Software voor de beoordeling van primaire waterkeringen

D-Soil Model

Dutch Delta Systems

Test Plan

D-Soil Model

Test Plan

Version: 2.0
Revision: 00

April 2016

D-Soil Model, Test Plan

Published and printed by:
Deltares
Boussinesqweg 1
2629 HV Delft
P.O. 177
2600 MH Delft
The Netherlands

telephone: +31 88 335 82 73
fax: +31 88 335 85 82
e-mail: info@deltares.nl
www: https://www.deltares.nl

Contact:
Helpdesk Water
Rijkswaterstaat WVL
P.O. 2232
3500 GE Utrecht
The Netherlands

telephone: +31 88 797 7102
www: http://www.helpdeskwater.nl

Title
D-Soil Model

Client	Project	Reference	Pages
RWS	1230088-026	1230088-026-DSC-0002	10

Classification
-

Keywords
WTI2017, hydraulic structures, structural failure, calculation kernel, wave pressure, linear load calculation, quadratic load calculation

Summary
This document contains the 2nd version of the test plan for D-Soil Model, including the comments of H. Walls (Rijkswaterstaat).

Samenvatting
Dit document bevat de 2de versie van het testplan voor D-Soil Model, inclusief de commentaren van H. Walls (Rijkswaterstaat).

References

Bokma, J., 2016. *WTI 2017 D-Soil Model - Technical Design.* Tech. Rep. 1209430-003-DSC-0021, Deltares.

Deltares, 2016. *D-Soil Model Handleiding.*

Icke, J. *Eisen aan toelevering van softwarecomponenten aan het Cluster Softwareontwikkeling.* Tech. Rep. 1209430-005-DSC-002, Deltares.

Knoeff, J., 2012. *Basisuitgangspunten WT12017. Watersystemen, Faalmechanismen en Softwareontwikkeling.* Tech. Rep. 1206004-002-GEO-0002, Deltares.

Putten, H. van and P. Witlox, 2015. *Overall Testplan Software WTI2017 - RingToets, HydraRing, D-Soil Model en Faalmechanisme bibliotheken.* Tech. Rep. 1220079-005-DSC-0006, Deltares.

Putten, H. van and P. Witlox, 2016. *Acceptatietestplan – Plan voor acceptatie van de WTI2017 software.* Tech. Rep. 1230088-036-DSC-0001, Deltares.

Zwan, I. van der, 2016. *Functional Design D-Soil Model.* Tech. Rep. 1230088-026-DSC-0001, version 2, Deltares.

Version	Date	Author	Initials	Review	Initials	Approval	Initials
1.0	July 2015	dr. V. Trompille		P. Witlox ir. K.S. Lam		ir. J. Icke	
2.0	April 2016	dr. V. Trompille		dr.ir. J.G. van Putten		ir. J. Icke	

Status
final

Contents

List of Tables

1 Introduction

1.1 Purpose and scope of this document

This document contains the test plan for D-Soil Model.

The document will not give any background on the context of the WTI project. For this purpose the reader is referred to the WTI2017 and to its supporting technical reports and their background reports underneath (paragraph 1.3).

This document will not describe how the requirements of the functional design are implemented in the program but described how the requirements of the functional design are tested.

1.2 Other system documents

The full documentation on D-Soil Model comprises the following documents.

Title	**Content**	Authors
Requirements and Functional Design (Van der Zwan, 2016)	Description of the requirements and functional design.	Raymond van der Meij, Irene van der Zwan
Technical Design (Bokma, 2016)	Description of the implementation of the functional design	John Bokma
Technical Specification	Description of the arguments and usage of different software components, generated from inline comment with Doxygen	John Bokma
Test Plan	This document	Virginie Trompille
Test Report	Actuated results of the test plan.	Virginie Trompille
User Manual (Deltares, 2016)	Description of the different functionalities available in the *User Interface* and background information.	Deltares

1.3 Assumptions and constraints

CNS 1 As a general constraint, the development process needs to comply with the general process description for WTI software, contained in a separate document (Icke).

CNS 2 As a general constraint, the program needs to comply with the relevant general requirements and further design rules for the programming, documentation and testing of WTI software. This set of requirements and rules is contained in a separate document (Knoeff, 2012).

1.4 Test levels

According to Van Putten and Witlox (2015), the program D-Soil Model should be tested on four different levels (V-Model):

1 Component Testing
2 Integration Testing
3 System Testing
4 Acceptance Testing

Component Testing (Unit tests)

Component Testing is testing on code level using the unit tests. For each relevant function, a unit test is defined within the C# solution. A relevant function is a function that actually performs part of the calculation, validation or I/O of the core. Properties and purely administrative functions do not have unit tests.
Refer to chapter 2 for more information.

Integration Testing (Integration tests)

Integration Testing is testing on functional level using integration tests. These types of tests combine multiple functions to prove that high level functionality works. For this, a unit test is defined within the C# solution for each method with high level functionality.
Refer to chapter 3 for more information.

System Testing (Benchmarks and test scripts)

System Testing is testing the functioning of the complete system:

- ◇ The *User Interface* must function properly: this testing must proved that the functional and non-functional requirements are met;
- ◇ All possible errors must be handled and reported properly (including the minimum and maximum values of input).

