Deltares

Enabling Delta Life

DAM Engine

Technical Design

Deltares

DAM Engine

Technical Design

1210702-000

Deltares

Title
DAM Engine

Client	Project	Reference	Pages
Deltares - Geo engineering DKS	1210702-000	1210702-000-GEO-0004	52

Classification
-

Keywords
Dike, safety assessment, design, software, macro stability, piping

Summary
This document contains the technical design for DAM Engine, a software module that computes the strength of a complete dikering with respect to several failure mechanisms, such as macro stability and piping.

Samenvatting
Dit document bevat het technisch ontwerp voor DAM Engine, een software module die een gebruiker in staat stelt om voor een dijktraject berekeningen uit te voeren voor verschillende faalmechanismen, waaronder macrostabiliteit en piping.

References
Refer to chapter 11.

Version	Date	Author	Initials	Review	Initials	Approval	Initials
0.5	Nov 2018	Tom The		John Bokma André Grijze		Maya Sule	

Status
draft
This is a draft report, intended for discussion purposes only. No part of this report may be relied upon by either principals or third parties.

Deltares

Contents

Deltares

Deltares

List of Figures

List of Tables

1 Introduction

1.1 Purpose and scope of this document

This document contains the technical design for the DAM Engine, a computational engine for the automated calculation of the strength of dikes. DAM was developed by Deltares with and for STOWA for all water authorities. This document describes the full intended architecture of the DAM Engine. What will actually will be implemented depends on the requirements of the clients using this DAM Engine. If some functionality is not (yet) needed, then that part does not need to be implemented.

1.1.1 Future options

As mentioned above this document contains some options that will not be implemented in the first release, but are foreseen to be implemented in the near future. Therefore although sometimes a reference will be made to these options, these will not be described in detail yet.

That applies in particular to the following subjects:

- NWO module("Niet Waterkerende Objecten")
- WBI failure mechanisms (Piping, Macrostability)

1.2 Other system documents

The full documentation on the program comprises the following documents.

Title	Content
DAM Engine- Architecture Overall (The, 2017a)	Description of overall architecture of the DAM Engine and its components.
DAM Engine- Functional Design (Zwan, 2017a)	Description of the requirements and functional design.
DAM Engine- Technical Design (this document) (The, 2017b)	Description of the implementation of the technical design of DAM Engine.
DAM Engine- Technical documentation (Doxygen, 2017)	Description of the arguments and usage of different software components, generated from in-line comment with Doxygen.
DAM Engine- Test Plan (Trompille, 2017a)	Description of the different regression and acceptation tests, including target values.
DAM Engine- Test Report (Trompille, 2017b)	Description of the test results (benchmarks and test scripts).
Architecture Guidelines (Kleijn *et al.*, 2017)	Architecture guidelines that are used by DSC-Deltares.

Table 1.1: *DAM Engine system documents.*

1.3 Document revisions

1.4 Document revisions

1.4.1 Revision 0.1

First concept of the document.

1.4.2 Revision 0.2

Adapted based on reviews of this document by Jan Noort and André Grijze.

1.4.3 Revision 0.3

Adapted based on review of this document by John Bokma.

1.4.4 Revision 0.4

- Updated the interface description of the failure mechanism wrapper.
- Added a chapter on adding a new Failure Mechanism.

1.4.5 Revision 0.5

- Added a chapter on Failure Mechanism implementations.
- Added description of the implementation of the WBI Macrostability kernel.

2 System Architecture

This chapter contains diagrams describing the modules and submodules of the DAM Engine and how they interact. In chapter 6 a short description of each module/submodules is given.

2.1 DAM components

DAM Engine is part of the whole DAM system that contains several components. Please see Figure 2.1 for an overview of the components of DAM. In (The, 2017a) a description of the overall architecture of the DAM system can be found.

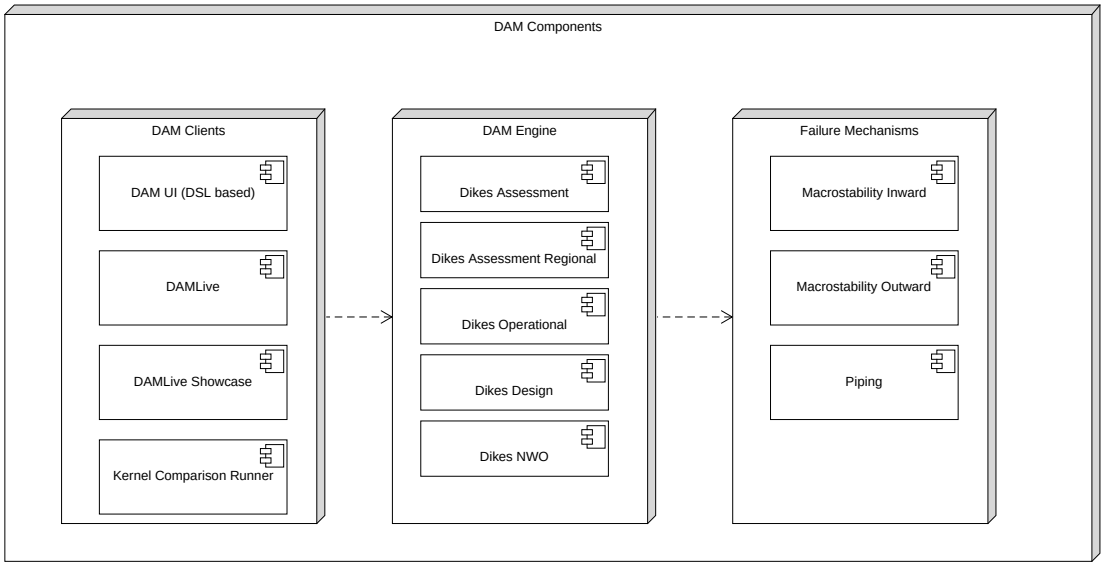

Figure 2.1: *DAM Engine and its components.*

The arrows illustrate the dependencies of the components.

2.2 DAM Engine data flow

Please see Figure 2.2 for an overview of the data flow within the DAM system.

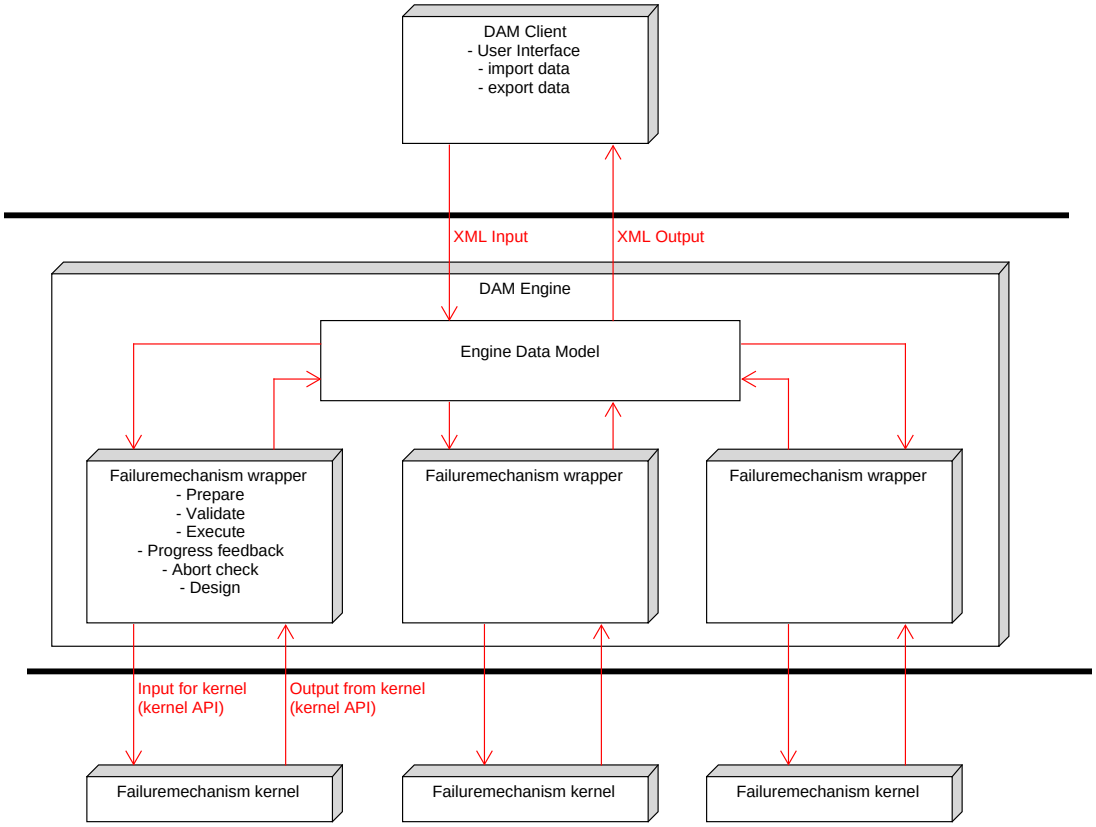

Figure 2.2: *DAM Engine and its components.*

The red arrows illustrate the dataflow between the main DAM components.
As can be seen the data exchange between the DAM Engine and the failuremechanism kernel (bottom of the picture) is done through the API that is defined by the failuremechanism kernel. The data exchange between the DAM Engine and the DAM client (top of the picture) is done through XML files (one for input and one for output), which are well defined by XML schemas (XSD's).

