TU/e

Project Fingerpaint

URD

User Requirements Document

Authors:
Tessa Belder
Lasse Blaauwbroek
Thom Castermans
Roel van Happen
Benjamin van der Hoeven
Femke Jansen
Hugo Snel

Junior Management:
Simon Burg
Areti Paziourou
Luc de Smet

Senior Management:
Mark van den Brand
Lou Somers

Advisor:
Ion Barosan

Customer:
Patrick Anderson

April 25, 2013

Abstract

This document describes the User Requirements of Fingerpaint . The User Requirements Document (URD) is based on the ESA standard for software development, as set by the European Space Agency (ESA) [1].

Contents

Document Status Sheet

General

Document title:	User Requirements Document
Identification:	URD0.1
Author:	Tessa Belder, Roel van Happen, Benjamin van der Hoeven, Femke Jansen, Hugo Snel
Document status:	Draft

Document history

Version	*Date*	*Author*	*Reason of change*
0.1	24-Apr-2013	Tessa Belder Roel van Happen Benjamin van der Hoeven Femke Jansen Hugo Snel	Initial version.

Document Change Records since previous issue

General

Datum:	24-Apr-2013
Document title:	User Requirements Document
Identification:	URD0.1

Changes

Page	*Paragraph*	*Reason to change*
-	-	Initial version.

Chapter 1

Introduction

1.1 Purpose

The user requirements document (URD) contains the requirements for Fingerpaint . These requirements are a negotiated agreement between prof.dr.ir. P.D. Anderson and Group Fingerpaint . All of the listed requirements, and only these, will be implemented in Fingerpaint, according to their priorities. Any changes to these requirements require the full consent of both parties.

1.2 Scope

Fingerpaint is an application which visualises fluid mixing on a mobile device. Users can define the initial concentration, as well as manipulate the mixing protocol. The resulting fluid distribution can be stored and analyzed by the user for comparison purposes.

1.3 List of definitions

2IP35	The Software Engineering Course
Client	prof.dr.ir. P.D. Anderson
CM	Configuration Manager
CPR	Capability Requirement
CNR	Constraint Requirement
TU/e	Eindhoven University of Technology
SEP	Software Engineering Project
SR	Software Requirements
SRD	Software Requirements Document
TBC	To Be Confirmed
TBD	To Be Defined

1.4 List of references

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

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

1.5 Overview

The remainder chapters describe the user requirements in more detail. Chapter 2 gives a general description of

- 2.1 - The relation to other systems,
- 2.2 - The main capabilities,
- 2.3 - Constraint information and justification,
- 2.4 - User charactaristics,
- 2.5 - The operational environment, and
- 2.6 - Assumptions and dependencies.

Chapter 3 gives a detailed list of the system's capabilitiy requirements in section 3.1, and a list of the constraint requirements is given in section 3.2.

Chapter 2

General description

This chapter describes general aspects of the application as requested by the client.

2.1 Product perspective

The aim of this project is to deliver an application that allows the user to easily visualize the mixing of fluids. A user interface should be provided to specify initial details, after which output (including intermediate results) should be shown on the screen. All of this should be possible using an easy to use, attractive interface on a mobile device.

A similar project was initiated around eleven years ago. The result of this project was a MATLAB implementation that achieved a similar goal as our project. However, its user interface is by now outdated, and it is impossible to comfortably use this solution on a mobile device. Part of this original solution was a FORTRAN implementation for computing the necessary matrices. This implementation is still available, and we are to use it as a black box to compute the matrices we need for our solution. The client has several matrices, all denoting various kinds of mixers, which we can use for our implementation.

2.2 General capabilities

The system should be able to simulate the flow and mixing of a number of fluids, given some constraints such as movement of the walls and initial concentration of the fluids. The main user interface should be run on a mobile device, such as an iPhone. On this device, three important aspects of a problem should be specified. The first parameter to be specified is the initial concentration of the fluids in the mixer, which should be indicated by tapping on and dragging over the screen. The second is the protocol for moving the mixer and the size of these steps (for example, first move the upper wall 5 steps to the right, then move the lower wall 0.6 steps to the left). The third parameter to be specified is the When the initial parameters have been set, the computation is offloaded to a server, which computes the flow of the fluids. Intermediate results and the final result are sent back to the mobile device to be shown to the user in a graphical way (i.e. a two-dimensional image of the current separation of the fluids).

A history of past simulations is stored on the device, to compare previous runs with the current. The result of runs should also be exportable to easily sharable formats, such as .png

or .pdf, with support for an alpha channel (to realize transparency). For more interactive results, entire runs should be exportable to animated .png or .gif files.

2.3 General constraints

The user interface should be suitable for mobile devices, so it is easy to visualize the results and show them to other people without much hassle. To make it even easier to quickly demonstrate mixing results to others, the actual computation on the server should not take too long (a couple of seconds at most). We do not want to be locked to one specific type of device, so we have chosen to design a cross-platform solution. To easily share results, it should also be possible to export the result of a mixing run to image files, and entire runs to animated files.

