Project Fingerpaint

SRD-1.1

Software Requirements Document

Authors:
Tessa Belder (0739377)
Lasse Blaauwbroek (0749928)
Thom Castermans (0739808)
Roel van Happen (0751614)
Benjamin van der Hoeven (0758975)
Femke Jansen (0741948)
Hugo Snel (0657700)

Junior Management:
Simon Burg
Areti Paziourou
Luc de Smet

Senior Management:
Mark van den Brand, MF 7.096
Lou Somers, MF 7.145

Technical Advisor:
Ion Barosan, MF 7.082

Customer:
Patrick Anderson, GEM-Z 4.137

Eindhoven - June 23, 2013

Abstract

This document describes the Software Requirements of FINGERPAINT. This project is part of the Software Engineering Project (2IP35) and is one of the assignments at Eindhoven University of Technology. The Software Requirements Document (SRD) is based on the ESA standard for software development, as set by the European Space Agency (ESA) [1]. This document presents a specification of solutions for the requirements described in the URD [2].

Contents

Document Status Sheet

Document Status Overview

General

Document title:	Software Requirements Document
Identification:	SRD-1.1
Author:	Tessa Belder, Thom Castermans, Benjamin van der Hoeven, Roel van Happen, Femke Jansen, Lasse Blaauwbroek
Document status:	Approved by the customer

Document History

Version	*Date*	*Author*	*Reason of change*
0.0	21-May-2013	Tessa Belder, Benjamin van der Hoeven, Roel van Happen, Femke Jansen, Thom Castermans	Initial version.
0.1	23-May-2013	Roel van Happen,Thom Castermans	Revision after feedback from the technical advisor.
1.0	3-June-2013	-	Approved by the customer.

Document Change Records Since Previous Issue

General

Date:	7-June-2013
Document title:	Software Requirements Document
Identification:	SRD-1.1

Changes

Page	*Paragraph*	*Reason to change*
10	2.7	Updated text in this section w.r.t. updated figure 2.1.

Chapter 1

Introduction

This chapter lists general information about this document.

1.1 Purpose

This document provides a translation of all the user requirements listed in section 3 of the URD [2]. Although the URD describes the wishes of the client, the goal of the SRD is to represent the developers' view of what the FINGERPAINT application must be able to do. Note that the software requirements listed in this document are implementation-independent: that is, the requirements describe *what* FINGERPAINT must do, but not *how* the requirements will be realized. The requirements are modelled in a logical model, which provides a simplified view of the content and behaviour of the application.

1.2 Scope

FINGERPAINT is an application designed and developed by Group Fingerpaint for prof. dr. ir. P.D. Anderson. The application provides a cross-platform tool to visualise fluid mixing. Users can define the initial concentration distribution, as well as manipulate the mixing protocol. The resulting fluid distribution can be stored by the user on their device, for later reference.

1.3 List of definitions and abbreviations

1.3.1 Definitions

Client Prof.dr.ir. P.D. Anderson.

Firefox A web browser developed by Mozilla.

Google Chrome A web browser developed by Google.

Internet Explorer A web browser developed by Microsoft.

iOS A mobile operating system developed by Apple.

iOS Safari A web browser developed by Apple designed for devices running iOS.

iPhone A line of smartphones developed by Apple.

iPad A line of tablet computers developed by Apple.

Opera A web browser developed by Opera Software.

Safari A web browser developed by Apple.

System administrator A person who is employed to maintain and operate a computer system and/or network. After the SEP project has been completed, this person will be responsible for maintaining the FINGERPAINT application.

1.3.2 Abbreviations

2IP35	The Software Engineering Project
ADD	Architectural Design Document
CM	Configuration Manager
GUI	Graphical User Interface
SEP	Software Engineering Project
SR	Software Requirements
SRD	Software Requirements Document
TU/e	Eindhoven University of Technology
URD	User Requirements Document

1.4 List of references

[1] ESA, *ESA Software Engineering Standards.* ESA, March 1995.

[2] Group Fingerpaint, "User requirements document," *SEP*, 2013.

[3] prof.dr.ir. P.D. Anderson, "prof.dr.ir. P.D. Anderson's homepage." http://www.mate.tue.nl/mate/showemp.php/19. [Online; accessed 1-May-2013].

[4] Group Fingerpaint, "Architectural design document," *SEP*, 2013.

[5] COLEY consulting, "Moscow prioritisation." http://www.coleyconsulting.co.uk/moscow.htm. [Online; accessed 24-April-2013].