2.3 DAM Engine components

The DAM Engine itself also consists of several modules. These can be seen in see Figure 2.3

All of the submodules inside the Main Modules are completely independent. All of the submodules inside the Supporting Modules are also independent. But all these submodules can be used by each of the main modules. The arrows show the allowed dependencies.

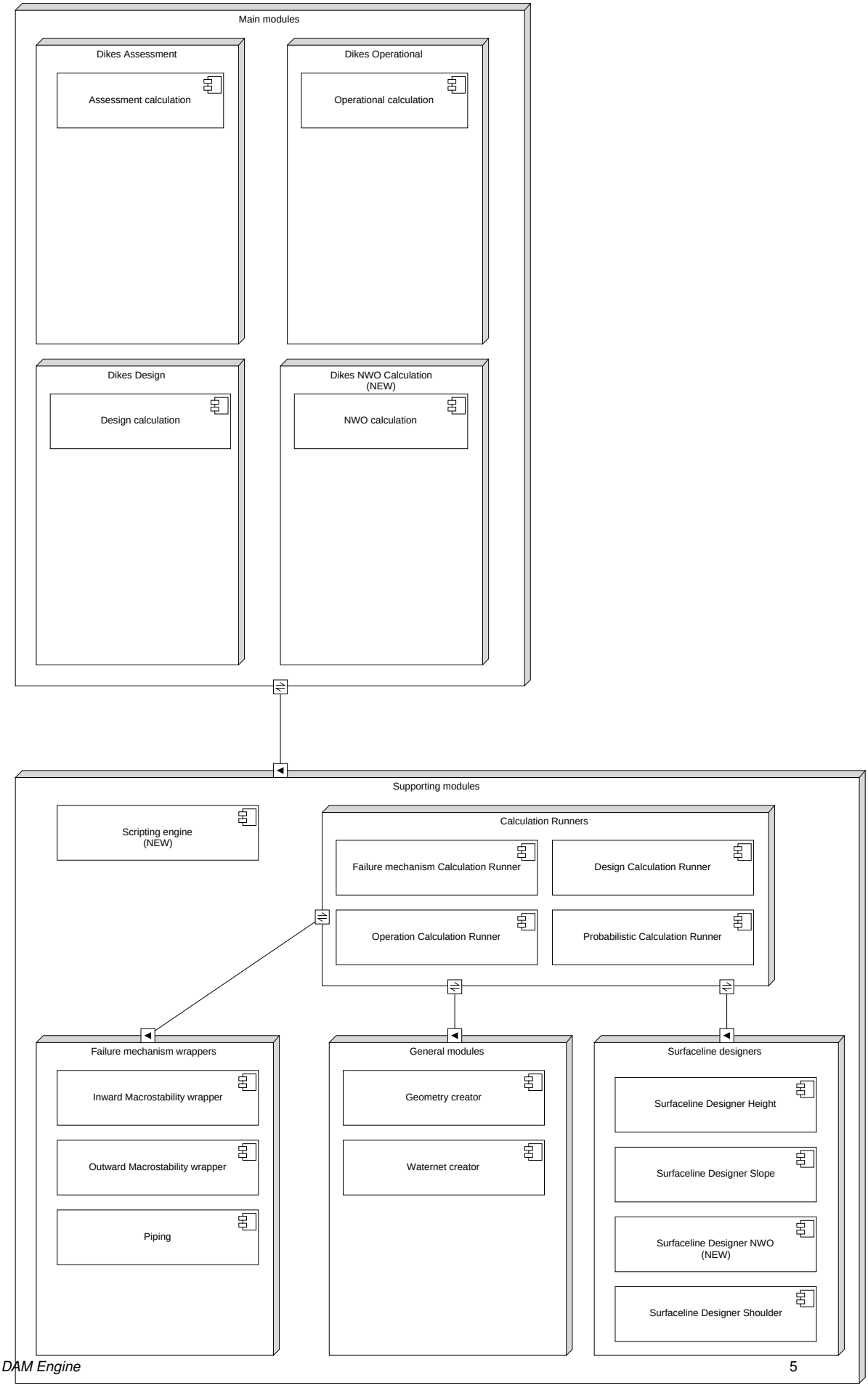

Figure 2.3: *DAM Engine and its components*

2.4 DAM Engine sequence and activity diagrams

In this section the sequence diagrams, showing the use of the submodules are shown. For each sequence diagram a corresponding activity diagram is also shown

