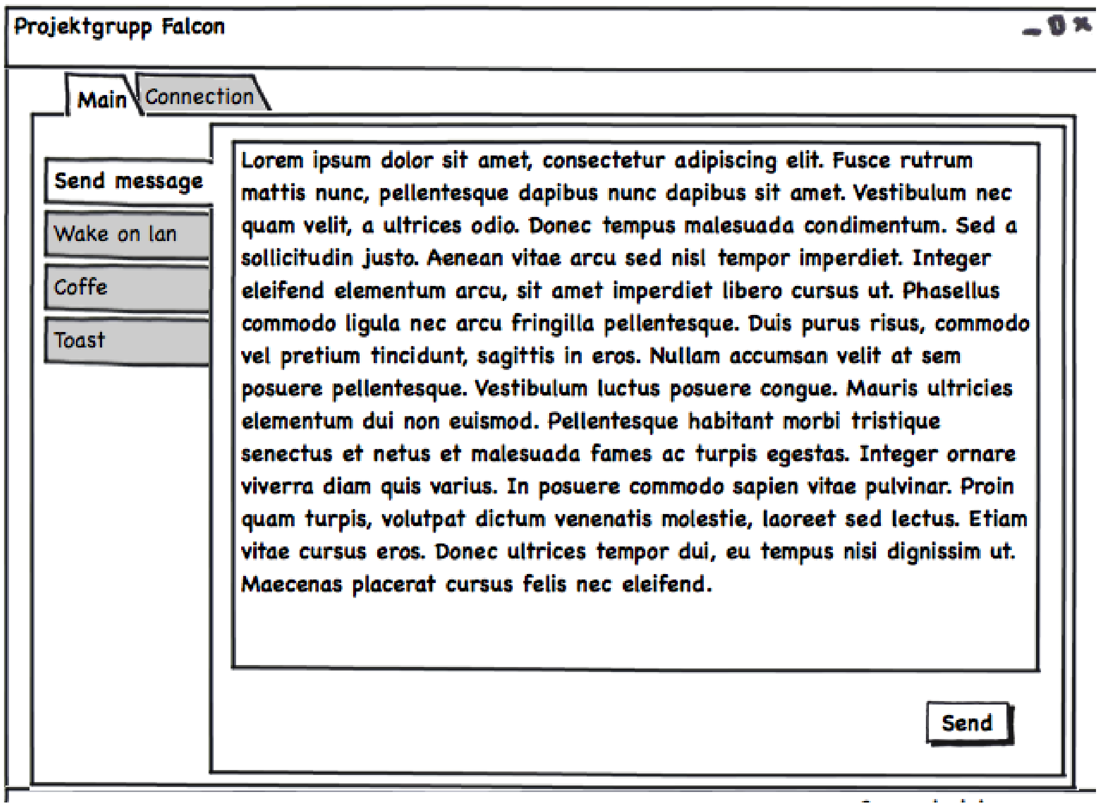
Exceptional flow - nothing happens

(1-5 same as normal flow)

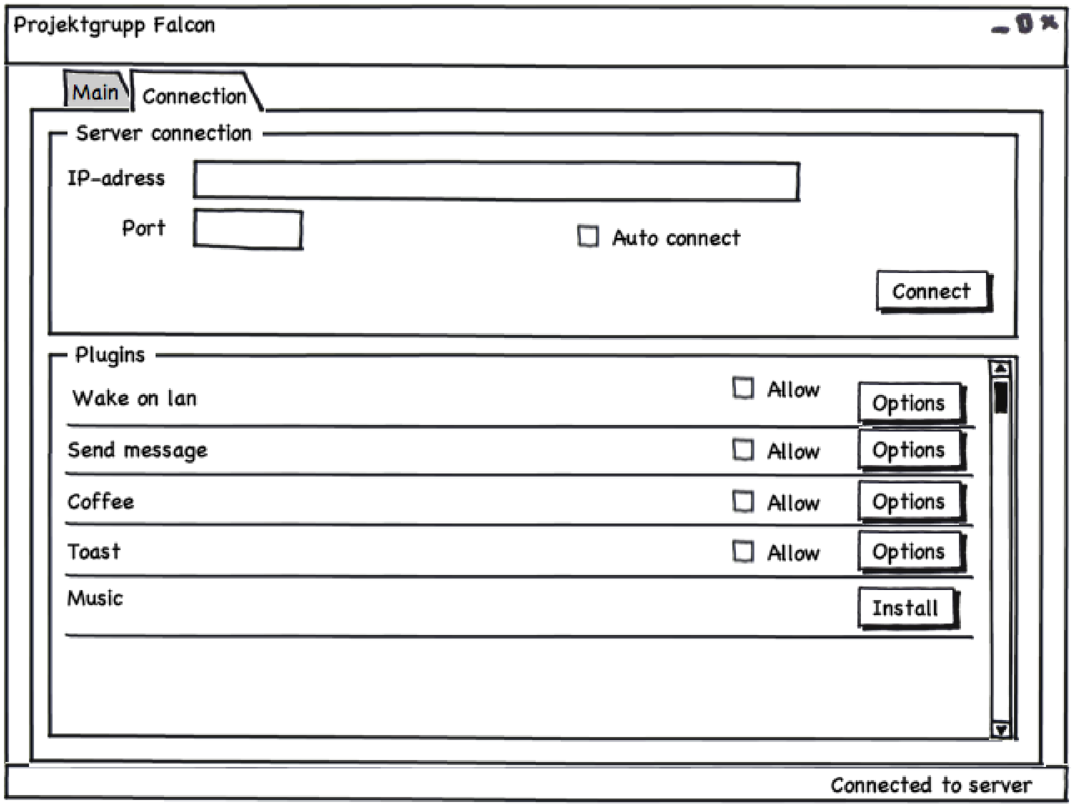
6. Fails to send the plugincall somewhere along the way. (client)
7. Nothing happens (server)

3.2 GUI

Client application mock-ups
(NOTE: this is how the vision was, not how the GUI turned out)



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce rutrum mattis nunc, pellentesque dapibus nunc dapibus sit amet. Vestibulum nec quam velit, a ultrices odio. Donec tempus malesuada condimentum. Sed a sollicitudin justo. Aenean vitae arcu sed nisl tempor imperdiet. Integer eleifend elementum arcu, sit amet imperdiet libero cursus ut. Phasellus commodo ligula nec arcu fringilla pellentesque. Duis purus risus, commodo vel pretium tincidunt, sagittis in eros. Nullam accumsan velit at sem posuere pellentesque. Vestibulum luctus posuere congue. Mauris ultricies elementum dui non euismod. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Integer ornare viverra diam quis varius. In posuere commodo sapien vitae pulvinar. Proin quam turpis, volutpat dictum venenatis molestie, laoreet sed lectus. Etiam vitae cursus eros. Donec ultrices tempor dui, eu tempus nisi dignissim ut. Maecenas placerat cursus felis nec eleifend.



Connected to server