2.4 User characteristics

As documented before, the user of the application should be able to specify initial parameters, and, after the application has sent these off to the server, should be able to view the results. The user can then store these results to reference later, and to export these results to image files with transparency.

2.5 Environment description

The main device for the user interface is the mobile device. We are planning to create a cross-platform solution, which means it will be possible to use the application on various kinds of devices. Examples of supported devices are Apple iPhones and Android phones or tablets. The initial concentration of the fluids, the mixing protocol and the shape of the mixer will be specified on such a device.

As mobile devices typically do not have the power (both processing power and battery capacity) to perform intensive computations, the hard work of computing the matrices will be offloaded to a server. The starting parameters described above will be distributed to the server, which has an efficient FORTRAN implementation to solve the problem. While solving, intermediate results are sent back to the mobile device for displaying.

2.6 Assumptions and dependencies

This section contains some assumptions for the application to function properly.
As a mentioned in the previous section, the application uses the FORTRAN implementation on the server to perform all the calculations. Therefore, we assume this server always answers requests within a few seconds.

Chapter 3

Specific requirements

3.1 Capability requirements

This chapter explicitely states all requirements and constraints off the application to be developed. The final product will be delivered confirm these requirements. Any requirements following from further request will be added here.
The requirements are prioritized using the MoSCoW model [2]. This model assigns one out of four priorities to each requirement:

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.

CPR01	*could have*
Users can set a geometry for the canvas	
CPR02	*must have*
Users can define a initial concentration distribution with black and white	
CPR03	*TBC*
Users can define a initial concentration distribution with more than two different colors	
CPR04	*could have*
Users can choose which colors are used for the initial concentration distribution	
CPR05	*must have*
Users can define a mixing protocol for a rectangular geometry as a sequence of movements of the upper and lower walls	
CPR06	*could have*
Users can define a mixing protocol for a non-rectangular geometry as a sequence of movements that are applicable to the geometry	

CPR07	*must have*
Users can define a step to indicate the timeperiod that each movement from the mixing protocol is applied	
CPR08	*could have*
Users can define a different step for each separate movement in the mixing protocol	
CPR09	*must have*
Users can view an image of the endresult of applying the mixing protocol on the initial concentration distribution	
CPR10	*should have*
Users can save the image from CPR09 locally to their device, without losing transparency (i.e. PNG or GIF format)	
CPR11	*should have*
Users can remove previously stored images from their device	
CPR12	*could have*
Users can view an animation of applying the mixing protocol on the initial concentration distribution	
CPR13	*could have*
Users can save the animation from CPR12 locally to their device, without losing transparency (i.e. APNG or AGIF format	
CPR14	*could have*
Users can remove previously stored animations from their device	
CPR15	*should have*
Users can view the mixing performance of the mixing protocol in a graph	
CPR16	*should have*
Users can save the performance results locally on their device	
CPR17	*should have*
Users can retrieve the performance results that are stored locally on their device	
CPR18	*should have*
Users can retrieve performance results from multiple mixing protocols simultaneously, after which they are depicted in one graph	
CPR19	*should have*
Users can remove performance results that are stored on their device	

3.2 Constraint requirements

This section contains all constraint requirements for the application. In order to keep this section brief, some of these requirements contain references to requirements from the previous section.

CNR01	*must have*
The application has input interface and an output interface	
CNR02	*could have*
The application has a history interface	
CNR03	*could have*
The application has a settings interface	
CNR04	*must have*
The input interface provides the functionality described in requirements CPR02, CPR05, CPR07.	
CNR05	*should have*
The input interface provides the functionality described in requirements CPR03.	
CNR06	*could have*
The input interface provides the functionality described in requirements CPR01, CPR04, CPR06, CPR08.	
CNR07	*must have*
The output interface provides the functionality described in requirement CPR09.	
CNR08	*should have*
The output interface provides the functionality described in requirements CPR10, CPR15, CPR16.	
CNR09	*could have*
The output interface provides the functionality described in requirements CPR12, CPR13.	
CNR10	*should have*
The history interface provides the functionality described in requirements CPR11, CPR17, CPR18, CPR19.	
CNR11	*could have*
The history interface provides the functionality described in requirement CPR14.	
CNR12	*must have*
The application runs on devices runnning on iOS versions 5 and higher.	
CNR13	*should have*
The application runs on devices runnning on Android version 4.0 and higher	
CNR14	*could have*
The application runs on devices runnning on Windows 8	
CNR15	*must have*
Waiting time between submitting input and receiving output is not longer than 5 seconds	
CNR16	*should have*
Waiting time between submitting input and receiving output is not longer than 3 seconds	
CNR17	*could have*
Waiting time between submitting input and receiving output is not longer than 1 second	