1.5 Overview

The remainder of this document describe the software requirements in more detail. Chapter 2 gives a general description of:

- relation to current projects (2.1);
- relation to predecessor and successor projects (2.2);
- function and purpose (2.3);
- environment (2.4);

- relation to other systems (2.5);
- general constraints (2.6);
- model description (2.7).

Chapter 3 gives a detailed description of the functional requirements of the system in 3.1 and a list of non-functional requirements is given in 3.2. The requirements traceability matrix is described in chapter 4.

Chapter 2

General Description

In this chapter we discuss the relation of this project to the "outside world": if there are any related projects running currently and if there were related projects in the past. Then, the purpose of the FINGERPAINT application and the environment in which it operates are discussed. After that, its relation to other systems is covered. Finally, some general constraints are described and a description of the logical model is given.

2.1 Relation to current projects

No other current projects are related to FINGERPAINT.

2.2 Relation to predecessor and successor projects

FINGERPAINT has multiple predecessor projects. These projects resulted in multiple Matlab[1] tools that are available on the client's web page [3]. FINGERPAINT will combine some of the functionality of these tools into a mobile web application. FINGERPAINT will be developed in such a way that the client can easily extend the application with new mixers. When the development of FINGERPAINT is completed, FINGERPAINT is no longer responsible for the application after the final deliverable produced in the SEP project. This means that the client may change, add or remove the application's functionality.

2.3 Function and purpose

FINGERPAINT is an application that serves as an educational tool for anyone who wants to gain a deeper understanding of the process of mixing in general, and in particular for students at the TU/e. By interacting with the application, users can quickly and easily find out what the effects of a certain mixer and mixing protocol are on an initial distribution. The user can thus obtain a better understanding about the way this mixer functions. FINGERPAINT may also be used as a quick and convenient way to observe whether a mixing protocol renders good or bad mixing results.

[1]`http://www.mathworks.nl/products/matlab/`

2.4 Environment

Fingerpaint is a web application that is developed primarily for use on mobile devices. This means the application will mostly be accessed through web browsers on smartphones and tablets. It is expected that the application will mostly be used on iPhones and iPads. Therefore, Fingerpaint must support iOS Safari version 6.0 and above. Furthermore, Fingerpaint should support Firefox version 20 and above, and Google Chrome version 26 and above. Lastly, if time permits, Fingerpaint could also support Internet Explorer version 10 and above, Opera version 12.1 and above, and Safari version 6.0 and above.

To support the significant share of smartphones and tablets that run on Android, Fingerpaint should run on devices running on Android version 4.0 and higher. Lastly, if time permits, Fingerpaint could also run on devices running on Windows 8.

The hardware used by the users must be able to run at least one of the supported operating systems and browsers. Also, the application obviously works better on screens that have a diagonal of at least about 4 inches and a resolution of at least about 540x960 pixels - the larger the screen, the easier it is to draw on it (up to a certain maximum, about the size of a desktop monitor with a diagonal of 20 inches). This is because the user draws with their finger and since fingers have a certain size, the screen should be large enough to comfortably draw and see what has been drawn at the same time.

The application must support the following screen resolutions:

- "Phone portrait": 540x960 pixels;
- "Phone landscape": 960x540 pixels;
- "Tablet portrait": 800x1280 pixels;
- "Tablet landscape": 1280x800 pixels;
- "Desktop": 1600x900 pixels.

2.5 Relation to other systems

The Fingerpaint application is an independent system. However, just like every other web application, it is dependent in some way on the browser. That is, the correct (according to the HTML standard[2]) rendering of the web page is done by the browser. Of course, the application will be tested in multiple browsers and built in such a way that it displays correctly in as many browsers as possible.

2.6 General constraints

The user interface should be suitable for mobile devices, so it will be easy to share the visualised results with other people, and to quickly try out new ideas for mixers wherever the user may be.

[2] http://www.whatwg.org/

We assume that the server can compute the displacement of fluids reasonably fast[3], so the mixing run can be executed quickly. When the new concentration distribution has been computed, this concentration distribution is sent back to the client device along with a metric to indicate the performance of the mixer. The results are then visualised on the client's device.

As we do not want to be locked to one specific type of device, we have chosen to design a cross-platform solution. While this means that desktop PCs should also be able to run the application, we do not actively support such devices. We will instead concentrate on mobile devices.

It should be possible to save mixing runs on the client device for later reference. For each saved run, we store the initial distribution, the mixer and protocol used, the resulting fluid distribution and the resulting performance metric.