2.4.1 Dikes assessment

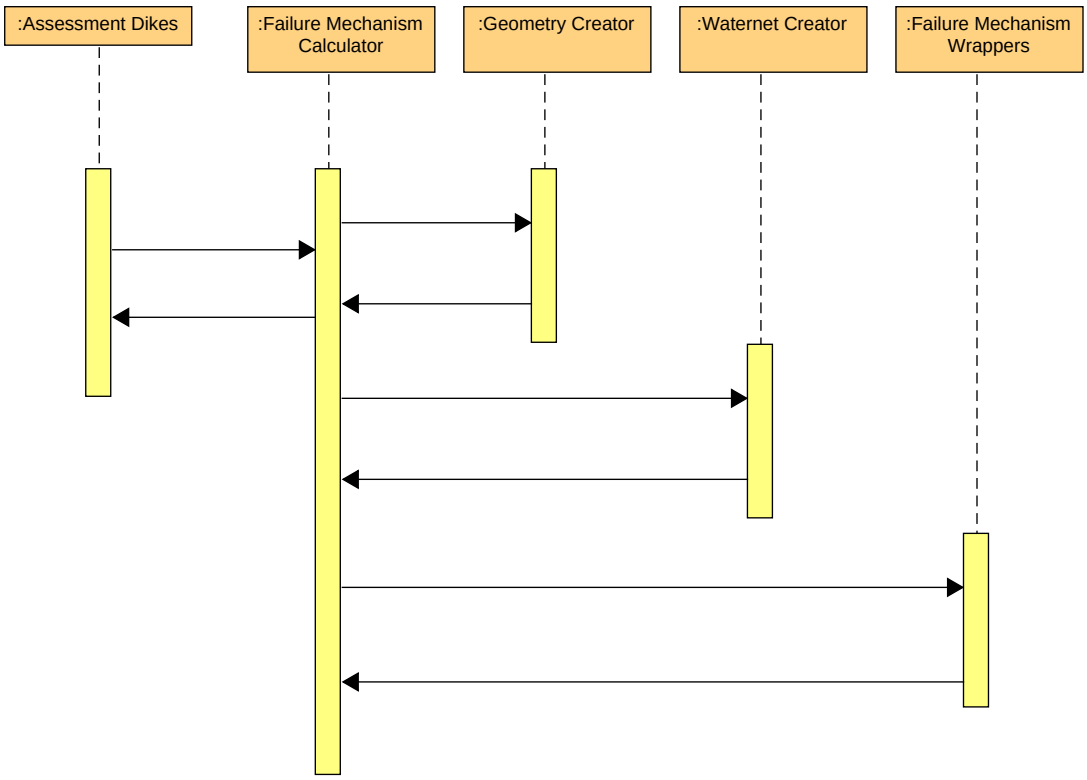

Figure 2.4: *DAM Engine Assessment sequence diagram.*

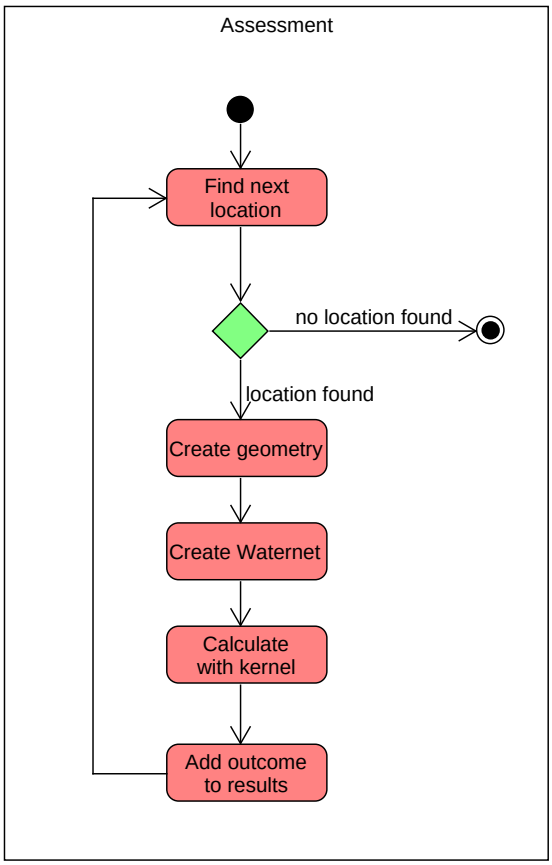

Figure 2.5: *DAM Engine Assessment activity diagram.*

2.4.2 Dikes operational

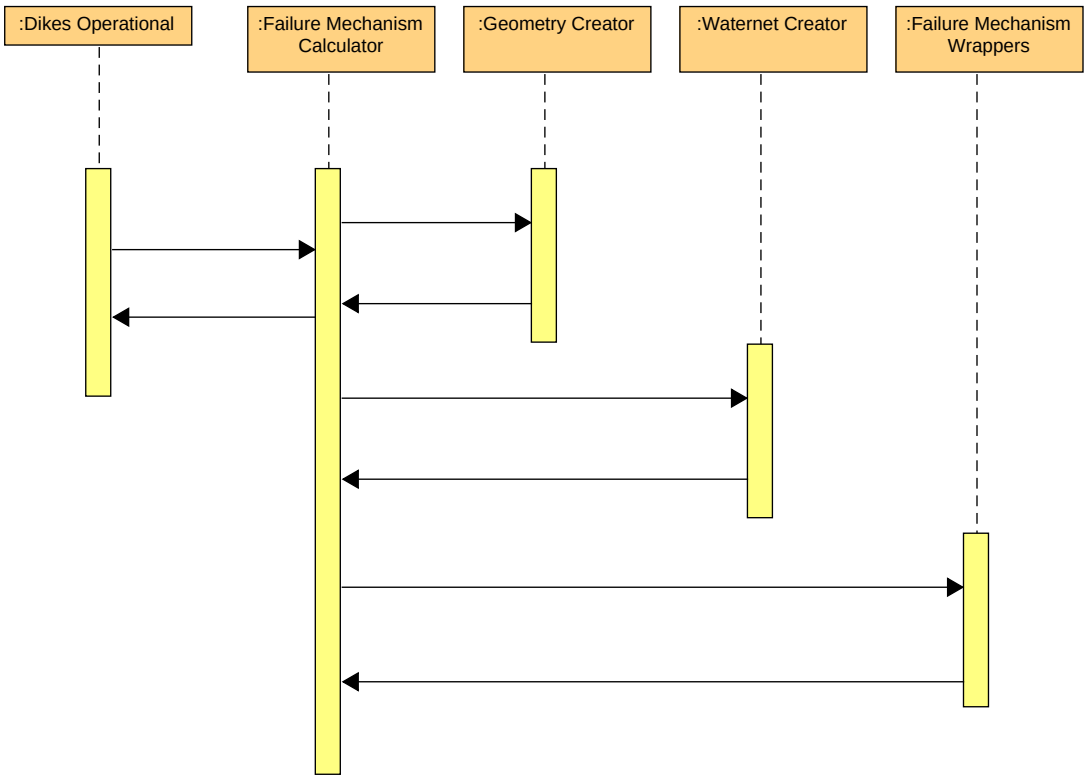

Figure 2.6: *DAM Engine Operational sequence diagram.*

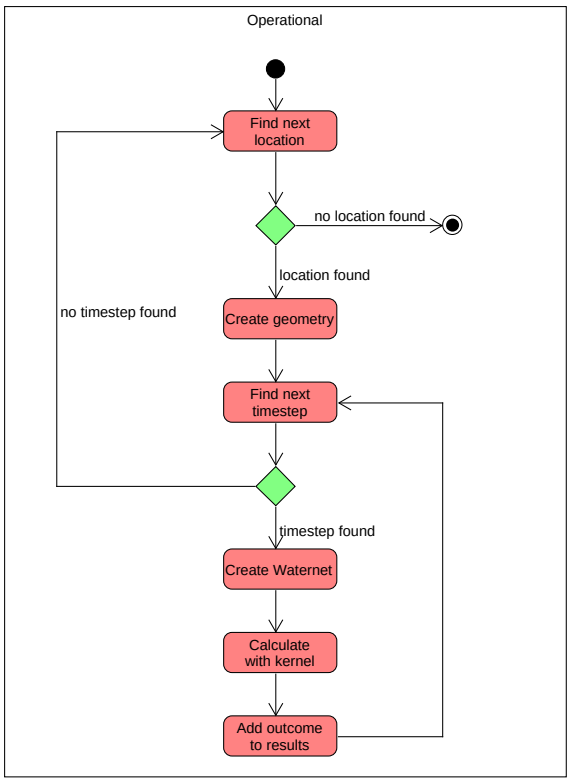

Figure 2.7: *DAM Engine Operational activity diagram.*

2.4.3 Dikes design

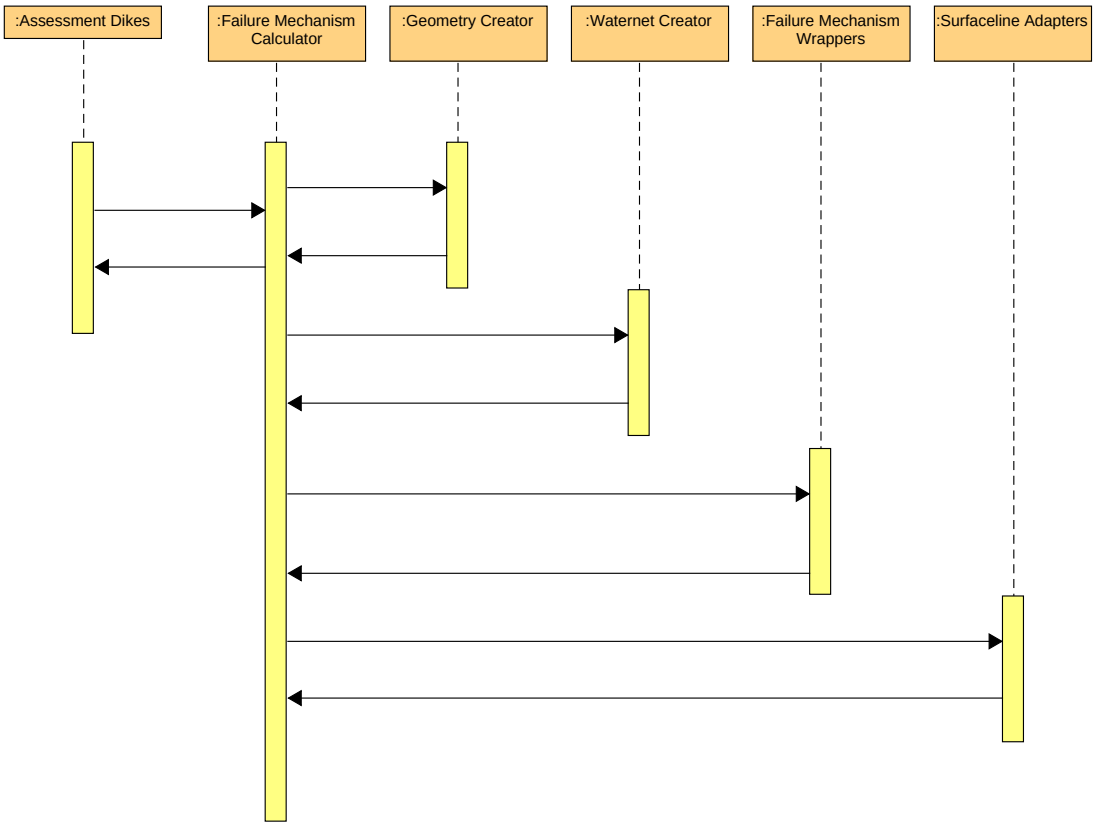

Figure 2.8: *DAM Engine Design sequence diagram.*

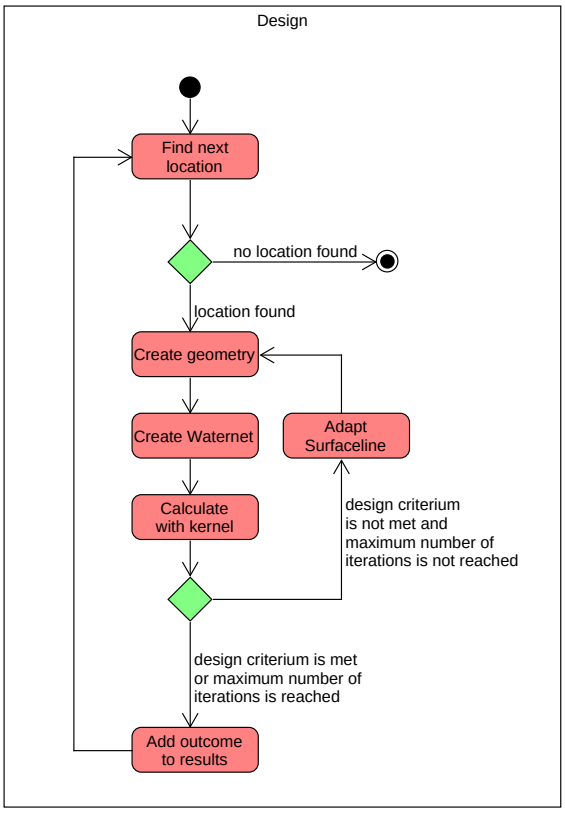

Figure 2.9: *DAM Engine Design activity diagram.*

2.4.4 Dikes NWO calculation

This is not yet to be implemetend in DAM Engine and therefore this paragraph has not yet been written.

3 Architectural Choices

3.1 Architecture guidelines

Within Deltares, DSC, a document is being written about Archtitecture Guidelines (Kleijn *et al.*, 2017). Although it is still a work in progress DAM Engine should adhere to those guidelines. More specific guidelines are added in the following sections of this chapter.

3.2 Design principles

These are the main design principals to maintain during the DAM Engine development.

- No circular references between objects. When it is really unavoidable, then do it through a generic interface (e.g. IParentObject)
- The calculation will support parallellization. So do not use global variables and avoid using statics.
- Failure mechanisms will be connected through wrapper classes, which will share a common IFailureMechanism interface
- Surfaceline designer classes will share a common ISurfacelineDesigner interface
- The DAM Engine must provide progress information of the calculation, so clients of the DAM Engine can show a progressbar
- The DAM Engine must provide the possiblity to abort a calculation within a reasonable timespan.
- There should be no User Interface elements shown anytime during the calculation.

