Eindhoven University of Technology

OpenACCEL

User Requirements Document

Authors:
Edward Brinkmann (0720803)
Carl van Dueren den Hollander (0743652)
Philippe Denissen (0719003)
Kevin de Ridder (0745581)
Leo van Gansewinkel (0781263)
Loc Tran (0766987)
Roel Jacobs (0767494)
Roy Stoof (0767157)
Jacco Snoeren (0772658)

Project Manager:
Erikos Alkalai

Quality Manager:
Duc Luu

Senior Management:
Lou Somers

Advisor:
Erik Scheffers

Customer:
Kees van Overveld

May 9, 2014
Abstract

This document contains the User Requirements Document (URD) for OpenACCEL which is developed by team OpenACCELL and will be used by all Bachelor College students to help them get an understanding of mathematical modeling. OpenACCEL is a web-based application which is part of the Software Engineering Project (2IP35) at Eindhoven University of Technology.

This document complies with the Software Requirements Document (SRD) from the Software Engineering Standard, as specified by the European Space Agency (ESA) [1].
Contents

1 Introduction .......................................................... 5
  1.1 Purpose ......................................................... 5
  1.2 Scope ......................................................... 5
  1.3 List of Definitions & Abbreviations ............................. 5
     1.3.1 Definitions .............................................. 5
     1.3.2 Abbreviations .......................................... 6
  1.4 List of references ............................................. 6
  1.5 Overview .................................................... 6

2 General Description .................................................. 8
  2.1 Product perspective .......................................... 8
  2.2 General capabilities ......................................... 8
     2.2.1 Edit/Run script ...................................... 8
     2.2.2 Help/Demo ............................................. 8
     2.2.3 IO/Edit ................................................ 9
     2.2.4 Analysis ............................................... 9
     2.2.5 Genetic Optimization ............................... 9
  2.3 General constraints ........................................... 9
  2.4 User characteristics ......................................... 9
     2.4.1 Students ............................................... 9
     2.4.2 Teachers .............................................. 10
     2.4.3 Administrators ..................................... 10
     2.4.4 Source Code Contributors ......................... 10
  2.5 Environment Description .................................. 10
  2.6 Assumptions and dependencies ............................... 11

3 Specific Requirements ............................................. 12
  3.1 Capability requirements ................................... 12
     3.1.1 General ................................................. 12
     3.1.2 ACCEL language support ............................ 13
     3.1.3 Model Construction ................................. 13
     3.1.4 Model execution .................................... 14
     3.1.5 Help & demo ......................................... 15
     3.1.6 IO/Edit tab ......................................... 15
     3.1.7 Analysis tab ....................................... 16
     3.1.8 Genetic Optimization tab ......................... 16
Document Status Sheet

Document status overview

General
Document title: User Requirements Document
Identification: URD1.4.pdf
Author: K.A. de Ridder, E.K. Brinkmann, T.L. Tran
Document status: Final

Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Reason of change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>25-04-2014</td>
<td>T.L. Tran</td>
<td>First draft version</td>
</tr>
<tr>
<td>0.2</td>
<td>30-04-2014</td>
<td>E.K. Brinkmann</td>
<td>Second draft based on customer feedback</td>
</tr>
<tr>
<td>0.3</td>
<td>02-05-2014</td>
<td>E.K. Brinkmann</td>
<td>New version based on feedback from customer, project managers, advisor and internal reviews</td>
</tr>
<tr>
<td>1.0</td>
<td>06-05-2014</td>
<td>T.L. Tran, E.K. Brinkmann</td>
<td>New version based on feedback from customer, for management review.</td>
</tr>
<tr>
<td>1.1</td>
<td>06-05-2014</td>
<td>E.K. Brinkmann, K.A. de Ridder</td>
<td>Version for customer approval. Based on internal review and feedback from advisor.</td>
</tr>
<tr>
<td>1.2</td>
<td>07-05-2014</td>
<td>E.K. Brinkmann</td>
<td>Incorporated feedback from customer on v1.0</td>
</tr>
<tr>
<td>1.3</td>
<td>08-05-2014</td>
<td>T.L. Tran</td>
<td>Modify List of built-in AC-CEL functions</td>
</tr>
<tr>
<td>1.4</td>
<td>09-05-2014</td>
<td>T.L. Tran</td>
<td>Modify use cases</td>
</tr>
</tbody>
</table>
Document Change Records since previous issue

General
Datum: May 9, 2014
Document title: User Requirements Document
Identification: URD1.4.pdf

Changes

<table>
<thead>
<tr>
<th>Page</th>
<th>Paragraph</th>
<th>Reason to change</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>A.1.3</td>
<td>Modified precondition and added extra steps</td>
</tr>
<tr>
<td>22</td>
<td>A.7</td>
<td>Modified precondition and added extra steps</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Purpose

This User Requirements Document (URD) contains the requirements for OpenACCEL. These requirements are a negotiated agreement between the client, Kees van Overveld, and the OpenACCEL team. All of the requirements, and only these, will be implemented in OpenACCEL according to their priorities. Any further changes will require negotiation between both parties.

1.2 Scope

OpenACCEL is an open-sourced and re-engineered application based on ACCEL. It is being developed by team OpenACCEL at the TU/e. The application is mainly used by students and teachers during the Bachelor College course Introduction to Modeling as a tool to aid students in understanding and building mathematical models.