2.7 Model description

On a very high level, the architecture of the Fingerpaint application can be divided into different tiers communicating through communication channels. A graphical representation of the relation between tiers and channels can be found in figure 2.1.

2.7.1 Client tier

Of the Client tier, there are an arbitrary number of instances. These instances are the physical machines of the users of the Fingerpaint application (phones, tablets, laptops, desktops).

Client Browser and Client Persistence

The Client Browser provides the user with a Graphical User Interface. All interactions of the user with the application are performed through this GUI. Each Client instance runs a Client Browser (one of the browsers as specified in section 2.4) and has a persistent storage facility (Client Persistence). The Client Browser can use this facility to store data that is specific to the user and does not need to be stored in a central location.

2.7.2 Application Server tier

The Application Server is a physical machine maintained by the system administrator of the application. It is used to distribute the application and provide services to the application.

HTTP Server

The application is distributed on demand to the Client instances using HTTP by the HTTP server. The HTTP server is a piece of software that responds to requests by serving a file from a static collection of content, which is used to serve the actual application.

[3]Refer to chapter 6 of the ADD [4] for exact requirements.

Application Service

Whenever centralised data or a simulation is required by the application running on a Client, this is handled by the Application Service in response to a request from the Client Browser. This service is responsible for communicating any simulations and their results and also for serving data from the persistent storage. Data from the persistent storage can be a list of mixers for example.

2.7.3 Application Persistence tier

The Application Service may use a global persistent storage facility in order to store data that needs to be available to all Clients. This storage facility is provided in the Application Persistence tier and is communicated to through the Application Persistence Communication channel. This tier may be on a different physical machine than the Application Server, but in practice, it will likely run on the same hardware.

2.7.4 Simulator Server

The simulations that need to be done for the application run on a dedicated machine although in practice, this will likely be the same physical machine as the Application Server.

Simulator Service and Fortran Module

Whenever a Client wishes to run a simulation, it interfaces with the Simulator Service indirectly through the Application Service Communication channel, the Application Service and the Simulator Service Communication channel. The Simulator Service uses an existing Fortran Module to calculate the result of a simulation.

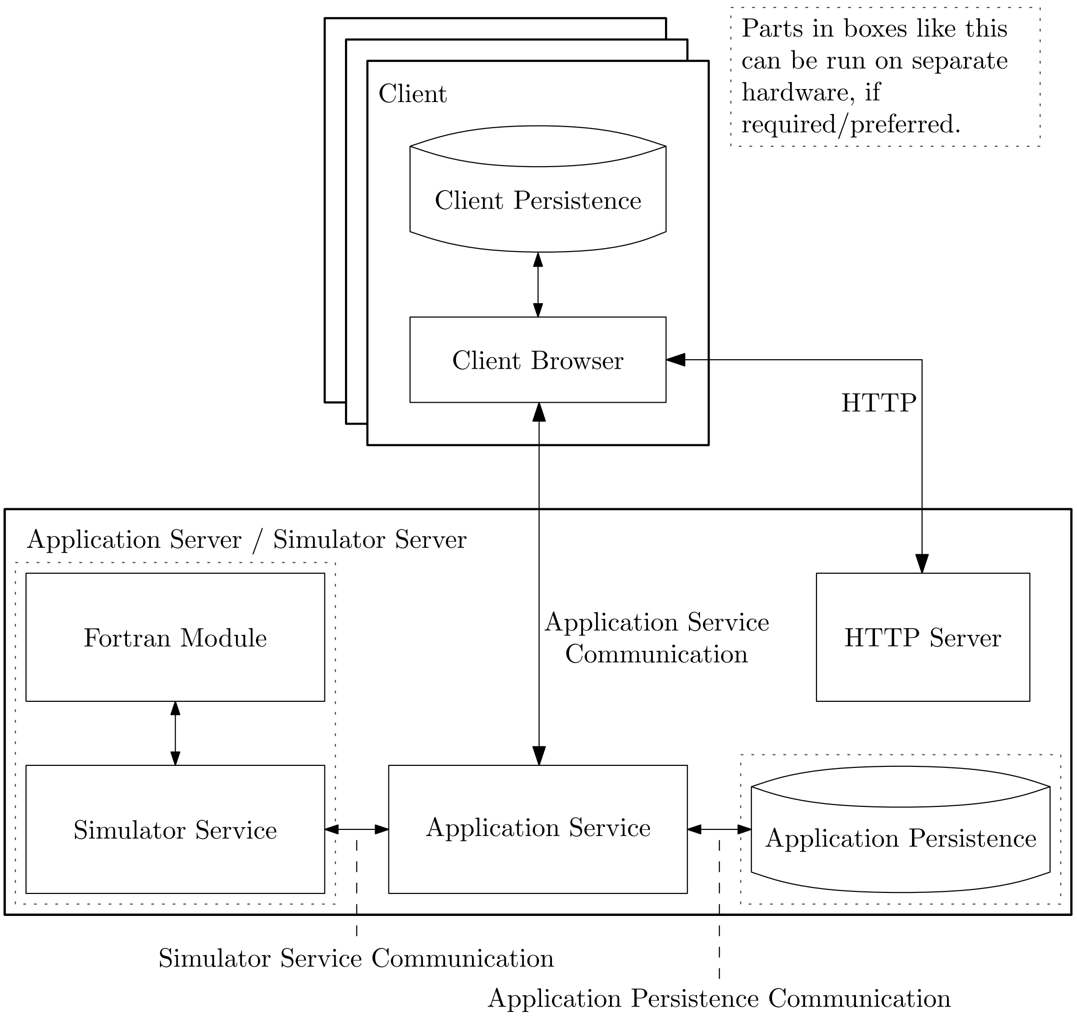
Parts in boxes like this can be run on separate hardware, if required/preferred.
Client
Client Persistence
Client Browser
HTTP
Application Server / Simulator Server
Fortran Module
Application Service Communication
HTTP Server
Simulator Service
Application Service
Application Persistence
Simulator Service Communication
Application Persistence Communication