3.3 Programming environment

The DAM Engine will be developed in C# with the .NET 4.5 framework. The development environment will be Visual Studio 2015.

3.4 Error handling

Errors within the DAM Engine are handled through the standard exception handling of the .NET framework. Error messages must contain as much information as possible, so a user can trace back the error to the input data.
Errorhandling with a failuremechanism kernel is done through the mechanism that is supplied by the API of the specific kernel.
Errorhandling with DAM Client is done by passing the error messages as part of the output XML file.
In fact it is the same mechanism that is used for exchanging the regular data (input and output), as shown in Figure 2.2.

The DAM Engine should be able to issue the error messages in different languages. In the first implementation only the 2 following languages will be supported:

- Dutch (NL)
- English (US)

For translations, the standard Windows mechanism with language resource dll's will be used. Note: the current implementation of DAM uses another mechanism for translations, that will not be applied here, because it is dependent on the DSL (Delta Shell Light) library, which will not be used for the DAM Engine.

3.5 Libraries and components

DAM Engine uses other libraries and components.

For now we foresee only the use of the following libraries:

- Failure mechanisms.
- Math.NET.

Other libraries may be used under the condition that they are open source and free components, that are free to redistribute.
All libraries should be listed in a manifest accompanying the release of DAM Engine. The list should also specify under which license each specific library is distributed.

Note: the current implementation of DAM uses the DSL (Delta Shell Light) library. This library will explicitly not be used for the DAM Engine, because this library is being made obsolete.

3.5.1 Failure mechanisms

The failure mechanisms are treated as external libraries. Some failure mechanisms were part of the source of the original DAM program. With the new architecture of DAM Engine this will no longer be the case. These failure mechanisms will be placed in a DAM failure mechanisms library, that is not part of DAM Engine anymore. The following failure mechanisms are currently supported by the original DAM program:

- Piping Bligh (not opensource).
- Piping Sellmeijer VNK (not opensource).
- Piping Sellmeijer 4 forces (not opensource).
- D-Geo Stability inward (not opensource, but commercial).
- D-Geo Stability outward (not opensource, but commercial).
- D-Geo Horizontal Balance (not opensource, but commercial).

After the original failure mechanisms have been implemented, the new WBI failure mechanisms will be added:

- WTI Piping
- WTI Macrostability inward

3.5.2 Math.Net

Math.Net is a library that is distributed under the MIT/X11 license. Therefore it meets the conditions about open source and free redistribution.

4 Data Model

This chapter contains diagrams describing the main data objects of the DAM Engine and their relation to each other. In chapter 5 a short description of these data objects is given.

4.1 Main Data Model

The main data model can be seen in see Figure 4.1 It is not fully worked out, but just a global overview. The details will be filled in when programming the DAM Engine. This is because we do not intend to write a big upfront design.

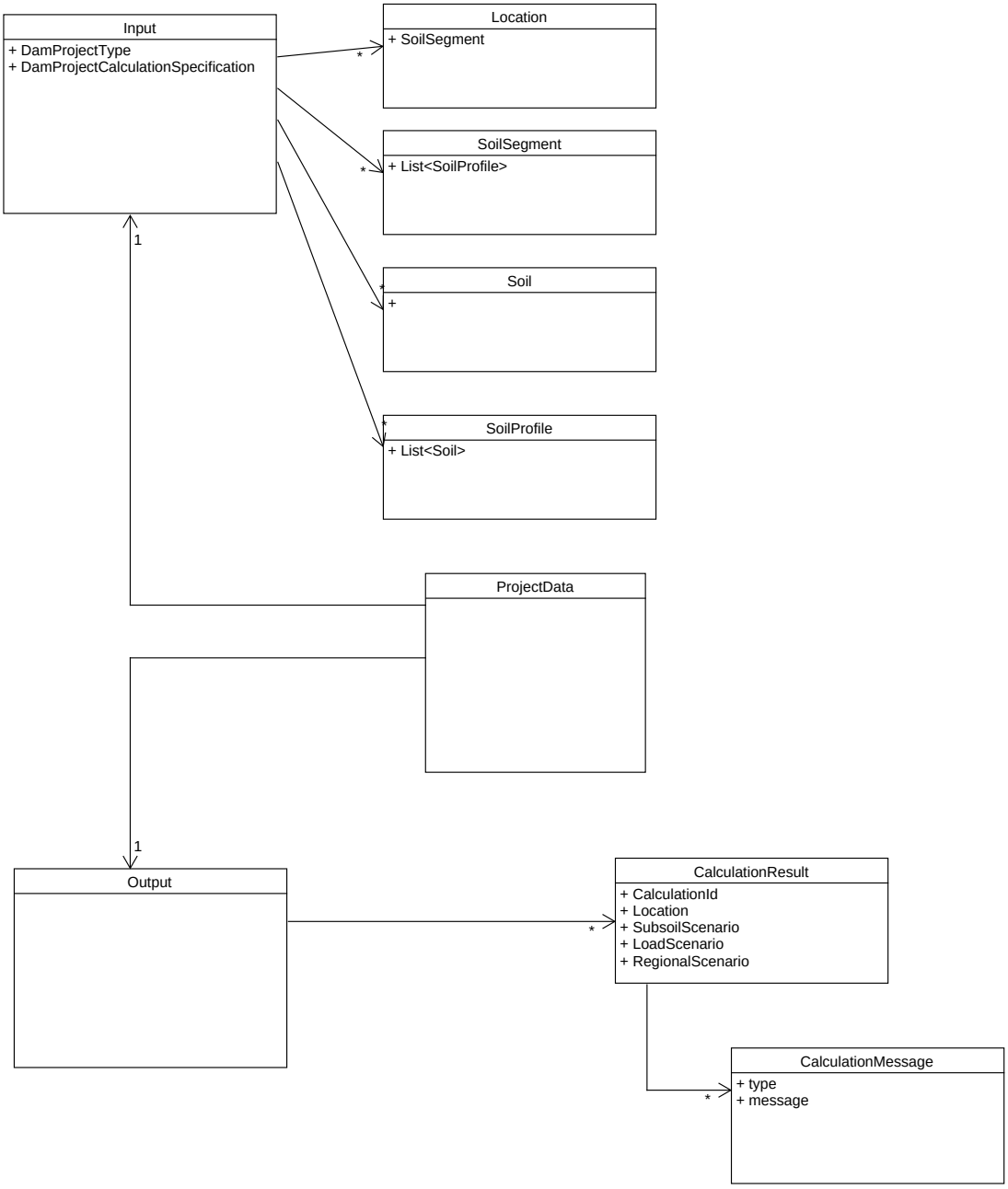

Figure 4.1: *DAM Engine main data model.*

4.2 Location

The data model of the Location class can be seen in see Figure 4.2

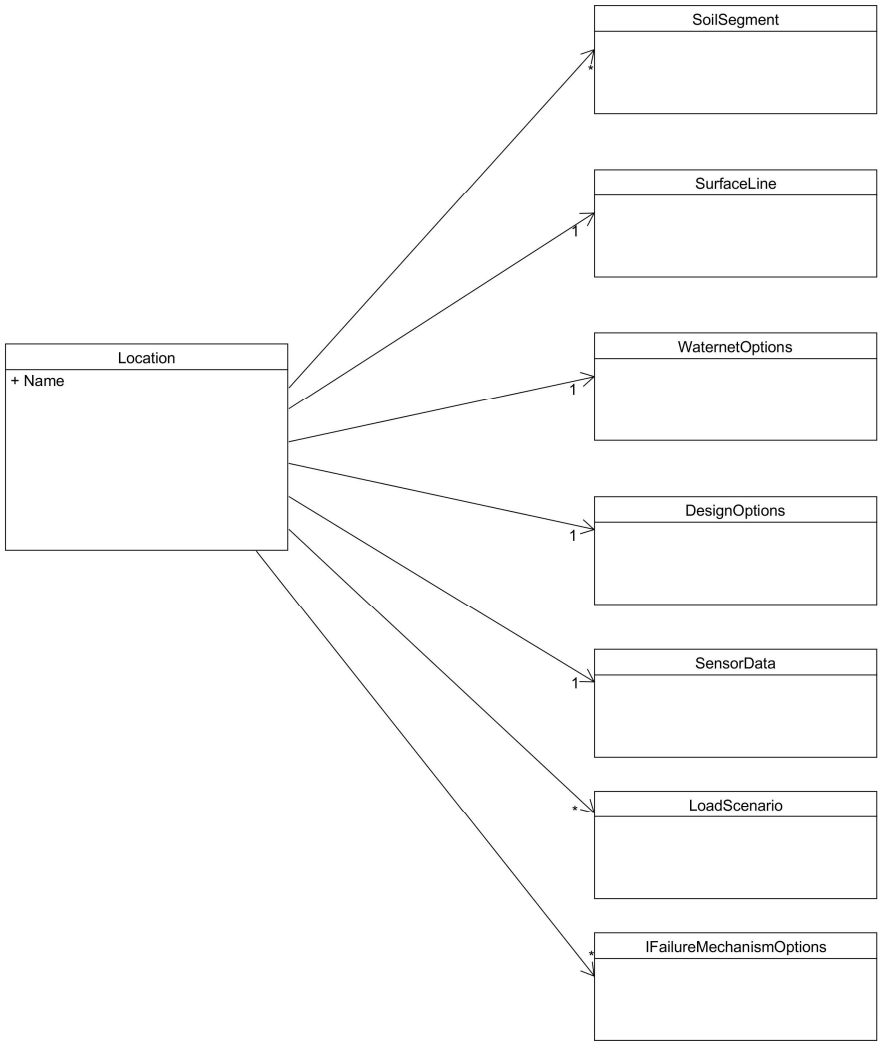

Figure 4.2: *DAM Engine Location object.*

5 Data Description

5.1 Type enumerations

5.1.1 MainMechanismType

The following main failure mechanisms are implemented.

