Software for the assessment of primary flood defences

BM - Gras Buitentalud

WTI2017

Test Plan

BM - Gras Buitentalud

Test Plan

Version: 1.2
Revision: 00

Nov. 2016

BM - Gras Buitentalud, Test Plan

Published and printed by:

Deltares	telephone:	+31 88 335 82 73
Boussinesqweg 1	fax:	+31 88 335 85 82
2629 HV Delft	e-mail:	info@deltares.nl
P.O. 177	www:	https://www.deltares.nl
2600 MH Delft		
The Netherlands		

Contact:

Helpdesk Water	telephone:	+31 88 797 7102
Rijkswaterstaat WVL	www:	http://www.helpdeskwater.nl
P.O. 2232		
3500 GE Utrecht		
The Netherlands		

Title
BM - Gras Buitentalud

Client	**Project**	**Reference**	**Pages**
RWS	1230088-041	1230088-041-DSC-0001	16

Classification
-

Keywords
Dike, grass revetment, erosion, wave, wave run-up, wave impact, run-up, impact, revetment transition, WTI 2017, safety assessment, software

Summary
This document contains the test plan for BM - Gras Buitentalud, an application that computes the erosion of a grass revetment in the wave run-up zone and the wave impact zone.

Samenvatting
Dit document bevat het test plan voor BM - Gras Buitentalud, een User Interface applicatie die een gebruiker in staat stelt om berekeningen uit te voeren aangaande de erosie van grasbekleding als gevolg van golf-klap- en golfoploop effecten.

References
Refer to chapter 5.

Version	Date	Author	Initials	Review	Initials	Approval	Initials
1.0	March 2016	dr. V. Trompille		dr.ir. J.G. van Putten		ir. J. Icke	
1.1	July 2016	ir. A.A. Markus		dr.ir. J.G. van Putten		ir. J. Icke	
1.2	Nov. 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 the standalone program BM - Gras Buitentalud that computes the erosion of a grass revetment in the wave runup zone and in the wave impact zone. This program is part of the WTI 2017 program.

The document will not give any background on the context of the WTI project and on the derivation or motivation of the supported physical models. For this purpose the reader is referred to the VTV2017 and to its supporting technical reports and their background reports underneath.

This test plan describes the tests that must take place before the program can be released. The tests must ensure that all the requirements are fulfilled with and that the quality of the program is sufficient (cf. **??**). The numeric results and the testing of the User Interface are reported in the Test Report.

1.2 Other system documents

The full documentation on the program comprises the following documents.

Title	**Content**
Functional Design (**?**)	Description of the requirements and functional design.
Technical Design (**?**)	Description of the implementation of the technical design of BM - Gras Buitentalud.
Technical documentation (**?**)	Description of the arguments and usage of different software components, generated from in-line comment with Doxygen.
Test Plan (this document)	Description of the different regression and acceptation tests, including target values.
Test Report (**?**)	Description of the test results (benchmarks and test scripts).
User Manual (included with the software)	Description of the different functionalites available in the *User Interface* and background information.

1.3 Requirements

1.3.1 Functional requirements

In the Functional Design of the *User Interface* (**?**), the following requirements are defined and must be therefore tested. They will be tested at System Testing level (see chapter 4).

REQ 1 The *User Interface* must support the input of all required data.

REQ 2 The *User Interface* must support the validation of all required input data.

REQ 3 The *User Interface* must be able to start the Grass Run up Assessment calculation.

REQ 4 The *User Interface* must be able to retrieve the results of the Grass Run up Assessment calculation.

REQ 5	The *User Interface* must support the display of the results of the Grass Run up Assessment calculation.
REQ 6	The *User Interface* must be able to start the Grass Impact Assessment calculation.
REQ 7	The *User Interface* must be able to retrieve the results of the Grass Impact Assessment calculation.
REQ 8	The *User Interface* must support the display of the results of the Grass Impact Assessment calculation.
REQ 9	The *User Interface* must be able to read and write all data to its own file for storage and reuse of the data.
REQ 10	The *User Interface* must offer functionality to switch between all of its parts.
REQ 11	The *User Interface* must be available in Dutch.
REQ 12	The *User Interface* must offer simple visualization of the input for the Grass Impact Assessment Calculation.
REQ 13	The *User Interface* must offer simple visualization of the input for the Grass Impact Run up Calculation.