Figure 2.1: The different tiers of the system

Chapter 3

Specific requirements

This chapter lists all specific software requirements of the application to be developed, both functional and non-functional requirements. The requirements are categorised according to the interface they belong to, as described in section 2.7.

Each requirement has a specific priority, based on the MoSCoW model [5]:

- *must have*; requirements with this priority are essential for the product, and must be implemented.
- *should have*; requirements with this priority are not essential for the product to work. However, they are nearly as important as the *must have*'s and are therefore expected to be implemented.
- *could have*; requirements with this priority are a nice addition to the product, and may be implemented, if time and budget allow this.
- *won't have*; requirements with this priority will not be implemented in this version of the product, but may be nice to implement in future versions.

Only those user requirements from the URD [2] with a priority higher than *won't have* will be translated to software requirements in this chapter.

In some of the requirements in this section, the term "fast enough" is mentioned: for a precise definition, refer to chapter 6 of the ADD [4].

3.1 Functional requirements

This section lists the functional requirements for the FINGERPAINT application.

3.1.1 HTTP Server

SRQ1 *must have*
The HTTP Server must be able to serve static files (files that are not dynamically generated) to the Client Browser.

3.1.2 Application Service

SRQ2 *must have*
The Application Service must be able to retrieve all the available mixers from the Application Persistence.

SRQ3 *must have*
The Application Service must be able to send all available mixers to the Client Browser.

SRQ4 *must have*
The Application Service must allow for adding new mixers to the Application Persistence.

SRQ5 *must have*
The Application Service must allow for removing existing mixers from the Application Persistence.

SRQ6 *must have*
The Application Service must be able to provide all available geometries to the Client Browser.

SRQ7 *must have*
The Application Service must be able to act as a relay between the Client Browser and the Simulation Service.

3.1.3 Application Persistence Communication

SRQ8 *must have*
The Application Persistence Communication must be able to access all the data from the Application Persistence.

SRQ9 *must have*
The Application Persistence Communication must be fast enough to serve the data from the Application Persistence to the Application Service.

3.1.4 Application Persistence

SRQ10 *must have*
The Application Persistence must deliver all saved mixers to the Application Service.

SRQ11 *must have*
The Application Persistence must be able to save new mixers provided by the Application Service.

SRQ12 *must have*
The System Administrator is able to add new mixer types for a geometry.

SRQ13 *must have*
The System Administrator is able to remove mixer types for a geometry.

3.1.5 Application Service Communication

SRQ14 *must have*
The Application Service Communication must be able to access all the functionality from the Application Service.

SRQ15 *must have*
The Application Service Communication must be fast enough to provide the functionality from the Application Service to the Client Browser.

3.1.6 Client Browser

Change the drawing tool

The user can change the drawing tool that is used to paint on the canvas. The current shape (circle and square) and size of the drawing tool is displayed on the user interface. The user interface contains options to change the current shape and size of the drawing tool.

SRQ16 *must have*
The user interface displays the shape of the tool that is currently selected.

SRQ17 *should have*
The user interface displays the size of the tool that is currently selected.