- Macrostability inward.
- Macrostability outward.
- Piping.
- Horizontal Balance.

5.2 Scenarios

The verb Scenarios is widely (ab)used within DAM. It is good to define in which context scenarios are used and how they are to be called. Simply using the word scenario is not enough. Within DAM we have 3 types of scenarios:

- Subsoil scenario.
- Design scenario.
- Regional scenario.

5.2.1 Subsoil scenario

Used as part of the stochastic subsoil schematization. A subsoil scenario defines a possible 1D- or 2D-profile that applies to a certain location.

5.2.2 Design scenario

Used for Design calculation. In a design calculation a new surfaceline is designed for a location, based on a target failure factor (e.g. due to new requirements), or load (e.g. a higher waterlevel).

5.2.3 Regional scenario

For regional assessments there are certain scenarios that have to be evaluated, depending on the circumstances (e.g. drought, type of dike etc.). Part of the assessment is the determination which scenarios have to be evaluated and then calculating those scenarios.

5.3 Main Data Model

5.3.1 Input

DamProjectType

The following Dam project types are supported

- Assessment
- Operational
- Design
- NWO (not yet implemented)

DamProjectCalculationSpecification

This class specifies which failuremechanism is to be calculated and it also contains the specific options for the selelected mechanism (e.g. which calculation model)

Locations

This is a collection of locations, with each location containing the location specific data.

Soil Segments
This is a collection of soil segments, with each segment containing the subsoil data for a specific failure mechanism.

Soils
This is a collection of soils, with each soil containing the soil parameters needed for the calculation of all failure mechanisms.

5.3.2 Output

CalculationResults
A calculation result holds the result for a specific location, a specific failure mechanism, and a specific subsoil scenario of a specific segment defined in the location data.

CalculationMessages
These are all the message that are generated by the calculation. A message must contain as much information as possible to trace back the information tho the input data (e.g. a specific location, a specific failure mechanism, and a specific subsoil scenario of a specific segment defined in the location data).

5.4 Location

SoilSegment
A soil segment contains the subsoil data for a specific failure mechanism.

SurfaceLine
A surfaceline describes the dike profile in a specific location. In the Design calculation it can also be the new dike profile, which can meet design criteria in a specific design scenario.

WaternetOptions
The options that support the creation of a waternet in a specific location.

DesignOptions
The options that will be used in the Design calculation (e.g. how to design a shoulder when needed).

SensorData
The sensor data can be used to define a waternet based on live sensor data. This sensor data holds information about ID and location of the sensor. The actual sensor readings are defined as timeseries readings for each sensor in each location.

DesignScenario
Used for Design calculation. A design scenario contains the following items:

- Riverlevel low
- Riverlevel high
- Dike table height
- Required safety factor for each specified failure mechanism
- Uplift criterium for each specified failure mechanism
- Waternet options for each specified failure mechanism

IFailureMechanismOptions
Specific options for each location for each failure mechanism.

6 Module Description

6.1 DAM Engine main modules

6.1.1 Assessment Dikes

This module performs an assessment for general dikes (e.g. primary dikes).

Primary assessment calculation
This is the main submodule of the primary assessement. This submodule contains the main loop of the calculation.

6.1.2 Assessment Regional Dikes

This module performs an assessment for regional dikes.

Regional assessment calculation
This is the main submodule of the regional assessement. This submodule contains the main loop of the calculation.

Regional scenario selector
This submodule generates all the scenarios that have to be evaluated for a specific location. The scenarios are selected based on the local conditions.

Regional schematization factor calculator
This submodule calculates the schematization factor in a location based on all results of all scenarios in a location.

6.1.3 Design Dikes

This module performs an design calculation for all types of dikes.

Primary design calculation
This is the main submodule of the primary design calculation. This submodule contains the main loop of the calculation.

6.1.4 Operation module

This module performs a time series based calculation for all types of dikes.

Time series based calculation
This is the main submodule of the time series based calculation. This submodule contains the main loop of the calculation.

6.1.5 NWO Calculation

This module performs an NWO (Niet Waterkerende Objecten) calculation for primary dikes.

Primary NWO calculation
This is the main submodule of the NWO calculation. This submodule contains the main loop of the calculation.

6.2 DAM Engine supporting modules

6.2.1 Failure mechanism wrapper interface

For each failuremechanism kernel a specific wrapper will be written. This wrapper must implement a specific interface, so the DAM Engine can support the use of the failuremechanism kernel. The interface that must be implemented is IFailureMechanism.
Example: Lets say that for the failure mechanism piping we have 3 kernels: Bligh, Sellmeijer and VNK. Then for each of these kernels a calculation wrapper has to be written.
Another example: D-Geo Stability kernel has the ability to calculate the failure mechanism macrostability inwards and the failure mechanism macrostability outwards. In this case 2 wrappers (one for each failure mechanism) are needed for this single kernel.
The next methods are defined in the IFailureMechanism interface

- Prepare()
- Validate()
- Execute()
- Design()
- PostProcess()
- RegisterProgressFeedback()
- RegisterAbortCheck()

Next to that, each wrapper can have properties that hold data that are specific to the failure mechanism.
Example: D-Geostability needs parameters specifying the grid, tangent lines et. These can be passed as properties to the wrapper directly.

6.2.1.1 Prepare

The purpose of this method is to fill a dataobject that implements the IKernelDataInput interface. This dataobject will be needed for the other methods in this interface. The kernel input will be based on the general dam kernel input, the possible addiotional kernel properties and when required calculations in order to determine certain input. Furthermore it initializes the kernel data output (IKernelDataOutput).

```
/// <summary>
/// Prepares the failure mechanism input based on general dam kernel
input and failure mechanism specific properties.
/// </summary>
/// <param name="damKernelInput">The general dam kernel input.</param>
/// <param name="iterationIndex">The number of the current iteration</param>
/// <param name="kernelDataInput">The kernel data input.</param>
/// <param name="kernelDataOutput">The kernel data output.</param>
/// <returns>
/// Result of the prepare
/// </returns>
PrepareResult Prepare(DamKernelInput damKernelInput, int iterationIndex,
out IKernelDataInput damKernelInput, out IKernelDataOutput kernelDataOutput);
```

This method returns:

```
public enum PrepareResult
{
Successful,
Failed,
NotRelevant
};
```

The method has the following parameters:

- `DamKernelInput damKernelInput`: the main input data object; it contains data from the DAM Engine.
- `IKernelDataInput damKernelInput`: in this object the data is filled that is needed by the specific failuremechanism kernel; it will be passed to the failuremechanism kernel as input; each failuremechanism kernel wrapper will have its own implementation of IKernelDataInput.
- `IKernelDataOutput kernelDataOutput`: in this object all the output of the failuremechanism kernel is stored; it is also used for intermediate results; each failuremechanism kernel wrapper will have its own implementation of IKernelDataOutput.

6.2.1.2 *Validate*

```
 /// <summary>
/// Validates the kernel data input.
/// </summary>
/// <param name="kernelDataInput">The kernel data input.</param>
/// <param name="kernelDataOutput">The kernel data output.</param>
/// <param name="messages">The messages.</param>
/// <returns>
/// Number of errors that prevent a calculation
/// </returns>
int Validate(IKernelDataInput kernelDataInput, IKernelDataOutput
kernelDataOutput, out List<LogMessage> messages);
```

The purpose of this method is to validate the data that will be used as input for the failure mechanism.
It returns an integer:
0: no errors. A calculation is possible. It is possible that there are warning messages.
> 0: number of error messages that prevent a calculation. In this case, the calculation result (as part of the IKernelDataOutput) will be set to reflect this.
This method has the following parameters:

- `IKernelDataInput kernelDataInput`: kernel input data.
- `IKernelDataOutput kernelDataOutput`: kernel output data.
- `List<LogMessage> messages`: a list of messages produced by the validation

6.2.1.3 *Execute*

```
 /// <summary>
/// Performs a failure mechanism calculation based on the input.
/// </summary>
/// <param name="kernelDataInput">The kernel data input.</param>
/// <param name="kernelDataOutput">The kernel data output.</param>
/// <param name="messages">The messages.</param>
void Execute(IKernelDataInput kernelDataInput, IKernelDataOutput
kernelDataOutput, out List<LogMessage> messages);
```

This method performs the actual calculation of the failure mechanism.
This method has the following parameters:

- `IKernelDataInput kernelDataInput`: kernel input data.
- `IKernelDataOutput kernelDataOutput`: kernel output data.
- `List<LogMessage> messages`: a list of messages produced by the validation

Each failuremechanism kernel wrapper will have its own implementation of IKernelDataOutput.