1.3.2 Non-functional requirements

Besides the functional requirements, BM - Gras Buitentalud must meet some non-functional requirements. These non-functional requirements are the same as required for WTI software. All non-functional requirements (NFR, U, TAB, TOI, PSR, R, SLA and Top) are given in the spreadsheet that lists all WTI requirements (**?**). The non-functional requirements with relevance to the development of this application and its proposed use are listed in the Functional Design (**?**).

Not all non-functional requirements can be addressed via automatic tests or even by manual testing. Only the non-functional requirements given in Table 1.2 can be tested via unit/integration tests or test-scripts and will be therefore reported in the Test Report.

Table 1.2: *"Non-Functional Requirements" relevant at Test level for BM - Gras Buitentalud*

Nr.	**Requirement**	**Testing level**
NFR3	Data definitions will follow existing and emerging standards such as IRIS as much as possible.	System Testing
NFR6	Sufficient error checks on the validity and completeness of data during import or input, as well as during a computation.	System Testing
NFR8	Output of detailed deterministic results must enable all users to trace back the correctness of the implemented mechanism models.	System Testing
NFR12	The user-interface may not cause crashes during regular usage.	System Testing
NFR14	Consistency between the input data and the output data must be guaranteed.	System Testing
NFR15	The general required code coverage is 80%, except for the Delta Shell Light components, therefore the code coverage of 60% is required.	Component and Integration Testing
U123	De WTI Software moet tot een eenduidig en reproduceerbaar resultaat leiden.	System Testing
U124	De WTI Software moet robuust zijn voor kleine variaties in de invoer. Onder 'robuust' wordt verstaan: nooit een software crash. Dus: ofwel een melding dat invoer onjuist is, ofwel een melding dat een som niet convergeert, ofwel een antwoord retourneert.	System testing
U129	Alle componenten binnen een bibliotheek moeten foutcodes retourneren met een gestandaardiseerde (nog nader vast te stellen) betekenis of gebruiken excepties om fouten door te geven.	System testing
U131	Om de bibliotheken te kunnen testen levert elke bibliotheek testsoftware mee in de vorm van unit tests.	Component and Integration Testing
U132	De IO laag sluit altijd aan op een intern bestand voor opslag van de invoer- en (tussen)resultaten.	System Testing
U133	De WTI tools worden standaard alleen uitgeleverd met een Nederlandstalige UI en met Nederlandstalige meldingen en rapportages.	System Testing
U145	De naamgeving van objecten, parameters, functies moet over alle applicaties heen consistent zijn. Voor dat doel moet gebruik worden gemaakt van de actuele versies van de terminologielijst en de parameterlijst.	System Testing
TAB2	De applicaties moeten door gebruikers gebruikt kunnen worden met standaardrechten. Het moet dus niet nodig zijn om de applicatie uit te voeren met administrator rechten, met uitzondering van installatie.	System Testing
Top 10	De gebruiker kan bepalen waar de gegevens worden neergezet.	System Testing

1.4 Test levels

According to **?**, the program BM - Gras Buitentalud 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[1]. 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 main calculator must provide the correct answers to confirm its performance according to the functional design;
◇ All possible errors must be handled and reported properly;
◇ The *User Interface* must function properly.

Refer to chapter 4 for more information.

Acceptation level (Acceptance tests)

The Acceptance Test of BM - Gras Buitentalud 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 (**?**).

[1]A C# solution is essentially a collection of source files and their relations. This collection is used to build the actual program. Besides the source code for the program itself it also contains the source code for the unit and system tests.

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 **?**.

Note that within the context of this test plan the components which are not part of the program development, i.e. external component but also the DSL components, are not included in the test coverage.[2]

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]For the DSL components separate test plans and test reports are available.