SRQ18 *must have*
A circle-shaped drawing tool can be selected on the graphical user interface.

SRQ19 *should have*
A square-shaped drawing tool can be selected on the graphical user interface.

SRQ20 *should have*
A graphical user interface is present to adjust the size of the current tool.

Defining an initial concentration distribution

The user can select a circle or square-shaped drawing tool to draw with. He can draw on the canvas by swiping with his finger or drawing with the mouse. In addition, the user can reset the drawn distribution to a completely white concentration distribution.

SRQ21 *must have*
A graphical interface is present to select the colour (black or white) to paint with for the initial concentration distribution.

SRQ22 *must have*
A graphical interface is present to define an initial concentration distribution with the selected color on the canvas. It must be possible to define this distribution by means of “painting” on the canvas.

SRQ23 *must have*
If the circle-shaped drawing tool in SRQ18 is selected, this tool will be used when the canvas is "painted".

SRQ24 *should have*
If the square-shaped drawing tool in SRQ19 is selected, this tool will be used when the canvas is "painted".

SRQ25 *must have*
A graphical user interface is present to reset the current initial concentration distribution to a completely white state.

Select a geometry, mixer and initial concentration distribution

The user can select a rectangular mixer geometry. Then the user can select a mixer that can operate on the selected geometry and a initial concentration distribution. There are three types of initial concentration distributions that the user can choose from. The first type is a `blank` concentration distribution, which simply means that the user will be presented with a clean canvas when this option is selected. The second type is a `load` option, so the user can load previously saved concentration distributions that are stored on their device. The third type of concentration distribution is a `predefined` distribution, meaning that the user can choose from a few predefined distributions that are already present in the application. The selected concentration distribution is then loaded onto the canvas. This translates to the following software requirements:

SRQ26 *must have*
The application contains a graphical interface to select an initial mixer, geometry and options for the initial concentration distribution.

SRQ27 *must have*
A `rectangle` geometry can be selected in the user interface.

SRQ28 *must have*
If a geometry is selected, a user interface is presented to select an appropriate mixer.

SRQ29 *should have*
A `square` geometry can be selected in the user interface.

SRQ30 *could have*
A `circle` geometry can be selected in the user interface.

SRQ31 *could have*
A `journal bearing` geometry can be selected in the user interface.

SRQ32 *must have*
After a geometry and appropriate mixer is slected in the user interface, a blank canvas can be selected and loaded.

SRQ33 *should have*
After a geometry and appropriate mixer is slected in the user interface, a list of all previously saved concentration distribution can be retrieved in the user interface.

SRQ34 *should have*
After the list of concentration distribution from SRQ33 is presented, a previously saved concentration distribution can be selected and loaded onto the canvas.

SRQ35 *could have*
After a geometry and appropriate mixer is slected in the user interface, a list of all predefined concentration distribution can be retrieved in the user interface.

SRQ36 *could have*
After the list of concentration distribution from SRQ35 is presented, a predefined concentration distribution can be selected and loaded onto the canvas.

Define mixing protocol for specific geometries

A mixing protocol consists of movements of the geometry (if applicable), and how long these movements are executed. Different movement types are possible for each geometry. A step of a mixing protocol has a certain duration (the step size, D). This duration D can be any multiple of 0.25. If a value is not a multiple of 0.25, it will be rounded to the nearest valid value. For example, 4.2 is rounded up to 4.25, while 4.1 is rounded down to 4. Each movement is performed for D time units, which can be set using SRQ37. Depending on the geometry, a step has additional parameters signifying which part moves and how:

For the *rectangular* or *square* geometries, a wall movement is defined by either T or B, denoting a movement of the top or bottom wall, respectively. These movements can be combined with a L or R denoting whether the wall is moving left or right. This means there are four movement modifiers: TL, TR, BL and BR.

The *circular* geometry only supports one (unnamed) modifier.

For the *journal bearing* geometry, movements are defined by rotating the outer or inner circle. The outer circle is denoted with an O and the inner circle is denoted with an I. These modifiers can be combined with modifiers denoting the direction of the rotation, R and L. This results in four possible modifiers: OL, OR, IL and IR.

The user interface will contain options to input modifiers where appropriate and to indicate the step-size associated with those modifiers.

A graphical interface is present to define how many times the current protocol should be run when the simulation has started.

SRQ37 *must have*
A graphical user interface is present to associate a valid step-size (as defined above) to a step.

SRQ38 *must have*
A graphical user interface is present to indicate how many times the currently defined protocol should be executed when the simulation starts.