6.2.1.4 *Design*

This method implements a design calculation. Based on certain design parameters (e.g. target failure factor, new load parameters, design strategies, etc.) a new design is made for the input data (e.g. a new surfaceline). This method has the following parameters:

- `IKernelDataInput kernelDataInput`: kernel input data.
- `IKernelDataDesignInput kernelDataInput`: design input.
- `IKernelDataOutput kernelDataOutput`: kernel output data.
- `IKernelDataDesignOutput kernelDataOutput`: design output; it contains the adapted input data (a.g. a new designed surfaceline) and other design results (e.g. number of iterations needed, success or failure etc.).
- `List<LogMessage> messages`: a list of messages produced by the design.

Based on the given criteria a new design is determined, which will meet the required criteria. If such a design is not possible, that will be reported back.

6.2.1.5 *PostProcess*

```
 /// <summary>
/// Fills the dam result based on the kernel output.
/// </summary>
/// <param name="damKernelInput">The dam kernel input.</param>
/// <param name="kernelDataOutput">The kernel data output.</param>
/// <param name="resultMessage">The result message.</param>
/// <param name="designResults">The design results.</param>
void PostProcess(DamKernelInput damKernelInput, IKernelDataOutput
kernelDataOutput, string resultMessage, out List<DesignResult> designResults);
```

This method has the following parameters

- `DamKernelInput damKernelInput`: the main dam input data object; it contains data from the DAM Engine.
- `IKernelDataOutput kernelDataOutput`: kernel output data.
- `string resultMessage`: this describes the result of the calculation.
- `DesignResult designResult`: the main dam output data object.

This method fills the DAM Engine Output object with the results of the failuremechanism kernel (IKernelDataOutput).

6.2.1.6 *RegisterProgressFeedback*

This method registers a callback function into the failuremechanism kernel wrapper that can report back progress status from the failuremechanism kernel wrapper to the calling application. The calling application provides the callback function that should be called.

6.2.1.7 *RegisterAbortCheck*

This method registers a callback function into the failuremechanism kernel wrapper. The calling application provides the callback function that should be called. If the function reports back that an abort was requested, the failuremechanism kernel should abort the calculation and return to the calling application with an appropriate error message.

6.2.2 Failure mechanism wrapper implementations

For now the next three implementations of failure mechanism wrappers are foreseen. In the future more can be added. Note also that for a specific failure mechanism multiple implementations can be created. E.g. Piping:

- piping Bligh
- piping Sellmeijer 2 forces
- piping Sellmeijer 4 forces
- piping VNK model

Macrostability inwards
Calculation wrapper for Macrostability inward. Note that (as already mentioned above) for each specific kernel implementation for a failure mechanism, a separate wrapper has to be build (e.g. Slope/W and D-Geo Stability)

Macrostability outwards
Calculation wrapper for Macrostability outward.

Piping
Calculation wrapper for Piping.

6.2.3 Surfaceline designers

A collection of surfaceline designers to support the design calculation. Each designer should adhere to the ISurfaceLineDesigner interface.

Surfaceline Designer Height
Adapts the surfaceline by adding extra height to the dike crest.

Surfaceline Designer Slope
Adapts the surfaceline by changing the slope of the dike on the inside.

Surfaceline Designer Shoulder
Adapts the surfaceline by adding a shoulder or enlarging the shoulder on the inside of the dike.

Surfaceline Designer NWO
Adapts the surfaceline by adding a NWO on a specific place in the surfaceline.

6.2.3.1 Calculation Runner

Failure mechanism Calculation Runner
This submodule calculates a specific failure mechanism by calling the IFailureMechanism interface of the wrapper implementation.

Design Calculation Runner
This submodule performs a design calculation for a specific failure mechanism by calling the IFailureMechanism interface and several surfaceline designers through their ISurfaceLineDesigner interface.

Operational Calculation Runner
This submodule can perform a calculation based on sensor readings (as time-series). The load on the systems (the waternet) will be based on those sensor readings. This can be used in operational systems like DamLive.

Probabilistic Calculation Runner
This submodule performs a probabilistic calculation for a specific location and failure mechanism. The outcome is a failure probability for that location and failure mechanism.

6.2.4 General submodules

Geometry creator
This submodule combines a surfaceline with a subsoil scenario. The output is a geometry that can be used by the failure mechanisms to perform a calculation.

Waternet creator
A waternet describes the waterpressures in the dike embankment. The waterpressures are a result of the load on the system (outer waterlevel and polderlevel). This submodule determines the waternet that will be used by the failure mechanism kernels. At first only the current DAM implementation will be used as a waternet creator. Later on new implementations can be made and applied. E.g. specific for each failure mechanism, or an implementation based on a numerical model like DgFlow.

6.2.4.1 Scripting engine

To enable advanced users to experiment with how the DAM Engine works a Python scripting engine is implemented as a submodule. The scripting engine has acces to the data model and the submodules through well defined interfaces.

7 Programming Interface

This is the definition of the programming interface. The only way to communicate to the DAM Engine is through this interface. In the assembly Deltares.DamEngine.Interface.dll a class hass been defined: `Interface`, which provides the properties and methods which can be used to interact with the DAM Engine.

7.1 Initialization

```
/// <summary>
/// Initializes a new instance of the <see cref="Interface"/> class.
/// </summary>
/// <param name="modelInput">Xml string containing the model input.</param
public EngineInterface(string modelInput)
```

The class has to be instantiated with an (XML) string which adheres to the XSD definition of the inputfile for the DAM Engine (See Appendix A).

7.2 Validation

```
/// <summary>
/// Validates the model.
/// </summary>
/// <returns>Validation messages in an XML string</returns>
public string Validate()
```

This will validate the model and returns the messages in an XML string which adheres to the XSD definition of a message list (See Appendix B).

7.3 Calculation

```
/// <summary>
/// Performs the calculation.
/// </summary>
/// <returns>The output of the calculation in an XML string</returns>
public string Run()
```

This will perform the calculation of the model and returns an XML string which adheres to the XSD definition of the output of the DAM Engine (See Appendix C).

7.4 Interaction

The DAM Engine interacts with the calling application through delegates. The following delegates are used by the DAM Engine:

```
/// <summary>
/// Sends the current progress status
/// </summary>
/// <param name="progress">The progress; this is a number between
0 and 1.</param>
public delegate void ProgressDelegate(double progress);

/// <summary>
/// Sends log message
/// </summary>
/// <param name="logMessage">The log message.</param>
public delegate void SendMessageDelegate(LogMessage logMessage);

/// <summary>
/// Check if a user abort is requested
/// </summary>
/// <returns>true if user requested an abort; else false</returns>
public delegate bool UserAbortDelegate();
```

These delegates can be assigned to the properties of Interface:

```
public ProgressDelegate ProgressDelegate
public SendMessageDelegate SendMessageDelegate
public UserAbortDelegate UserAbortDelegate
```

8 XML Serialization

8.1 Generating serialization code

For the XML serialization a Visual Studio Tool is used. This tool XSD.exe creates objects based on XML schema definitions (*.xsd). To use this tool the following steps should be taken:

- Start the Developer Command Prompt (for VS2015) and go to the folder containing the XSD's.
- Create the classes by running the batchfile "GenerateClasses.bat". This generates 2 source files containing the generated objects (DamInput.cs and DamOutput.cs).
- The 2 source files are then copied to the correct locations in the source tree, so they can be compiled.

The batchfile contains the following lines:

```
REM Start the Developer Command Prompt (for VS2015) and go to this
directory.  Then start this batchfile.
xsd /c /l:cs /n:Deltares.DamEngine.Io.XmlInput DamInput.xsd
copy DamInput.cs ..\src\Deltares.DamEngine.Io\DamInput.cs
xsd /c /l:cs /n:Deltares.DamEngine.Io.XmlOutput DamOutput.xsd
copy DamOutput.cs ..\src\Deltares.DamEngine.Io\DamOutput.cs
```

The classes in the generated source files can be serialized into XML strings using the .NET library class XmlSerializer, which is part of the System.Xml.Serialization assembly.
The following classes are used for transferring the Dam Engine data model into the serializer objects and back:

- FillDamFromXmlInput
- FillXmlInputFromDam
- FillDamFromXmlOutput
- FillXmlOutputFromDam

8.2 Changing the XSD definition

When the interface has to be changed because parameters are added, changed or removed you can do that as follows:

- Change the XSD.
- Regenerate the serializer objects.
- Adapt the transfer objects.

Note: DO NOT manually change the code of the generated objects DamInput.cs and DamOutput.cs, because the changes will be lost when these files are regenerated!

9 Adding a Failure Mechanism

Adding a new failure mechanism to DAM Engine is something that should be releative easy to do. The architecture of DAM Engine has been setup in a way that all failure mechanism specific code is put as much as possible in 1 place.

When a new failure mechanism is to be implemented, several steps have to be taken.

- Add Failure Mechanism Wrapper (this is the main place for all failure mechanism specific code).
- Create an Instance of a Failure Mechanism Wrapper.
- Add Failure Mechanism Specific Data to Data Model.
- Add Failure Mechanism Specific Data to XML Input.
- Add Failure Mechanism Specific Data to XML Output.

In the following sections the steps are illustrated with the piping Bligh Failure mechanism.

9.1 Add Failure Mechanism Wrapper

Create a Failure Mechanism Wrapper that conforms to the Failure Mechanism Wrapper interface as described in section 6.2.1. The implementation of the wrapper for Piping Bligh can be found in the class Deltares.DamEngine.Calculators.KernelWrappers.DamPipingBligh.