1.3 List of Definitions & Abbreviations

1.3.1 Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEL</td>
<td>A web-based modeling application which is currently hosted at [5]</td>
</tr>
<tr>
<td>OpenACCEL</td>
<td>The new and improved application which will be developed by team OpenACCEL, based on ACCEL</td>
</tr>
<tr>
<td>Bachelor College</td>
<td>The new format of Bachelor education introduced by the TU/e in 2012</td>
</tr>
<tr>
<td>Just-fit</td>
<td>Fitting the graph view to exactly fit the plot</td>
</tr>
<tr>
<td>Convenient-fit</td>
<td>Fitting the graph view to the nearest larger integer, such that the grid width and height are integers</td>
</tr>
<tr>
<td>GPLv3</td>
<td>Open source license [6].</td>
</tr>
<tr>
<td>Model</td>
<td>A set of definitions consisting of functional relations between input and output quantities. A model is constructed and/or executed in OpenACCEL</td>
</tr>
<tr>
<td>Script</td>
<td>The underlying lines of code of the model, as inputted by the user in OpenACCEL. Sometimes interchangeable with model</td>
</tr>
</tbody>
</table>
Descartes | Set of functions used for displaying graphics within ACCEL
Quantity | A synonym for variable which may have a unit
Quantity categories | A Quantity belongs to one of four possible categories:

Cat. I: input variables which can be changed by the user at runtime.

Cat. II: output variables whose value depends on variables in the other three categories

Cat. III: input variables whose value cannot be influenced by the user

Cat. IV: variables that depend on category I, III and IV variables and are used as input to category II variables.

1.3.2 Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPLv3</td>
<td>General Public License version 3</td>
</tr>
<tr>
<td>URD</td>
<td>User Requirements Document</td>
</tr>
<tr>
<td>TU/e</td>
<td>Eindhoven University of Technology</td>
</tr>
<tr>
<td>SPEA</td>
<td>Strength Pareto Evolutionary Algorithm</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
</tbody>
</table>

1.4 List of references


1.5 Overview

The remainder of this document consists of a general discussion in chapter 2, followed by an extensive list of requirements in chapter 3.

Sections 2.1 and 2.2 discuss the product perspective and general capabilities respectively.
Section 2.3 discusses general constraints OpenACCEL has to comply with. Section 2.4 describes the different user groups that will be using OpenACCEL and section 2.5 describes the environment in which OpenACCEL will operate.

Chapter 3 lists all requirements, together with their respective priorities, divided in logical categories.
Chapter 2

General Description

2.1 Product perspective

The Introduction to Modeling course is a core course in the Bachelor College at the TU/e, in which over 2000 students take part. To support this course, a web application named ACCEL was created that allows students to model real-life problems using a systematic approach. However, ACCEL has evolved over time to a point at which its current architecture is ill-suited to support such a large userbase. Team OpenACCEL will therefore rebuild ACCEL from the ground up into a more reliable, maintainable and reusable platform that can continue to support its growing audience. OpenACCEL is intended to be made available under an open source license at a future point in time.

2.2 General capabilities

OpenACCEL will have the capabilities described below, divided into several categories corresponding to the different pages/tabs in ACCEL’s user interface.

2.2.1 Edit/Run script

The actual construction of models in OpenACCEL will mainly be done on the Edit/Run script page. Users can add definitions to, and remove them from, a model. Each time a new definition is entered, it may contain quantities which have not yet been defined. The system keeps track of those undefined quantities in a todo-list. Once this list is empty, the system will display the output of the model. The system will allow the user to manipulate input quantities during runtime.

2.2.2 Help/Demo

OpenACCEL allows users to view documentation about the functional programming language used in OpenACCEL on the Help/Demo page. It contains the notation of all built-in functions and other features, and provides an extensive explanation of them. This page also contains a list of demo scripts made by other students and teachers. The user can run these demo scripts in OpenACCEL and modify them to his/her liking.
2.2.3 IO/Edit

OpenACCEL includes a textfield in which the user can view and edit the entire model. The model code contains all model definitions together with optional comments and annotations. Changes made in this textfield will automatically propagate to the model.

2.2.4 Analysis

OpenACCEL comes with intuitive fully automatic sensitivity analysis [5], showing the relation of output quantities as functions of input quantities. The system allows the user to select input and output quantities and will show a plot of the relation between the two.

2.2.5 Genetic Optimization

When a model has been defined and contains quantities that are registered for Pareto optimization [2], OpenACCEL allows the user to optimize the model using SPEA.

2.3 General constraints

OpenACCEL should be able to support the large number of students that will follow the Modeling course and should therefore be reliable and robust. It should run in all w3 compliant browsers [3]. Also, videolectures were recorded containing instructions of how to use ACCEL to accomplish certain tasks, and these videolectures will not/cannot be redone. Because of this, OpenACCEL should be backward-compatible with ACCEL to the extent that all help documentation and videolectures will still apply. Finally, OpenACCEL should be modular and easy to maintain and extend, as it is intended to be made open source at some future point in time and will hopefully be extended on a regular basis.

2.4 User characteristics

This section is about which users will make use of the OpenACCEL application. These users are divided into the following four groups:

1. TU/e Students following the Introduction to Modeling course
2. Teachers
3. Administrators
4. Source Code Contributors

2.4.1 Students

The main users of OpenACCEL will be the students following the course Introduction to Modeling at the TU/e. Students will use OpenACCEL during the course to construct mathematical models and perform simulations and analysis on them. They can share models with each other either privately by copying the code of the model directly or publicly by publishing models to the website. Students will also be required to use OpenACCEL for solving assignments and exercises and to inspect example models provided by the lecturers.
2.4.2 Teachers

The second largest user group of OpenACCEL consists of the teachers of the Introduction to Modeling course. The responsible lecturers will use OpenACCEL to teach the students about mathematical modeling and will let them make assignments and exercises with it. They can provide demo scripts with instructions of how to model certain things and require students to upload solutions to the website for grading.

