Interactive multi-user program, I.M.U.P.

Operating systems and multicore programming (1DT089)

Project proposal for group 4: Niklas Andersen(791027-0334), Henrik Bihlar(920407-2913), Linda Eriksson(900803-1941), Viktor Källberg(910510-2553), Tobias Källman Andersson(920917-1413), Elias Lundeqvist(910702-2775)

Version 1, 2013-04-22

Table of contents

1. Introduction

The purpose of the project is to develop a network platform for simple multiplayer games. The user will enter the platform where all games and users online are visible. From there on the user can invite any player(s) from the online-list to play any game.
This seems a good way to learn how to combine a graphical user interface (GUI), different languages and interactivity, all in one project. Since this is the first time for most of us to get acquainted to a graphical interface, we are really looking forward to develop this platform.
During the brainstorming session our main goal was to create an interactive multi-user program which could be used by multiple machines, and the actual programs connected to the platform was of less importance.
We also had plans to make an application for smartphones, but that task might be too time consuming for our project, and therefore we chose not to do it.
We think that the main challenge of this project will be to master all the new techniques, and combining them to make a synchronized, interactive, and concurrent program.

The games created will be chosen in the aspect of concurrency, and every user will be a process, able to communicate with other users via message passing. The idea is to use a collector function that synchronizes all users and games.