9.2 Create an Instance of the Failure Mechanism Wrapper

To be able to use the Failure Mechanism wrapper an instance has to be instantiated. For all failure mechanisms this is done in Deltares.DamEngine.Calculators.KernelWrappers.Common.KernelWrapperHelper.CreateKernelWrapper().

9.3 Add Failure Mechanism Specific Data to Data Model

This is only needed if the data that is already implemented in the DAM Engine is not enough to meet the necessary input data for the failure mechanism kernel. Or if more result data has to be added to accommodate the passing of the failure mechanism output. For piping Bligh e.g. the Soil parameter Soil.PermeabKx has been added (among other parameters) as input. As output the output class PipingDesignResults has been added.

9.4 Add Failure Mechanism Specific Data to XML Input

This is only needed when the data model has been changed (see section 9.3) to accommodate more input properties. To change the input XML definition, the DamInput.xsd definition or one of its dependent xsd's has to be adapted (see Appendix A). Also the reader and writer routines have to be adapted. This can be done in the class Deltares.DamEngine.Interface.FillDamFromXmlInput.

9.5 Add Failure Mechanism Specific Data to XML Output

This is only needed when the data model has been changed (see section 9.3) to accommodate more output properties. To change the output XML definition, the DamOuput.xsd definition or one of its dependent xsd's has to be adapted (see Appendix C). Also the reader and writer routines have to be adapted. This can be done in the class Deltares.DamEngine.Interface.FillXmlOutputFromDam.

10 Failure Mechanism Implementations

10.1 Piping Bligh
TODO...

10.2 Piping Sellmeijer 4 Forces
TODO...

10.3 Piping Sellmeijer VNK
TODO...

10.4 WBI Piping Sellmeijer Revised
TODO...

10.5 Macrostability Inwards
TODO...

10.6 Macrostability Outwards
TODO...

10.7 Macrostability Horizontal Balance
TODO...

10.8 WBI Macrostability Inwards
For WBI a new macrostability kernel has been build. A Functional Design (Zwan, 2017b) and a Technical Design (Markus, 2016) is available. Currently this kernel only supports macrostability inwards for the model Uplift Van. The API of this kernel is based on an XML file that contains all the necessary data for the input of the kernel. The XML is defined with a set of XML schema's (XSD's). These XSD's can be found in chapter 2.3 of the Technical Design (Markus, 2016).

10.8.1 Initial implementation
The first implementation of the WBI macrostability kernel will not be a full implementation. It will implement the same options that were implemented in the original Macrostability Inwards implementation (which uses the D-Geo Stability 18.1 kernel).
Therefore the following input options (see section 10.8.2) will not be implemented:

- PreconsolidationStresses
- ConsolidationValues
- MultiplicationFactorsCPhiForUplift
- SpencerSlipPlanes
- SlipPlaneConstraints
- GeneticAlgorithmOptions
- LevenbergMarquardtOptions
- Waternet creation options (defined in Location)

Furthermore, no Waternet Daily will be specified. This was introduced in the WBI macrostability kernel when POP is requested. Also in the first implementation the waternet will be created by the DAM Engine. The waternet creator of the WBI macrostability kernel will not be used. So the parameters used for the waternet creation do not have to be filled in when calling the WBI macrostability kernel. Those are the parameters defined in Table 10.3.

10.8.2 Mapping of the DAM Engine data

The WBI macrostability kernel has to be filled with input, that can be obtained from the DAM Engine data. In the following tables a mapping of the needed data to the DAM Engine data is defined. The data is contained in the classes DamKernelInput and DamMacroStabilityInput.

WBI Macrostability StabilityModel	**DAM Engine** DamInput.xsd
MoveGrid	Fixed value: TRUE (default)
MaximumSliceWidth	Fixed value: 1.0 (default)
SearchAlgorithm	DamMacroStabilityInput -> FailureMechanismParametersMStab -> MStabParameters -> SearchMethod
ModelOption	DamMacroStabilityInput -> FailureMechanismParametersMStab -> MStabParameters -> Model
Orientation	DamMacroStabilityInput -> FailureMechanismParametersMStab -> MStabParameters -> GridPosition
SoilModel -> Soils	DamKernelInput -> Location -> SoilList (See Table 10.2)
SoilProfile	DamKernelInput -> SubSoilScenario -> SoilProfile2D
SurfaceLine	DamKernelInput -> Location -> SurfaceLine
Location	DamKernelInput -> Location (See Table 10.3)
PreconsolidationStresses	DO-NOT-IMPLEMENT
UniformLoads	generated (based on Location -> StabilityOptions -> Trafficload)
ConsolidationValues	DO-NOT-IMPLEMENT
MultiplicationFactorsCPhiForUplift	DO-NOT-IMPLEMENT
Waternets	generated
SpencerSlipPlanes	DO-NOT-IMPLEMENT
UpliftVanCalculationGrid	generated
SlipPlaneConstraints	DO-NOT-IMPLEMENT (See Table 10.4)
GeneticAlgorithmOptions	DO-NOT-IMPLEMENT
LevenbergMarquardtOptions	DO-NOT-IMPLEMENT

Table 10.1: *Mapping of the WBI macrostability kernel data to the DAM Engine.*

WBI Macrostability Soil	**DAM Engine** Soil
AbovePhreaticLevel	AbovePhreaticLevel
BelowPhreaticLevel	BelowPhreaticLevel
DilatancyType	DilatancyType
Cohesion	Cohesion
FrictionAngle	FrictionAngle
RatioCuPc	RatioCuPc
StrengthIncreaseExponent	StrengthIncreaseExponent
OCR	OCR
ShearStrengthModel	ShearStrengthModel

Table 10.2: *Mapping of the WBI macrostability kernel Soils to the DAM Engine Soils.*

The parameters defined in the following table for Location are all parameters that are used by

the waternet creator of the WBI macrostability kernel. They will not yet be implemented (as explained above).

WBI Macrostability Location	**DAM Engine** Location
DikeSoilScenario	TO-BE-ADDED
WaterLevelRiver	Scenario -> RiverLevel
WaterLevelRiverAverage	TO-BE-ADDED
WaterLevelRiverLow	Scenario -> RiverLevelLow
WaterLevelPolder	PolderLevel
DrainageConstructionPresent	TO-BE-ADDED
XCoordMiddleDrainageConstruction	TO-BE-ADDED
ZCoordMiddleDrainageConstruction	TO-BE-ADDED
MinimumLevelPhreaticLineAtDikeTopRiver	TO-BE-ADDED
MinimumLevelPhreaticLineAtDikeTopRiver	TO-BE-ADDED
UseDefaultOffsets	TO-BE-ADDED
PlLineOffsetBelowPointBRingtoetsWti2017	TO-BE-ADDED
PlLineOffsetBelowDikeTopAtPolder	Scenario -> PlLineOffsetBelowDikeTopAtPolder
PlLineOffsetBelowShoulderBaseInside	Scenario -> PlLineOffsetBelowShoulderBaseInside
PlLineOffsetBelowDikeToeAtPolder	Scenario -> PlLineOffsetBelowShoulderBaseInside
HeadInPLLine2Inwards	HeadPl2
HeadInPLLine3	Scenario -> HeadPl3
HeadInPLLine4	Scenario -> HeadPl4
AdjustPl3And4ForUplift	set to TRUE
PenetrationLength	ModelParametersForPlLines -> PenetrationLength
LeakageLengthOutwardsPl3	DO-NOT-IMPLEMENT
LeakageLengthInwardsPl3	generate based on ModelParametersForPlLines -> DampingFactorPl3
LeakageLengthOutwardsPl4	DO-NOT-IMPLEMENT
LeakageLengthInwardsPl4	generate based on ModelParametersForPlLines -> DampingFactorPl4

Table 10.3: *Mapping of the WBI macrostability kernel Slip Plane Location to the DAM Engine Location.*

The parameters defined in the following table for Constraints will not yet be implemented (as explained above).

WBI Macrostability Constraints	**DAM Engine** Constraints
SlipPlaneMinDepth	Location -> StabiltiyOptions -> MinimumCircleDepth
SlipPlaneMinLength	TO-BE-ADDED
CreateZones	TO-BE-ADDED
AutomaticForbiddenZones	TO-BE-ADDED
XEntryMin	TO-BE-ADDED
XEntryMax	TO-BE-ADDED
MaxAllowedAngleBetweenSlices	TO-BE-ADDED
RequiredForcePointsInSlices	TO-BE-ADDED

Table 10.4: *Mapping of the WBI macrostability kernel Slip Plane Constraints to the DAM Engine data.*

10.8.2.1 Mapping of pl lines to waternet

Dam Engine has a set of pl lines: Pl 1, Pl 2, Pl 3, Pl 4. Pl 1 is the phreatic line and it always exists. The other pl lines are optional. The pl lines from the Dam Engine are converted to a waternet that can be used in WBI Macrostability.

The waternet consists of a phreatic line, a list of head lines and a list of waternet lines. Each waternet line can have an associated head line. To create the waternet lines the soil profile data is used to determine the 'bottom aquifer' (the lowest set of one or more connected aquifer layers) and the 'in-between aquifer' (the first aquifer (layer set) that lies above the bottom aquifer).

Pl 1 becomes the phreatic line.
If Pl 2 exists it becomes a head line. If there is at least one aquifer, a waternet line is created along the top of the bottom aquifer + penetration length.
If Pl 3 exists it becomes a head line. If there are at least two aquifers, a waternet line is created along the top of the in-between aquifer.
If Pl 4 exists it becomes a head line. If there is at least one aquifer, a waternet line is created along the top of the bottom aquifer.