SRQ39 *must have*
A graphical user interface is present to associate a valid step-modifier (as defined above) to a step.

SRQ40 *must have*
A graphical user interface is present to add and remove steps to the protocol.

SRQ41 *must have*
If a geometry, a mixer, an initial distribution and a protocol are present, a graphical user interface must be available to execute the actual mixing simulation.

SRQ42 *must have*
A grahpical user interface is present to reset the currently defined protocol, thereby removing all currently defined steps.

Execute mixing runs

The user can choose to execute a single step of a mixing protocol, by defining a single step. This is done by choosing one wall movement and its duration. This invokes SRQ41, SRQ81, SRQ83, SRQ82 and SRQ44.

SRQ43 *must have*
After selecting a geometry, a mixer, an initial distribution and a single step, the user can execute a mixing run with this single step.

Visualising the results

The concentration distribution resulting from a mixing simulation is visualised in the user interface, using the canvas that was originally used to draw the initial concentration distribution. The performance results are visualised in a graph.

SRQ44 *must have*
Concentration distributions can be drawn on the client's canvas.

SRQ45 *must have*
Performance results can be plotted in a graph.

3.1.7 ClientPersistence

SRQ46 *should have*
The client persistence is capable of storing concentration distributions, mixing protocols and associated mixers and geometries. Results of a mixing run can also be stored.

Managing mixing protocols

To manage mixing protocols, the user can save and load the mixing protocols they have created. It is also possible to load a predefined mixing protocol.

SRQ47 *should have*
A graphical user interface is present to save the concentration distribution currently drawn on the canvas to the ClientPersistence.

SRQ48 *should have*
A graphical user interface is present to save the currently defined mixing protocol to the ClientPersistence.

SRQ49 *should have*
A graphical user interface is present to view all saved mixing protocols and remove a mixing protocol.

SRQ50 *should have*
A graphical user interface is present to view all saved mixing protocols appropriate to the currenctly selected geometry. A mixing protocol can be selected from this list and loaded into the current mixing protocol.

SRQ51 *could have*
After SRQ27, SRQ29, SRQ30 or SRQ31, the user can load a predefined mixing protocol. This opens a menu containing all suitable predefined mixing protocols for the selected geometry. Pressing the name of one such protocol selects it, which means it will be loaded into the application. After this it can be changed or used for simulation immediately.

Save and remove mixing runs

SRQ52 *must have*
After the mixing has been executed, a graphical user interface is present to save the mixing run. That is, the initial concentration distribution of the run is saved (if not already saved), the mixing protocol is saved (if not already saved), the performance result of the mixing simulation is saved and the used geometry and mixer are saved.

SRQ53 *must have*
A list with previously saved mixing runs can be viewed on the user interface, and a saved mixing run can be selected and removed.

Save an initial concentration distribution

SRQ54 *should have*
A graphical user interface is present to save the concentration distribution currently displayed on the canvas.

SRQ55 *should have*
During the SRQ54 action a name for the concentration distribution can be inputted.

SRQ56 *should have*
After specifying a name in SRQ55, the concentration distribution can be saved to the ClientPersistence under the inputted name.

SRQ57 *should have*
If the specified name from SRQ55 is already taken, a graphical user interface is present to overwrite the previously saved concentration distribution.

SRQ58 *should have*
The saving process can be cancelled at any time.

Load previously saved concentration distribution

SRQ59 *should have*
A graphical user interface is present to initiate the loading of a previously saved concentration distribution.

SRQ60 *should have*
After the loading process from SRQ59 has been initiated, a list of previously saved concentration distributions is presented.

SRQ61 *should have*
In the list from SRQ60, a previously saved distribution can be selected.

SRQ62 *should have*
A concentration distribution (for example the one selected in SRQ61) can be loaded into the canvas.

Remove previously saved concentration distributions

SRQ63 *should have*
A user interface is present to initiate the removal of a previously saved concentration distribution.

SRQ64 *should have*
After the removal action from SRQ63 has begun, the user interface will present a list of previously saved concentration distributions.

SRQ65 *should have*
After the list from SRQ64 is shown on the user interface, one or more previously saved concentration distributions can be selected for removal.

SRQ66 *should have*
If at least one item from SRQ65 is selected, a graphical user interface is presented to remove those items from storage. A confirmation will be required before the removal commences.

SRQ67 *should have*
If no distributions have been saved yet, and SRQ63 has been initiated, a message will be shown to inform the user that no items can currently be removed.