2. System architecture

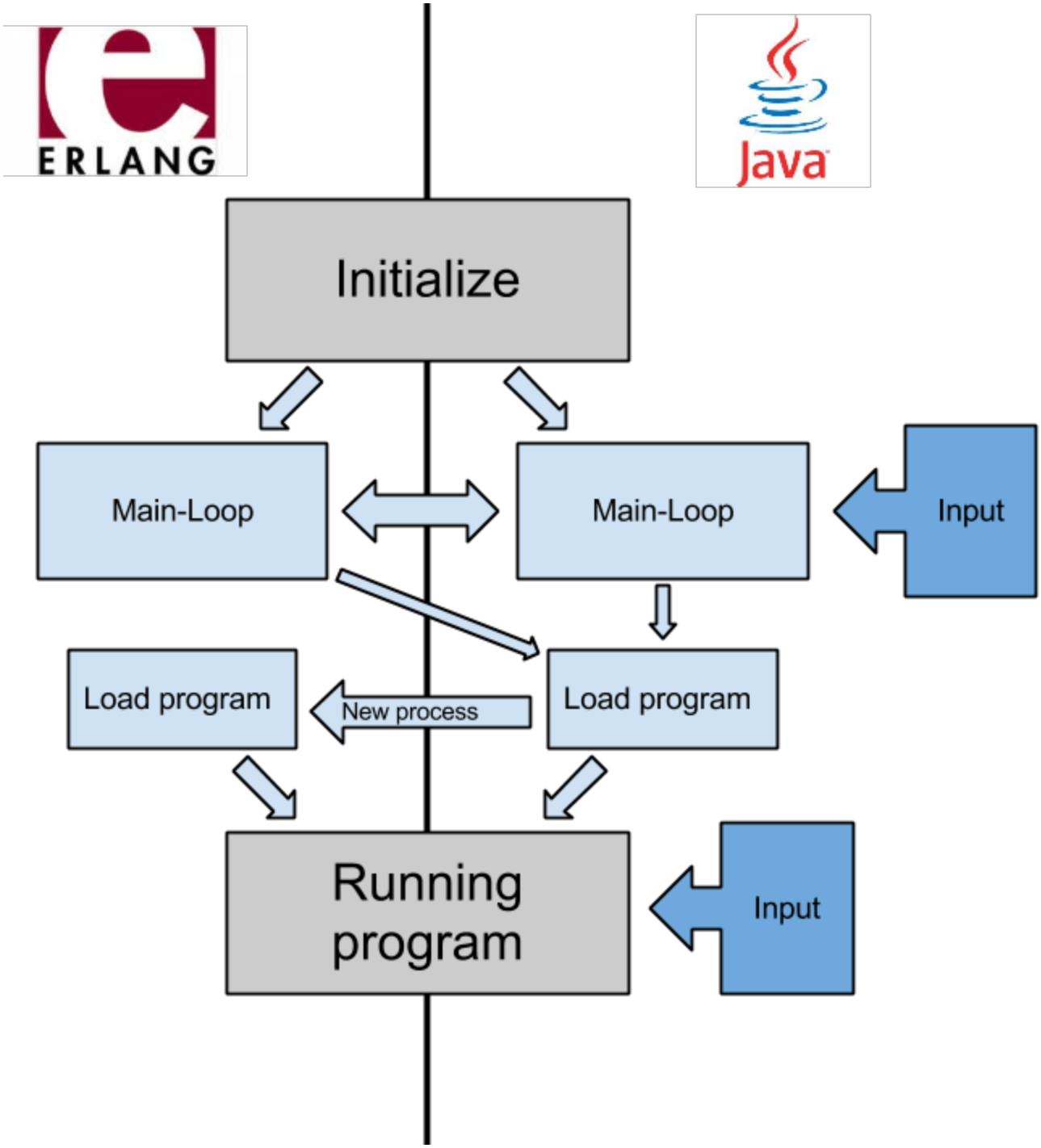

Image 1, System Design: *The system is divided into an Erlang part and a Java part*

- ***Initialize:*** *In this phase the settings for the session are managed. This includes network configuration and various cosmetic settings.*
- ***Main-Loop:*** *In this phase the program to be loaded is chosen, and its various settings are*

managed. Erlang and Java will need to exchange information here. This part will also keep track of every program that is running.

- ***Input:*** *Java takes care of input. This can be keyboard presses, mouse clicks or network input.*
- ***Load Program:*** *When everything has been set, it's time to load the specific settings for the program chosen. Java tells Erlang to create a new process for the program and to prepare for running it.*
- ***Running program:*** *This phase is a running program, in theory it can be anything.*

The system will use the two languages Erlang and Java. Erlang is going to handle the back-end communication, i.e. the networking and the various users. Java will handle the GUI, certain calculations and input, and the games.

Our system is divided into five big components; these are named; Initialization, Main-loop, Input, Load Program and Running program. Erlang and Java will communicate between each other in these components, e.g. there will be a main-loop in Erlang and a main-loop in Java. We have chosen this particular structure because it is our belief that message passing and the building of programs will be straightforward.

The following is a short description of how the system should work:

- Connect to server or host a server, with some kind of user id
- GUI with programs and people connected
- When a program has been chosen it is loaded with its connected users, messages and settings. If someone decides to load a program, the other users should be given a prompt to join or decline the game.

The focus lies on getting the framework to work successfully, the games are a side effect. Hence, we will not put a large effort on the games. Example games of what we're going to implement are tic-tac-toe and bomberman.

3. Concurrency models

The plan is to use Erlang and the actor model for concurrency and Java for the graphical interface. One of Erlangs many advantages is that concurrent programming is relatively easy, therefore Erlang will be one of the languages during this project. Another of Erlang's advantages is the ease of network communication through it's message handling. One of the big disadvantages of Erlang is the graphical part, and therefore Java will be used for this particular part of the project. We thought that it could be worthwhile to combine two different languages. One other option is to only use Scala because of the simplicity and the affinity to Java, but decided not to use it because of the loss of the opportunity to combine

two different languages.

4. Development tools

As we have some experience of Java and Erlang, we will continue to use the tools that we are used to.

- **Source code editor:** Netbeans will be used for Java since it has great support for testing, documentation and debugging. Emacs will be used for Erlang due to its auto indenting feature.
- **Revision control and source code management:** We will use Assmebla as our git-repository since we used it in our mini project and also because it is a private repository free to use. Git has a great number of users which implies that it is good.
- **Build tool:** The Make tool will be used since that is what we are used to and we didn't consider any other alternatives since Make works as intended.
- **Unit testing:** JUnit will be used in Java, and EUnit will be used for Erlang.
- **Documentation generation:** Javadoc will be used for the Java code and EDoc for the Erlang code, as they both are associated with their respective languages. This will generate more time for us to concentrate on completing the project.

5. Process evaluation

There were some complications in arranging the brainstorming session and thus we had to do parts of the work at home and the group was limited for the discussion. We agreed on a mutual proposal with relative ease even though parts of the group were absent.
Even though we had our complications everything were straightforward - we came up with some ideas and picked the idea that seemed most suitable for this project.
We as a group have to work on accessibility and synchronization in terms of meetings and communication, the solution to this is by booking group rooms with a set time every day when we have nothing else scheduled.