2.4.3 Administrators

The administrators of OpenACCEL are responsible for maintaining the demo scripts on the server and for ensuring that OpenACCEL is available to its users.

2.4.4 Source Code Contributors

Since OpenACCEL is meant to be made open source, another group of users is future contributors to the source code. Contributors will inspect the code and architecture of OpenACCEL and hopefully improve it by making contributions.

Figure 2.1 displays the interaction between the different users and OpenACCEL within it’s environment.

2.5 Environment Description

We will now analyse the environment that OpenACCEL operates in, in which we consider OpenACCEL as a black box. The domain of a software system is defined as the area of business and technology in which it operates. OpenACCEL’s domain is that of a web-based tool for use by both students and teachers in the context of a Bachelor College course at Eindhoven University of Technology. As mentioned in 2.4, students and teachers form OpenACCELs largest user base.

OpenACCEL will run on a server provided by the customer and will be waiting for requests. OpenACCEL as a web-application will consist of different files which are served to clients in response to different requests at different points in time. Model construction, compilation and execution all take place solely in the browser of the user, without server-side intervention or control. Uploading and downloading scripts will require communication with a server that stores these scripts. There is no further communication with other systems.
2.6 Assumptions and dependencies

The customer will take care of hosting OpenACCEL and making it available to its target audience. Furthermore, he will ensure that all demo scripts required are available in a directory on a server accessible to OpenACCEL.
Chapter 3

Specific Requirements

In this chapter all requirements and constraints of the product to be developed are specifically stated. The product will adhere to these requirements. Furthermore any requirements resulting from additional requests are included here.

The requirements are based on both the use cases which are analyzed in appendix A and on meetings with the customer. For prioritizing the specific requirements for the program, the MoSCoW model will be used [4].

The capital letters in MoSCoW stand for:

M  Must have, these requirements are essential for the product
S  Should have, these requirements are not critical for the product to work, but are nearly as important as the must haves, meaning they must be implemented if at all possible.
C  Could have, requirements which are not critical to the product’s success. If they can be implemented with little development costs, they can increase the Clients satisfaction.
W  Won’t have, these requirements will not be implemented in this project. However, they would be nice to have in future versions of the product.

3.1 Capability requirements

The capability requirements are subdivided into categories corresponding to the different tabs in ACCEls user interface.

3.1.1 General

These are general requirements not belonging to a specific section of the user interface.

| UCR1 | Summary screen of ACCEL functionality | must have |
| UCR2 | Visual feedback for clickable items | could have |
| UCR3 | Ask for confirmation when the user is about to leave the page | won’t have |
### 3.1.2 ACCEL language support

This section contains requirements related to the ACCEL language constructs that have to be supported for use within the definitions of models.

<table>
<thead>
<tr>
<th>UCR4</th>
<th>Defining (the value of) a quantity with a functional expression</th>
<th>must have</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCR5</td>
<td>Operator precedence</td>
<td>must have</td>
</tr>
<tr>
<td>UCR6</td>
<td>All currently supported ACCEL functions listed in Appendix B</td>
<td>must have</td>
</tr>
<tr>
<td>UCR7</td>
<td>User defined functions</td>
<td>must have</td>
</tr>
<tr>
<td>UCR8</td>
<td>Recursion with memoization</td>
<td>must have</td>
</tr>
<tr>
<td>UCR9</td>
<td>Recursive auto-mapping</td>
<td>must have</td>
</tr>
<tr>
<td>UCR10</td>
<td>Conditional expressions: if(guard, exp1, exp2) and cond(guard, exp1, exp2)</td>
<td>must have</td>
</tr>
<tr>
<td>UCR11</td>
<td>Quantified expressions</td>
<td>must have</td>
</tr>
<tr>
<td>UCR12</td>
<td>Aggregation with numerical indexing ([ val1, val2, ... ]) and indexing using key-value pairs ([ key1:val1, key2:val2, ... : ... ])</td>
<td>must have</td>
</tr>
<tr>
<td>UCR13</td>
<td>De-referencing with the dot notation for named properties (a.b for property b of quantity a)</td>
<td>must have</td>
</tr>
<tr>
<td>UCR14</td>
<td>De-referencing with the index notation for named and numbered properties (a[b] for property b of quantity a, where b is a number or string key)</td>
<td>must have</td>
</tr>
<tr>
<td>UCR15</td>
<td>Access of historical values of quantities with the history operator</td>
<td>must have</td>
</tr>
<tr>
<td>UCR16</td>
<td>Iterative execution of model with user-specified number of executions</td>
<td>must have</td>
</tr>
</tbody>
</table>

### 3.1.3 Model Construction

This section contains requirements related to constructing a mathematical model within OpenACCEL.