SRQ68 *should have*
If no access rights have been provided to the application, a *Insufficient access rights* error message is shown when SRQ63 is initiated.

Exporting results

It is possible to save an image of the results of a performed mixing run to the disk of the Client. This can be a image of the resulting concentration distribution, an animation of several intermediate concentration distributions (as indicated by the step-repeat from SRQ38) or a graph displaying the performance of a mixing run (a performance point is generated for each time the protocol was repeated using SRQ38).

SRQ69 *should have*
A graphical user interface is present to export an image of the current performance graph.

SRQ70 *could have*
A graphical user interface is present to export an animation or still frame of the current mixing protocol.

SRQ71 *should have*
The generated image or animation from SRQ69 and SRQ70 is presented to the ClientBrowser and the ClientBrowser is expected to handle further steps in order to save the image or animation to the disk.

Loading results

Previously saved results can be loaded into the application.

SRQ72 *should have*
A graphical user interface should be present to initiate the loading of previously saved results. The interface shows a list of these saved results.

SRQ73 *should have*
After the loading action from SRQ72 has been initiated, a saved result can be selected to load.

SRQ74 *should have*
When SRQ73 is executed, the concentration distribution resulting from the saved mixing simulation is loaded and the performance graph is displayed.

SRQ75 *should have*
A graphical user interface should be present to load multiple results to create a performance graph where multiple results are overlayed. The result of this graph can be saved to the disk.

Language selection

It is possible to change the language of the application. The standard language is English from SRQ93, and it should be relatively easy to add other languages.

SRQ76 *could have*
A graphical user interface is present to select one of the available languages for the application (these languages are SRQ93 and SRQ94).

Requirements regarding the presentation of results

The user is able to get the distribution that resulted from executing his mixing run. In addition, the mixing performance is given after each mixing step. He is also able to save and export these results. To this end a results window is available. In the results window, the final distribution is shown. There is also a `show statistics` button. When pressed, the performance of the mixing run is shown in a graph. In this new menu the `show statistics` button changes to `show result`. When this button is pressed, the image switches back to the default results window. The results window is reached by pressing the `start mixing` button in the mixing interface. The mixing interface is the window where the user can define his mixing protocol. In the mixing interface there is a `animate mixing` checkbox. If this checkbox was checked when the `start mixing` button is pressed, an animation of the mixing procedure is shown in the results window.

SRQ77 *must have*
In the results window the result of the user's mixing protocol is shown. This interface can be reached by pressing the `start mixing` button.

SRQ78 *must have*
The user can use the `show performance` button to view the performance graph of the mixing run.

SRQ79 *must have*
The user can use the `show result` button to switch back to the standard results window and view the result distribution of the mixing run.

SRQ80 *could have*
An animation of the mixing run is shown in the results window if the `animate mixing` checkbox was checked.

3.1.8 Simulator Service Communication

SRQ81 *must have*
The parameters (initial concentration distribution, geometry, mixing protocol, mixer and number of protocol applications) can be sent from the Application Service to the Simulation Service through the Simulator Service Communication channel.

SRQ82 *must have*
The results of a mixing simulation can be sent from the Simulator Service to the Application Service through the Simulator Service Communication.

3.1.9 Simulator Service

SRQ83 *must have*
The server executes the mixing run using the parameters received from the Simulator Service Communication. These parameters are passed as input to the Fortran implementation.

SRQ84 *must have*
The server can return the results from a mixing simulation it received from the fortran module to the Application Service through the Simulator Service Communication.

Information exchange

The Simulator Service must be able to send and receive matrix files from the Client Browser and to the Fortran Module. These matrices specify the characteristics of the mixers.

SRQ85 *must have*
The Simulator Service must be able to receive protocol information from the Client Browser through the Application Service.

SRQ86 *must have*
The Simulator Service must be able to pass protocol information to the Fortran Module.

3.1.10 Fortran Module

SRQ87 *must have*
The Fortran Module must accept input (matrix name, protocol information) from the Simulator Service.

SRQ88 *must have*
The Fortran Module must provide output (vectors) to the Simulator Service.

SRQ89 *must have*
The Fortran Module must have knowledge of all the geometries and mixers used in the application. This knowledge is not automatically transferred from the application to the Fortran module.

3.2 Non-functional requirements

3.2.1 Performance

SRQ90 *must have*
Waiting time between submitting input and receiving output is around 5 seconds on average.

SRQ91 *should have*
Waiting time between submitting input and receiving output is around 3 seconds on average.

SRQ92 *could have*
Waiting time between submitting input and receiving output is around 1 second on average.