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 **?** for UI solutions. For these tests, see the C# solution.

The following information must be provided in the Test Report:[1]

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

[1] Unit tests are related directly to the implementation in source code and therefore have no direct relation to the functional requirements, referred to above.

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. Both the source code and the external data for these tests and the unit tests are an integral part of the delivery.

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 (Benchmarks and Test Scripts)

The tests on this level are to provide proof of the fact that the program meets its acceptation criteria:

- ◇ all main functions must provide the correct answers;
- ◇ all possible errors must be handled and reported properly;
- ◇ the functionalities of the *User Interface* listed in the functional requirements (section 1.3.1) must be present and work properly;
- ◇ the non-functional requirements listed in section 1.3.2 must be present and work properly.

4.1 Test of the output

Based on the formula in the Functional Design of the Grass Run-up Zone and the Grass Wave Impact Zone kernels, benchmarks were created to test both kernels (**??**). The same benchmarks can be used to test BM - Gras Buitentalud.

For the Grass Run-up calculation, there are 13 benchmarks:

1. One stationary event, ‘not scaled’ computation of cumulative overload
2. One stationary event, ‘scaled’ computation of cumulative overload
3. One stationary event, ‘scaled’ computation of cumulative overload, z_{eval} = 2 m
4. One stationary event, ‘scaled’ computation of cumulative overload, z_{eval} = 3 m
5. One stationary event, ‘scaled’ computation of cumulative overload, α_{M} = 1.5
6. One stationary event, ‘scaled’ computation of cumulative overload, α_{S} = 0.8
7. One stationary event, ‘scaled’ computation of cumulative overload, α_{S} = 0.8 and α_{M} = 2
8. One stationary event, ‘scaled’ computation of cumulative overload, small waves
9. One stationary event, ‘scaled’ computation of cumulative overload, very small waves
10. One stationary event, ‘scaled’ computation of cumulative overload, very large waves
11. Synthetic storm data (time series), ‘not scaled’ computation of cumulative overload
12. Synthetic storm data (time series), ‘scaled’ computation of cumulative overload
13. Direct input storm data, ‘scaled’ computation of cumulative overload

For the Grass Wave Impact calculation, there are 9 benchmarks:

1. ‘Direct input’ storm event (Vechtdelta deelgebied A-1) - closed sod
2. ‘Direct input’ storm event (Vechtdelta deelgebied A-1) - open sod
3. ‘Direct input’ storm event (Vechtdelta deelgebied A-1) - fragmented sod
4. ‘Direct input’ storm event (Benedenrivierengebied) - closed sod
5. ‘Direct input’ storm event (Benedenrivierengebied) - open sod
6. ‘Direct input’ storm event (Kust) - open sod
7. ‘Synthetic’ storm event (Time series Q-variant)
8. ‘Synthetic’ storm event - Time series water level exceeds Q-Variant water level +0,03 m
9. ‘Direct input’ storm event (Vechtdelta deelgebied A -1) - closed sod - $t_{\text{s,top}} < t_{\text{s,top,min}}$

The output of those benchmarks will be checked via unit tests and the result will be visible on the build server.

4.2 Test of the error(s)/warning(s)

The error(s) generated by an incomplete input or input outside its limits can be tested at code level via the unit tests, see chapter 2, using the valid intervals of the input parameters given in the Functional Design (**?**). Results of those unit tests will be visible on the build server.

4.3 Test of the *User Interface*

The test of the *User Interface* will be executed by a tester using *Test Script(s)*. All the requirements listed in the Functional Design of the program (section 1.3.1) and all the relevant non-functional requirements (section 1.3.2) must be tested.

A *Test Script* describes a sequence of actions and the expected outcome. All the *Test Scripts* are part of a so-called *Test Document* that will be joined as Annex in the Test Report.

5 Literature

Rijkswaterstaat
Ministry of Infrastructure and the Environment

Deltares
Enabling Delta Life

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