<table>
<thead>
<tr>
<th>UCR17</th>
<th>Display a list of all definitions of the model</th>
<th>must have</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCR18</td>
<td>Automatically label quantity definitions with the category to which the defined quantity belongs (I, II, III and IV)</td>
<td>must have</td>
</tr>
<tr>
<td>UCR19</td>
<td>Delete a quantity definition from the model</td>
<td>must have</td>
</tr>
<tr>
<td>Requirement</td>
<td>Requirement Details</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td><strong>UCR20</strong></td>
<td>Modify a definition</td>
<td></td>
</tr>
<tr>
<td><strong>UCR21</strong></td>
<td>Clicking on the name of a user-defined quantity displays its definition</td>
<td></td>
</tr>
<tr>
<td><strong>UCR22</strong></td>
<td>Clicking on the name of a built-in ACCEL function displays its help page</td>
<td></td>
</tr>
<tr>
<td><strong>UCR23</strong></td>
<td>Undefined quantities appearing in the model are displayed in a todo-list</td>
<td></td>
</tr>
<tr>
<td><strong>UCR24</strong></td>
<td>Create a new, empty model, discarding any existing definitions of the current model</td>
<td></td>
</tr>
<tr>
<td><strong>UCR25</strong></td>
<td>Should have Display dependencies between quantities when clicking on the definition of a quantity</td>
<td></td>
</tr>
<tr>
<td><strong>UCR26</strong></td>
<td>Should have Display useful error messages during model compilation when exceptions occur</td>
<td></td>
</tr>
<tr>
<td><strong>UCR27</strong></td>
<td>Must have Changes made within the Edit/run script tab are automatically made visible in the IO/Edit tab</td>
<td></td>
</tr>
</tbody>
</table>

### 3.1.4 Model execution

These requirements have to do with running a finished model in the OpenACCEL runtime.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Requirement Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UCR28</strong></td>
<td>Must have Automatically (re)compile model when the model is extended or modified</td>
</tr>
<tr>
<td><strong>UCR29</strong></td>
<td>Must have Automatically run the script when the todo-list is empty</td>
</tr>
<tr>
<td><strong>UCR30</strong></td>
<td>Must have A model can only run when the to-do list is empty</td>
</tr>
<tr>
<td><strong>UCR31</strong></td>
<td>Must have Execution of the script can be paused when running</td>
</tr>
<tr>
<td><strong>UCR32</strong></td>
<td>Must have Execution of the script can be resumed when paused</td>
</tr>
<tr>
<td><strong>UCR33</strong></td>
<td>Must have Specify the number of iterations that the model should make before presenting the result</td>
</tr>
<tr>
<td><strong>UCR34</strong></td>
<td>Must have When the todo-list is empty, a list of category 2 (output) quantities and their current values is displayed</td>
</tr>
<tr>
<td><strong>UCR35</strong></td>
<td>Must have Create and display controls for all category 1 quantities that are modifiable by the user</td>
</tr>
<tr>
<td><strong>UCR36</strong></td>
<td>Must have When the model is compiled and contains a plot() function, display a Descartes canvas containing the graphical output</td>
</tr>
<tr>
<td><strong>UCR37</strong></td>
<td>Should have Runtime exceptions caused by SPEA shall abort the SPEA process</td>
</tr>
<tr>
<td><strong>UCR38</strong></td>
<td>Should have</td>
</tr>
</tbody>
</table>
Recoverable runtime exceptions that occur during interactive model execution, shall not abort execution

**UCR39** should have
Runtime exceptions during interactive model execution from which the system can *not* recover, shall stop model execution

### 3.1.5 Help & demo

These requirements have to do with browsing through help articles and browsing and loading demo scripts

**UCR40** could have
Browse through help articles, divided into different categories

**UCR41** could have
View all help articles in a specific category

**UCR42** could have
Show the contents of a single help article

**UCR43** could have
Search within the titles of help articles

**UCR44** could have
Ask for help via email. An email is generated containing some predefined content and sent to a pre-configured address

**UCR45** could have
Clicking on the name of another ACCEL function within a help article displays the category containing that function

**UCR46** should have
Display a list of all public demo scripts

**UCR47** should have
Select and start (load and compile) a script from the list from **UCR46**

**UCR48** could have
Submit the currently loaded script to the server and obtain a link to it

### 3.1.6 IO/Edit tab

These requirements have to do with viewing and editing the script of the model directly.

**UCR49** must have
There is a text field containing all definitions of the currently loaded model

**UCR50** should have
Allow comments in scripts

**UCR51** must have
Allow editing of script using the text field

**UCR52** must have
Changes made within the ‘IO/Edit’ tab are propagated to the ‘Edit/Run Script’ tab when the user leaves the ‘IO/Edit’ tab

**UCR53** should have
Show/hide values of all quantities in the current state
3.1.7 Analysis tab

This section contains requirements related to performing analysis on quantities in a model.

- UCR55: Visually represent a function $y = f(x)$
- UCR56: Visually represent a function $z = f(x, y)$ as contour plots
- UCR57: Automatically set the ranges of the selected input and output quantities, allowing manual override
- UCR58: Automatically scale the graph using just-fit
- UCR59: Automatically scale the graph using convenient-fit
- UCR60: Show a list of condition numbers for relative error propagation (numerical estimation)

3.1.8 Genetic Optimization tab

These requirements have to do with performing genetic optimization on models containing Pareto functions.

- UCR61: Initialize SPEA on a model containing Pareto quantities (ParetoMax and ParetoMin)
- UCR62: Calculate and show a Pareto plot
- UCR63: Calculate next generations of SPEA
- UCR64: Set size of populations
- UCR65: Set number of populations to be calculated
- UCR66: Show values of category I and II quantities, for a solution selected in the plot.
- UCR67: Post Tune optimization
- UCR68: Zoom in on plot
- UCR69: Zoom out of plot
- UCR70: Zoom to fit plot
3.1.9  Simulation tab

These requirements have to do with viewing the simulation of a model containing a `plot()` function.

- **UCR72**  
  Using Descartes, show model in a graphical view

- **UCR73**  
  Show all user input controls specified in the model

3.1.10 Network tab

This section contains requirements related to viewing the Directed Acyclic Graph structure underlying the model.