3.2.2 Interface

SRQ93 *must have*
An English interface is available.

SRQ94 *should have*
A Dutch interface is available.

3.2.3 Portability

SRQ95 *must have*
The application runs on iOS Safari version 6.0 and higher.

SRQ96 *should have*
The application runs on Firefox version 20 and higher.

SRQ97 *should have*
The application runs on Google Chrome version 26 and higher.

SRQ98 *could have*
The application runs on Internet Explorer version 10 and higher.

SRQ99 *could have*
The application runs on Safari version 6.0 and higher.

SRQ100 *must have*
The application runs on devices running on iOS version 6 and higher.

SRQ101 *should have*
The application runs on devices running on Android version 4.0 and higher.

SRQ102 *could have*
The application runs on devices running on Windows 8.

Chapter 4

Requirements traceability matrix

4.1 URD to SRD

The following matrix lists which SRQs implement the CPRs from the URD.

CPR	SRD	CPR	SRD	CPR	SRD
1	26, 27	2	28	3	29
4	30	5	31	6	21, 22, 32
7	16, 18, 23	8	25	9	16, 19, 24
10	17, 20	11	46, 47, 54, 55, 56, 57, 58	12	63, 64, 65, 66, 67, 68
13	33, 34, 59, 60, 61, 62, 67	14	35, 36	15	*won't have*
16	*won't have*	17	37, 39, 40	18	37, 39, 40, 43
19	42	20	37	21	38
22	48	23	49	24	50
25	37, 39, 40	26	37, 39, 40	27	37, 39, 40
28	51	29	41, 44, 81, 82, 83, 84, 85, 86, 87, 88, 89	30	52
31	53	32	62, 77	33	71
34	78, 79	35	69, 71	36	75
37	71	38	80	39	71
40	93	41	94		

The following matrix lists which SRQs implement the CNRs from the URD.

CNR	SRD	CNR	SRD	CNR	SRD
1	95	2	96	3	97
4	98	5	99	6	*won't have*
7	100	8	101	9	102
10	90	11	91	12	92
13	4				

4.2 SRD to URD

The following matrix lists all CPRs from the URD implemented by each SRQ.

SRD	CPR	SRD	CPR	SRD	CPR
1	-	2	-	3	-
4	-	5	-	6	-
7	-	8	-	9	-
10	-	11	-	12	-
13	-	14	-	15	-
16	9	17	10	18	7
19	9	20	10	21	6
22	6	23	7	24	9
25	8	26	1	27	1
28	2	29	3	30	4
31	5	32	6	33	13
34	13	35	14	36	14
37	17, 18, 20, 25, 26, 27	38	21	39	17, 18, 25, 26, 27
40	17, 18, 25, 26, 27	41	29	42	19
43	18	44	29	45	-
46	11	47	11	48	22
49	23	50	24	51	28
52	30	53	31	54	11
55	11	56	11	57	11
58	11	59	13	60	13
61	13	62	13, 32	63	12
64	12	65	12	66	12
67	12, 13	68	12	69	35
70	-	71	33, 35, 37, 39	72	-
73	-	74	-	75	36
76	-	77	32	78	34
79	34	80	38	81	29
82	29	83	29	84	29
85	29	86	29	87	29
88	29	89	29	90	-
91	-	92	-	93	40
94	41	95	-	96	-
97	-	98	-	99	-
100	-	101	-	102	-

The following matrix lists all CNRs from the URD implemented by each SRQ.

SRD	CNR	SRD	CNR	SRD	CNR
1	-	2	-	3	-
4	13	5	-	6	
7	-	8	-	9	-
10	-	11	-	12	-
13	-	14	-	15	-
16	-	17	-	18	-
19	-	20	-	21	-
22	-	23	-	24	-
25	-	26	-	27	-
28	-	29	-	30	-
31	-	32	-	33	-
34	-	35	-	36	-
37	-	38	-	39	-
40	-	41	-	42	-
43	-	44	-	45	-
46	-	47	-	48	-
49	-	50	-	51	-
52	-	53	-	54	-
55	-	56	-	57	-
58	-	59	-	60	-
61	-	62	-	63	-
64	-	65	-	66	-
67	-	68	-	69	-
70	-	71	-	72	-
73	-	74	-	75	-
76	-	77	-	78	-
79	-	80	-	81	-
82	-	83	-	84	-
85	-	86	-	87	-
88	-	89	-	90	10
91	11	92	12	93	-
94	-	95	1	96	2
97	3	98	4	99	5
100	7	101	8	102	9