Refer to chapter 4 for more information.

Acceptation level (Acceptance tests)

The Acceptance Test of D-Soil Model will be covered in the acceptance test for the overall WTI system (which includes acceptance tests for the stand-alone tools) and will be reported in a so-called Acceptance Report (Van Putten and Witlox, 2016).

1.5 Code coverage

Testing is considered to be ok when all the unit tests pass and when the code coverage of those tests is more than 60% so as prescribed in Van Putten and Witlox (2015) because it is a Delta Shell Light product.

To determine the code coverage of the *Calculation* components, the feature 'Code coverage' of Visual Studio can be used. This tool shows the percentage of the code that was tested in each assembly, class, and method, and is visible on the build server.

2 Component Testing (Unit tests)

The tests on code level are the unit tests. For each relevant function, a unit test is defined within the C# solution. A relevant function is a function that actually performs part of the calculation, validation or I/O of the core. Properties and purely administrational functions do not have unit tests.

These tests are considered to be ok when the unit tests pass and when the code coverage of those tests is more than 60%, as prescribed in Van Putten and Witlox (2015) for UI solutions. For these tests, see the C# solution.

The following information must be provided in the Test Report:

- Number of unit tests
- Code coverage of the unit tests
- Specify if all unit tests succeed or not

3 Integration Testing (Integration tests)

The tests on functional level are the integration tests. These types of tests combine multiple functions in the kernel to prove that high level functionality works. For this, an integration test is defined within the C# solution for each method with high level functionality.

These tests are considered to be ok when the integration tests pass. For these tests, see the C# solution.

The following information must be provided in the Test Report:

- ◇ Number of integration tests
- ◇ Code coverage of the integration tests
- ◇ Specify if all integration tests succeed or not

4 System testing (Test Scripts)

As D-Soil Model is a database and support software for schematize the subsoil, the system testing will only consist of executing test scripts and common tests for general User Interface functionalities.

A Test Script will be provided to the tester, describing a sequence of actions and the expected outcome. Another goal of the test script is testing against the functional and non-functional requirments in the Functional Design. An overview of the Test Scripts is listed below:

- **Language:**
  The test must be done only for Dutch language.

- **Layout:**
  The layout must be identical to the layout described in the Technical Design (**?**):
  - Menu items
  - Options in each menu items
  - Tool bar
  - Components
  - Tabs
  - Icon bar of tabs

- **Menu:**
  All available options under the different items of the menu must be tested.
  This includes the importation of data's via the *Import* option under the *File* menu: all the possibilities supported by D-Soil Model must be tested by checking that the data's are correctly and completely imported. Each importation of a file will have at least one test script. The files to be tested will be provided to the tester.

- **Menu bar:**
  All icons of the menu bar must be tested.

- **Tables toolbar:**
  All icons of the Tables toolbar (when available) must be tested:
  - Add
  - Delete
  - Cut
  - Copy
  - Paste
  - Fits columns
  - Edit
  - Edit Unit
  - Export (all available formats must be tested)

- **Map / Length Profile toolbar:**
  All icons of the Map and Length Profile toolbar (when available) must be tested:
  - Selection
  - Pan
  - Zoom to extents
  - Zoom to extents and reset aspect ratio
  - Zoom to data
  - Zoom by rectangle
  - Options
  - Show legend

- Open layer from file
- Export layers
- Save map
- Select web layer
- Selection
- Add location for split

◇ **Property Editors:**
The content of the property editor must be check for all selected objects.
The coupling between the content of the Property Editors and the Tables component must also be tested: changing a property in the Property Editors should affect the Tables component.

◇ **Default/Min/Max:**
The default, minimum and maximum values of all properties (as given in the Functional Design) must be tested.

◇ **Filters:**
All possible filters must be tested, by checking that the resulting list of is as expected (no more, no less). All possible combination of filters must also be tested.

All the *Test Scripts* are part of a so-called *Test Document* that will be joined as Annex in the Test Report.

4.1 Test Script template

A test script will consist of:

◇ The header of a test has a title.
◇ The description of a test.
◇ If applicable, the preconditions of a test. A precondition descripts the system state and data required to start the test;
◇ The sequence of steps (acts) that must be performed. For each step the expected output and the actual (test) result.
◇ Remarks and details on the test results can be put at the end of the test.

If a test step fails, just add “failed” in the column “Result”.
The test script also needs to describe which functional and non functional requirements are tested.

Test Title		
Test description: <description of the tests>		
Preconditions: <system state and data>		
Requirement(s): <describe which functional or/and non functional requirement(s) is (are) tested by this test script>		
Steps (test script)	***Output***	***Result***
Act 1	Expected Output	Passed
Act 2	Expected Output	Failed, Comment 1
	Expected Output	Passed
Act 3	-	-
Act 4	Expected Output	Failed, see Remark
Remarks (details, data sets, screenshots)		

Rijkswaterstaat
Ministry of Infrastructure and the Environment

Deltares
Enabling Delta Life

PO Box 177
2600 MH Delft
Boussinesqweg 1
2629 HV Delft
The Netherlands