Programming Life - Test and implementation plan

Group 5/E:
Felix Akkermans
Niels Doekemeijer
Thomas van Helden
Albert ten Napel
Jan Pieter Waagmeester

March 18, 2012

Contents

1 Introduction

In this report the different testing techniques we will use for this project will be explained. Because our solution has a clear division between server and client and because these will be developed in different programming environments, we will also need different testing strategies for the client and server. In chapter 2 a prioritization of the requirements can be found using the MoSCoW system. Chapter 3 will explain how we will test the server and client, and what strategies we will use. Lastly chapter 4 will cover the risk analysis, describing the risks for the successful implementation of the system.

2 MoSCoW prioritization

In this chapter we will specify our priorities of requirements using the MoSCoW model. This model divides requirements on how viable it is to implement certain features: Must-Haves are features that the application cannot do without. These are all necessary for the program to function properly. Should-Haves group the features that are high-priority, but are not critical for the system. Could-Haves are features that would be nice to be have, should the time allow it and Wont-Haves are features that will not be implemented (in this version of the program).

2.1 Must Haves

Available gates The application must be able to present a list of available gates to the user. These gates can be used to model the circuit.

Design circuit The user must be able to design a circuit by specifying gates (using a drag-and-drop) and the relations between these gates.

Available proteins The application must be able to present the user with an overview of available proteins to assign to signals.

Protein specification The user must be able to specify which protein is used for a certain signal.

Export circuit The application must to able to save a circuit.

Import circuit The application must be able to load an exported circuit.

Input values specification The user must be able to specify the input values used for the simulation of the circuit.

Circuit simulation The application must be able to simulate a valid circuit and present the output values to the user.

2.2 Should Haves

Re-use BioBricks The application can import pre-defined circuits as extra gates. This is not a necessity, but would be a great addition to the program (and will ease building circuits). Among others, protein specification, importing and exporting will be more difficult to implement.

Circuit validation The user must to be able validate his circuit in the application. This is not a must as simulation will fail if the circuit is invalid, but modeling will be faster if the user can easily get feedback.

2.3 Could Haves

Determine proteins by specifying circuit, input and output values It is possible to let an algorithm choose the best proteins for the signals in a circuit, given user specified input and output values. This feature should be a nice extra and will be implemented if time allows it.

Local back-up If, for whatever reason, a crash occurs (the connection drops, the server stops functioning, etc.), it would be nice to provide the user with a backup of his/her work. This feature has not much to do with the main goal of this application (creating and simulating a circuit), so that is why it is a could-have feature.

Multi-client The application must be able to handle multiple clients concurrently. This is not a point of attention, as modeling can easily be done one circuit at a time. Another issue is that implementing and properly testing this feature desires significant attention, that is why we will do it if we have enough time.

2.4 Wont Haves

Determine cirtcuit and proteins by specifying input and output values It is possible to let an algorithm design a circuit based on merely given input and output signals. We deem designing such an algorithm takes up a lot of time and is impossible to do properly in our limited timespan.

Biological plausibility It is near impossible to create a program in which a user can model a biological circuit that will work in the real world as there are just too much (unpredictable) factors to take into account. With our limited knowledge of the subject, we will not try to pursue a biological plausible implementation.

3 Implementation and tests

3.1 Order of implementation of features

3.1.1 Iterations

3.1.2 Milestones

3.2 Server Test plan

3.2.1 Unit testing

3.2.2 Integration testing

3.2.3 Acceptance testing

3.3 Client Test plan

3.3.1 Unit testing

3.3.2 Integration testing

3.3.3 Acceptance testing

3.4 Testing Client-server integration

Integration testing is done when the individual software modules are combined and need to be tested as a whole. One approach is to use unit tests which test the whole system. The server can be run on the same machine as the client, in this way the unit tests can test these two components.

4 Risk analysis

(what are the risks for the successful implementation of the system?)