- **UCR74**  
  Show directed acyclic graph of the model

- **UCR75**  
  Clicking on a node shows the definition and comment of the node

- **UCR76**  
  Clicking on a node highlights edges connected to that node

- **UCR77**  
  Number of levels of highlighted edges can be set using a slider

- **UCR78**  
  The graph can be reordered by dragging nodes

3.2 Constraint requirements

3.2.1 Interfaces

- **UCR79**  
  OpenACCEL must be compatible with the visual instructions in all current videolectures

- **UCR80**  
  OpenACCEL must be able to list, load and run all current demo and hidden scripts from a server provided by the customer

- **UCR81**  
  OpenACCEL must be able to load demo scripts directly via a URL parameter

- **UCR82**  
  OpenACCEL must be able to load hidden scripts directly via a URL parameter

- **UCR83**  
  OpenACCEL must be able to display a help topic directly via a URL parameter

- **UCR84**  
  The ‘verzinEenList’-function shall not be altered without permission from customer
3.2.2 Adaptability

**UCR85**
OpenACCEL must be designed in a modular fashion

3.2.3 Availability

**UCR86**
OpenACCEL must be free of charge

**UCR87**
OpenACCEL must be freely available

3.2.4 Standards

**UCR88**
All software components in OpenACCEL must be available under GPLv3 license

**UCR89**
All software components in OpenACCEL must, at the time of publication of this document, be maintained by an active community.

**UCR90**
If in doubt about the two requirements above, usage of software components must be approved by the customer.

**UCR91**
OpenACCEL must run and display correctly in the latest version - at the time of publication of this document - of chrome and safari.

**UCR92**
OpenACCEL must run and display correctly in the latest version - at the time of publication of this document - of firefox.

**UCR93**
If the users browser is not supported, OpenACCEL must notify the user and refuse to run.

**UCR94**
OpenACCEL itself must be available under the GPLv3 license.

3.2.5 Resources

**UCR95**
OpenACCEL must be at least as fast as ACCEL

3.2.6 Time scales

**UCR96**
If model compilation takes longer than 2 seconds, ask the user if he wishes to continue waiting or wants to abort compilation
Appendix A

Use cases

The following use cases describe sequences of actions that an actor can perform in order to compete a given task.

A.1 Write/Edit a model

A.1.1 Adding new definition

Goals: To write a new definition to a script.
Preconditions:
Summary: A new definition is added to the script.
Priority: Must have
Steps:

<table>
<thead>
<tr>
<th>Actor actions</th>
<th>System response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click Edit/Run script tab.</td>
<td>2. Changes view to Edit/Run script.</td>
</tr>
<tr>
<td>3. Enter definition in text field.</td>
<td>4. Shows the added definition in the list of definitions.</td>
</tr>
</tbody>
</table>

A.1.2 Edit definition

Goals: To edit a definition of a script.
Preconditions: At least one definition in a script.
Summary: A new definition is added to the script.
Priority: Must have
Steps:

<table>
<thead>
<tr>
<th>Actor actions</th>
<th>System response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click Edit/Run script tab.</td>
<td>2. Changes view to Edit/Run script.</td>
</tr>
<tr>
<td>3. Click definition in the list.</td>
<td>4. Shows the definition in the text field.</td>
</tr>
<tr>
<td>5. Change the definition.</td>
<td>6. Modifies the definition in the list.</td>
</tr>
</tbody>
</table>

Alternatives: for step 3, the user can also create a new definition with the same quantity, this will overwrite the definition of that quantity.
A.1.3 Delete definition

**Goals:** To delete a definition of a script.

**Preconditions:** The model contains at least one definition.

**Summary:** A definition is deleted from the script.

**Priority:** Must have

**Steps:**

**Actor actions:**
1. Click Edit/Run script tab.
2. Changes view to Edit/Run script.
3. Delete a definition from the list of definitions.
4. Deletes the definition from the model.

**System response:**
If the quantity was used in other definitions, add quantity to todo-list.

A.1.4 Adding multiple definitions

**Goals:** To add multiple definitions to a script.

**Preconditions:**

**Summary:** Multiple definitions added to the script.

**Priority:** Must have

**Steps:**

**Actor actions:**
1. Click IO/Edit tab.
2. Changes view to IO/Edit tab.
3. Enter definitions in text field.
4. Definitions added to text field.
5. User navigates to another tab
6. Definitions added to model.

**System response:**

**Alternatives** The user can choose to edit definitions in the text field when they are already present.

A.2 Loading a model using a demo script

**Goals:** To load a predefined demo script.

**Preconditions:** A demo script must be present.

**Summary:** A specified demo script is loaded.

**Priority:** Must have

**Steps:**

**Actor actions:**
1. Click on the Help/Demo tab.
2. Changes view to Help/Demo.
3. Choose a demo script in right list.
4. Selects script and changes view to Edit/Run script.

**System response:**

**Alternatives:** In step 3, the user can click on the buttons below the list to navigate through the list.

A.3 Loading a model stored on the clients device

**Goals:** To load a local model into OpenACCEL.

**Preconditions:** A model must be present on the clients device.

**Summary:** A model is loaded into OpenACCEL.
**A.4 Uploading a script to the server**

**Goals:** Adding a script to the database of OpenACCEL.

**Preconditions:** An executable script must be loaded.

**Summary:** A script is added to the database of OpenACCEL.

**Priority:** Could have

**Steps:**

*Actor actions:*  
1. Click on the Help/Demo tab.  
2. Click on the submit button.  
3. Fill in filename.  

*System response:*  
2. Changes to Help/Demo.  
4. Shows a window where user can fill in filename.  
6. Shows a link to the script.

**A.5 Loading a script using a link**

**Goals:** Loading a custom script from the database of OpenACCEL

**Preconditions:** A link to the script using use case A.4.

**Summary:** A script is loaded.

**Priority:** Could have

**Steps:**

*Actor actions:*  
1. Paste the link in the web browser.  

*System response:*  
2. Loads the script and shows Edit/Run script.

**Alternative:** In step 1, the user can manipulate the address of the web page of OpenACCEL when the user knows the exact script name.

**A.6 Manipulating model input values**

**Goals:** Manipulating definitions that are input quantities.

**Preconditions:** A script containing cat.I quantities.

**Summary:** The output of the model will change according to the changes to the values of the input quantities.

**Priority:** Must Have

**Steps:**
Actor actions: System response:
1. Click Edit/Run script tab. 2. Changes view to Edit/Run script, and shows a list of possible input quantities that can be changed.
3. Change the value using the slider. 4. Changes the value and updates the model.

Alternative: multiple alternatives for step 3, apart from a slider input quantities can also have check boxes, input text fields or buttons.

A.7 Executing model

Goals: A loaded script can be executed.
Preconditions: A.2 or A.3 should be satisfied
Summary: When a valid script is loaded.
Priority: Must have
Steps: Actor actions: System response:
Actor actions: System response:
1. Click IO/Edit tab. 2. Changes view to IO/Edit tab.
3. Click on the check current values button if it’s not enabled (works like a toggle). 4. Shows a line of comments below quantities that satisfy the precondition with the values of those quantities.

A.8 Checking current values of quantities

Goals: Showing the current values of quantities of a model.
Preconditions: A script with quantities which do not depend on other quantities.
Summary: A line with the current value of a quantity.
Priority: Should have
Steps: Actor actions: System response:

A.9 Checking possible units of quantities

Goals: Showing units of quantities.
Preconditions: Script where input quantities all have a unit and quantities that depend on them.
Summary: Shows the unit of quantities that depend on the input quantities.
Priority: Could have
Steps:
A.10 Displaying and modifying network

Goals: Manipulating network of a model.
Preconditions: A working script.
Summary: Shows the unit of quantities that depend on the input quantities.
Priority: Could have
Steps:

1. Click network tab. 2. Changes view to network tab.
3. Click and drag node in the network. 4. Highlights edges of the node and moves the node in the network.

A.11 Optimizing parameters using SPEA

Goals: Optimizing the input quantities of a model
Preconditions: A script containing definitions with Pareto functions.
Summary: Changed ranges of input variables according to the optimization.
Priority: Should have
Steps:

1. Click on Genetic optimization tab. 2. Changes view to genetic optimization.
3. Click on initialize SPEA. 4. Show plot with points and point with best values.
5. Click on a point in plot. 6. Shows a report of values for the quantities.
7. Click on update script. 8. Modifies the ranges of the input quantities according to the selected point.

A.12 Analysing relations between dependend quantities

Goals: Showing a plot of depended quantities.
Preconditions: An executable script must be loaded with depended quantities.
Summary: A plot is shown with the relation between the selected quantities.
Priority: Should have
Steps:

1. Click on the analysis tab. 2. Changes view to the analysis tab.
3. Choose a quantity from the arguments list and a different quantity from the results list. 4. Shows either a plot of the dependency between the argument quantity and the results quantity or tells the user the quantities are not depended.
A.13  Asking for help

A.13.1  Via email

Goals: Asking for help via email.
Preconditions: 
Summary: By clicking the “asking for help via email”-button the email environment will pop-up for setting up an email.
Priority: Must have
Steps:

<table>
<thead>
<tr>
<th>Actor actions:</th>
<th>System response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click the Help/Demo tab.</td>
<td>2. Changes view to the Help/Demo tab.</td>
</tr>
<tr>
<td>3. Click the “Asking for help via email”-button.</td>
<td>4. A mail-to window will pop-up.</td>
</tr>
</tbody>
</table>

A.13.2  Via help text

Goals: Asking for help via textarea
Preconditions: The list should have at least one function listed.
Summary: By clicking a function a summary of the function will be displayed.
Priority: Must have
Steps:

<table>
<thead>
<tr>
<th>Actor actions:</th>
<th>System response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Clicks on a function from the categories list.</td>
<td>4. All functions belonging to that category are listed.</td>
</tr>
<tr>
<td>5. Clicks on a function from the details list.</td>
<td>6. A text area with a summary about the function is shown.</td>
</tr>
</tbody>
</table>
# Appendix B

## Built-in ACCEL functions

### B.1 List of definitions

<table>
<thead>
<tr>
<th><strong>absolute value</strong></th>
<th>for a given value if value &gt; 0 then abs(value) = value, else abs(value) = -value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>convolution</strong></td>
<td>For two given functions produce a third function, which is a modified version of the two given functions.</td>
</tr>
<tr>
<td><strong>domain</strong></td>
<td>set of input arguments for which the function is defined</td>
</tr>
<tr>
<td><strong>dot product</strong></td>
<td>For two given vectors the dot product is defined as: $a_1 * b_1 + a_2 * b_2 + \ldots a_n * b_n$</td>
</tr>
<tr>
<td><strong>Euler's number (e)</strong></td>
<td>A mathematical constant approximately equal to 2.71828</td>
</tr>
<tr>
<td><strong>exponential</strong></td>
<td>for a given $n$ take $e^n$</td>
</tr>
<tr>
<td><strong>epoch</strong></td>
<td>January 1st, 1970</td>
</tr>
<tr>
<td><strong>factorial</strong></td>
<td>The product of all integers less than or equal to a given $n$</td>
</tr>
<tr>
<td><strong>inverse matrix</strong></td>
<td>For a given square matrix $A$ find a matrix $A^{-1}$ such that $AA^{-1} = I$ where I is the identity of $A$. That is, $I$ has the same size as $A$ and is filled with zeroes except for the diagonal, which is filled with ones</td>
</tr>
<tr>
<td><strong>linear interpolation</strong></td>
<td>Fitting a curve to a given set of points</td>
</tr>
<tr>
<td><strong>logarithm</strong></td>
<td>For a given $b$ and $y$ find an $x$ such that $x = \log_b(y)$</td>
</tr>
<tr>
<td><strong>matrix</strong></td>
<td>Multi-demensional array</td>
</tr>
<tr>
<td><strong>(zero) padded</strong></td>
<td>For a given vector $x$ extend $x$ with a number on both ends For zero padding, both ends will be extended with zeroes</td>
</tr>
<tr>
<td><strong>poisson distribution</strong></td>
<td>the probability of obtaining exactly $n$ successes in $N$ trials. see: <a href="http://mathworld.wolfram.com/PoissonDistribution.html">http://mathworld.wolfram.com/PoissonDistribution.html</a></td>
</tr>
<tr>
<td><strong>serialized (value)</strong></td>
<td>Modifying data structures and object states into a format that can be stored (for external processing)</td>
</tr>
<tr>
<td><strong>radians</strong></td>
<td>A unit to denote angles</td>
</tr>
<tr>
<td><strong>transpose matrix</strong></td>
<td>For a matrix $X$ and for all rows $i$ and columns $j$ $[X]<em>{ij} = [X^T]</em>{ji}$ where $X^T$ is the transpose of $X$</td>
</tr>
<tr>
<td><strong>vector</strong></td>
<td>one dimensional array</td>
</tr>
</tbody>
</table>
### B.2 Functions

The following table contains the set of functions available in Accel. A more elaborate explanation of the functions can be found on the Accel webpage [5].

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@(var1, var2)</td>
<td>returns the var2’th element of the vector var1</td>
</tr>
<tr>
<td>abs(var)</td>
<td>returns the absolute value of var</td>
</tr>
<tr>
<td>acos(var)</td>
<td>returns a value x such that x = cos(var)</td>
</tr>
<tr>
<td>add(var1, var2)</td>
<td>adds var1 and var2</td>
</tr>
<tr>
<td>and(var1, var2)</td>
<td>returns true if and only if both var1 and var2 are true</td>
</tr>
<tr>
<td>asin(var)</td>
<td>returns a value x such that x = sin(var)</td>
</tr>
<tr>
<td>atan(var)</td>
<td>returns a value x such that x = tan(var)</td>
</tr>
<tr>
<td>atan2(var1, var2)</td>
<td>returns the angle of a line with the horizontal extent var1 and vertical extent var2 in radians</td>
</tr>
<tr>
<td>bin(var1, var2)</td>
<td>returns ( \frac{\text{var2}!}{\text{var1}!} ), that is, the number of ways var2 distinguishable items can be chosen from a set of var1. Where ( x! ) is the factorial of ( x )</td>
</tr>
<tr>
<td>button()</td>
<td>creates and polls a button</td>
</tr>
<tr>
<td>ceil(var)</td>
<td>returns the smallest integer larger than var</td>
</tr>
<tr>
<td>check(val)</td>
<td>creates and polls a checkbox with value val</td>
</tr>
<tr>
<td>cos(var)</td>
<td>returns the cosine of angle var in radians</td>
</tr>
<tr>
<td>debug(var1, var2)</td>
<td>prints the values of var2 preceded by string var1</td>
</tr>
<tr>
<td>divide(var1, var2)</td>
<td>divides var1 with var2</td>
</tr>
<tr>
<td>do(var1, var2)</td>
<td>runs a Javascript fragment var1 on a serialized array var2</td>
</tr>
<tr>
<td>equal(var1, var2)</td>
<td>returns true if and only if var1 is equal to var2</td>
</tr>
<tr>
<td>exp(var)</td>
<td>returns the exponential of var</td>
</tr>
<tr>
<td>factorial(var)</td>
<td>returns the factorial of var</td>
</tr>
<tr>
<td>floor(var)</td>
<td>returns the smallest integer not larger than var</td>
</tr>
<tr>
<td>getTime</td>
<td>returns the time in miliseconds since January 1, 1970 (epoch time)</td>
</tr>
<tr>
<td>greaterThan(var1, var2)</td>
<td>returns true if and only if var1 is greater than var2</td>
</tr>
<tr>
<td>greaterThanEqual(var1, var2)</td>
<td>returns true if and only if var1 is greater or equal to var2</td>
</tr>
<tr>
<td>if(var1, var2, var3)</td>
<td>returns var2 if var1 is true else return var3</td>
</tr>
<tr>
<td>imply(var1, var2)</td>
<td>returns true if and only if var2 is true or if var1 is false</td>
</tr>
<tr>
<td>input(val)</td>
<td>creates and polls an input field with value val</td>
</tr>
<tr>
<td>lessThan(var1, var2)</td>
<td>returns true if and only if var1 is smaller than var2</td>
</tr>
<tr>
<td>lessThanEqual(var1, var2)</td>
<td>returns true if and only if var1 is smaller than or equal to var2</td>
</tr>
<tr>
<td>ln(var1)</td>
<td>returns the base e logarithm of var1</td>
</tr>
<tr>
<td>log(var1)</td>
<td>returns the base 10 logarithm of var1</td>
</tr>
<tr>
<td>max(var1, var2)</td>
<td>returns the larger of var1 and var2</td>
</tr>
<tr>
<td>min(var1, var2)</td>
<td>returns the smaller of var1 and var2</td>
</tr>
<tr>
<td>modulo(var1, var2)</td>
<td>returns the remainder of the division of var1 with var2</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>multiply(var1, var2)</td>
<td>returns the multiplication of var1 with var2</td>
</tr>
<tr>
<td>not(var1)</td>
<td>returns true if and only if var1 is false</td>
</tr>
<tr>
<td>notEqual(var1, var2)</td>
<td>returns true if and only if var1 differs from var2</td>
</tr>
<tr>
<td>or(var1, var2)</td>
<td>returns true if and only if var1 or var2 or both are true</td>
</tr>
<tr>
<td>poisson(var1, var2, var3)</td>
<td>returns the poisson distribution</td>
</tr>
<tr>
<td>pow(var1, var2)</td>
<td>returns var1 to the power var2</td>
</tr>
<tr>
<td>random()</td>
<td>returns a random number between and including 0 and 1</td>
</tr>
</tbody>
</table>
| ramp(x, x1, y1, x2, y2) | returns:  
  - y1 if x ≤ x1  
  - y2 if x ≥ x2  
  - \( \frac{(x-x1) \times (y2-y1)}{x2-x1} \) if \( x1 < x < x2 \)  
  - \( \frac{y1+y2}{2} \) if \( x1 = x2 \) |
<p>| round(var) | returns the nearest integer value to var |
| sin(var) | returns the sine of angle var in radians |
| slider(var1, var2, var3) | creates and polls a slider with value var1, minimum var2 and maximum var3 |
| sqrt(var) | returns the square root of var |
| subtract(var1, var2) | subtracts var2 from var1 |
| tan(var) | returns the tangent of angle var in radians |
| vAggregate(var1, var2, var3) | creates a vector consisting of the first var3 elements of var1, then all the elements of var2 and finally all the remaining elements of var1 |
| vAppend(var1, var2) | appends var2 to the vector var1 |
| vConcat(var1, var2) | concatenates var2 and vector var1 |
| vConvolve(var1, var2, var3, var4) | calculates the convolution of var1 with var2, shifted over var3. That is, for the return vector x, ( x[i] = var1[i] \times var3 + j \times var2[j] ). If var4 = 0 then the convolution is cyclic, no additional changes are made. If var4 = 1 the convolution is zero-padded, If var4 = 2 the convolution is padded with the first and last elements of var1 |
| vDom(var) | returns the domain of var |
| vDot(var1, var2) | returns the dot product of var1 and var2 |
| vGaussian(var1, var2) | creates a gaussian vector of var2 with var1 elements. That is, for a return vector x. ( x[i] = P \times \exp(-((i - var1/2) \times (i - var1/2)/2 \times var2 * var2)) ). Where P is taken such that the sum of all the elements of x equals to 1 |
| vLen(var) | returns the length of vector var |
| vMake(var1, var2) | returns a vector with var2 copies of var1 |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vMatInverse(var1)</td>
<td>returns the inverse of matrix var1</td>
</tr>
<tr>
<td>vMatMatMul(var1, var2)</td>
<td>multiplies matrix var1 with matrix var2</td>
</tr>
<tr>
<td>vMatSolve(var1, var2)</td>
<td>returns the solution of the linear set of equations with unknown vector x where var1•x = var2</td>
</tr>
<tr>
<td>vNormAbs(var)</td>
<td>returns the sum of the absolute values of the elements of vector var</td>
</tr>
<tr>
<td>vNormEuclid(var)</td>
<td>returns the square root of the sum of squares of the values of the elements of vector var</td>
</tr>
<tr>
<td>vNormFlat(var)</td>
<td>returns the sum of the values of the elements of vector var</td>
</tr>
<tr>
<td>vNormSq(var)</td>
<td>returns the sum of squares of the values of the elements of vector var</td>
</tr>
<tr>
<td>vNormalize(var)</td>
<td>returns the normalized version of vector var using the square root of squares</td>
</tr>
<tr>
<td>vRange(var)</td>
<td>returns the range of var</td>
</tr>
<tr>
<td>vSegment(var1, var2, var3)</td>
<td>returns the segment of var1, with starting point var2 and stop point var3</td>
</tr>
<tr>
<td>vSeq(var1, var2)</td>
<td>returns a vector with sequence [var1, var1 + 1, var1 + 2, ..., var2 - 1]</td>
</tr>
<tr>
<td>vSpike(var1, var2)</td>
<td>creates a vector with var2 elements filled with zeroes except the element with the var1’th index set to 1.</td>
</tr>
<tr>
<td>vTranspose(var)</td>
<td>returns the transposed matrix var</td>
</tr>
<tr>
<td>vVecRamp(var1, var2, var3)</td>
<td>var1 is a vector of horizontal coordinates, var2 is a vector of vertical coordinates. Returns the pair of subsequent horizontal coordinates enclosing var3, and do a linear interpolation on the var2-var3 interval.</td>
</tr>
<tr>
<td>uniminus(var)</td>
<td>returns -var</td>
</tr>
</tbody>
</table>