10.8.3 Mapping of the validation result

The WBI macrostability kernel returns the validation result when the Validate function is called. In the following table a mapping of the validation result to the DAM Engine data is defined.

WBI Macrostability WTIStabilityModelValidation.xsd	**DAM Engine** DamDesignResult.xsd -> StabilityDesignResults
Validations -> ValidationsType	ResultMessage -> LogMessageType

Table 10.5: *Mapping of the WBI macrostability kernel validation result to the DAM Engine.*

When the Validations part is empty it means that the input is Valid. When there are one or more validations, they are added to the messages of the design results. The type of message can be Info, Warning or Error. The validation fails when there is at least one Error message.

10.8.4 Mapping of the calculation result

The WBI macrostability kernel returns the calculation result when the Run function is called. In the following table a mapping of the calculation result to the DAM Engine data is defined. For now, only the parts that we currently use are described.

WBI Macrostability WTIStabilityModel-Result.xsd -> WTIStabilityModelResult	**DAM Engine** DamDesignResult.xsd -> StabilityDe-signResults
Calculated	CalculationResult
SafetyFactor	SafetyFactor
Messages	ResultMessage
MinimumSafetyCurve -> first Slice, TopLeftPoint X	CircleSurfacePointLeftXCoordinate
MinimumSafetyCurve -> last Slice, TopRightPoint X	CircleSurfacePointRightXCoordinate
ModelOption	StabilityModelType

Table 10.6: *Mapping of the WBI macrostability kernel validation result to the DAM Engine.*

Calculated is a boolean. When Calculated is true the CalculationResult is Succeeded, otherwise RunFailed. The type of message can be Info, Warning or Error. Presumed is that when Calculated is true, there are no error messages.

11 Literature

Doxygen, 2017. *DAM Engine - Technical documentation, Generated by Doxygen 1.8.10*. Tech. rep., Deltares.

Kleijn, E., A. Grijze, H. Elzinga, S. Hummel and T. The, 2017. *Architecture Guidelines*. Tech. rep., Deltares.

Markus, A., 2016. *WTI2017 Failure Mechanisms - MacroStability Kernel, Technical Design*. Tech. Rep. 1230095-002-HYE-0012, version 1.4, final, Deltares.

The, T., 2017a. *DAM Architecture Overall*. Tech. Rep. 1210702-000-GEO-0005, version 0.1, jan. 2017, concept, Deltares.

The, T., 2017b. *DAM Engine - Technical Design*. Tech. Rep. 1210702-000-GEO-0004, version 0.2, mar. 2017, concept, Deltares.

Trompille, V., 2017a. *DAM Engine - Test Plan*. Tech. Rep. 1210702-000-GEO-0006, version 0.1, jan. 2017, concept, Deltares.

Trompille, V., 2017b. *DAM Engine - Test Report*. Tech. Rep. 1210702-000-GEO-0007, version 0.1, jan. 2017, concept, Deltares.

Zwan, I. v., 2017a. *DAM Engine - Functional Design*. Tech. Rep. 1210702-000-GEO-0003, version 0.1, jan. 2017, concept, Deltares.

Zwan, v. d. I., 2017b. *WTI-2017 Failure mechanisms - MacroStability kernel, Requirements and Functional Design*. Tech. Rep. 12300953-002-HYE-0014, version 4, final, Deltares.

A DamInput

These are the XSD's that apply to the input XML of the DAM Engine.

A.1 DamInput.xsd

This is the DamInput XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema xmlns:ns0="http://deltares.nl/2008/ProfileDefinition" elementFormDefault="qualified" xm
  <xs:include schemaLocation=".\DamLocation.xsd" />
  <xs:include schemaLocation=".\DamSurfaceLine.xsd" />
  <xs:include schemaLocation=".\DamSoil.xsd" />
  <xs:include schemaLocation=".\DamSegment.xsd" />
  <xs:include schemaLocation=".\DamSoilProfile1D.xsd" />
  <xs:include schemaLocation=".\DamSoilProfile2D.xsd" />
  <xs:include schemaLocation=".\DamStabilityParameters.xsd" />
  <xs:include schemaLocation=".\DamTimeSerie.xsd" />
  <xs:include schemaLocation=".\DamSensor.xsd" />
  <xs:include schemaLocation=".\DamSensorLocation.xsd" />
  <xs:include schemaLocation=".\DamSensorGroup.xsd" />
  <xs:complexType name="Input">
    <xs:sequence>
      <xs:element name="Locations">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" name="Location" type="Location" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="SurfaceLines">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" name="SurfaceLine" type="SurfaceLine"
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Soils">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" name="Soil" type="Soil" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Segments">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" name="Segment" type="Segment" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="SoilProfiles1D">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" name="SoilProfile1D" type="SoilProfile
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="SoilProfiles2D">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" name="SoilProfile2D" type="SoilProfile
          </xs:sequence>
```

```
      </xs:complexType>
    </xs:element>
    <xs:element name="StabilityParameters" type="StabilityParameters" />
    <xs:element name="AquiferSoils" nillable="true">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" maxOccurs="unbounded" name="AquiferSoil">
            <xs:complexType>
              <xs:attribute name="Soilname" type="xs:string" use="required" />
              <xs:attribute name="IsAquifer" type="xs:boolean" use="required" />
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" maxOccurs="1" name="OperationalInputTimeSeries" nillable="true">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" maxOccurs="unbounded" name="TimeSerie" type="TimeSerie" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" maxOccurs="1" name="SensorData" nillable="true">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="1" maxOccurs="1" name="SensorGroups">
            <xs:complexType>
              <xs:sequence>
                <xs:element minOccurs="0" maxOccurs="unbounded" name="SensorGroup" type="SensorGroup" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element minOccurs="1" maxOccurs="1" name="Sensors">
            <xs:complexType>
              <xs:sequence>
                <xs:element minOccurs="0" maxOccurs="unbounded" name="Sensor" type="Sensor" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="SensorLocations">
            <xs:complexType>
              <xs:sequence>
                <xs:element minOccurs="0" maxOccurs="unbounded" name="SensorLocation" type="SensorLocati
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="DamProjectType" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="Operational" />
        <xs:enumeration value="Design" />
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="FailureMechanismSystemType" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:int">
        <xs:enumeration value="StabilityInside" />
        <xs:enumeration value="StabilityOutside" />
        <xs:enumeration value="Piping" />
        <xs:enumeration value="HorizontalBalance" />
```

```
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="PipingModelType" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="Bligh" />
          <xs:enumeration value="SellmeijerVnk" />
          <xs:enumeration value="Sellmeijer4Forces" />
          <xs:enumeration value="WtiSellmeijerRevised" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="StabilityModelType" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="Bishop" />
          <xs:enumeration value="UpliftVan" />
          <xs:enumeration value="BishopUpliftVan" />
          <xs:enumeration value="UpliftVanWti" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="AnalysisType" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="NoAdaption" />
          <xs:enumeration value="AdaptGeometry" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="ProjectPath" type="xs:string" use="optional" />
    <xs:attribute name="CalculationMap" type="xs:string" use="optional" />
    <xs:attribute name="MaxCalculationCores" type="xs:int" use="optional" />
  </xs:complexType>
  <xs:element name="Input" type="Input" />
</xs:schema>
```

A.2 DamLocation.xsd

This is the Location XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="Location">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="OperationalOptions" />
      <xs:element minOccurs="0" maxOccurs="1" name="DesignOptions">
        <xs:complexType>
          <xs:attribute name="RedesignDikeHeight" type="xs:boolean" use="required" />
          <xs:attribute name="RedesignDikeShoulder" type="xs:boolean" use="required" />
          <xs:attribute name="ShoulderEmbankmentMaterial" type="xs:string" use="required" />
          <xs:attribute name="StabilityShoulderGrowSlope" type="xs:double" use="required" />
          <xs:attribute name="StabilityShoulderGrowDeltaX" type="xs:double" use="required" />
          <xs:attribute name="StabilitySlopeAdaptionDeltaX" type="xs:double" use="required" />
          <xs:attribute name="SlopeAdaptionStartCotangent" type="xs:double" use="required" />
          <xs:attribute name="SlopeAdaptionEndCotangent" type="xs:double" use="required" />
          <xs:attribute name="SlopeAdaptionStepCotangent" type="xs:double" use="required" />
          <xs:attribute name="NewDikeTopWidth" type="xs:double" use="optional" />
          <xs:attribute name="NewDikeSlopeInside" type="xs:double" use="optional" />
          <xs:attribute name="NewDikeSlopeOutside" type="xs:double" use="optional" />
          <xs:attribute name="NewShoulderTopSlope" type="xs:double" use="optional" />
          <xs:attribute name="NewShoulderBaseSlope" type="xs:double" use="optional" />
          <xs:attribute name="NewMaxHeightShoulderAsFraction" type="xs:double" use="optional" />
          <xs:attribute name="NewMinDistanceDikeToeStartDitch" type="xs:double" use="optional" />
```

```
      <xs:attribute name="UseNewDitchDefinition" type="xs:boolean" use="required" />
      <xs:attribute name="NewWidthDitchBottom" type="xs:double" use="optional" />
      <xs:attribute name="NewSlopeAngleDitch" type="xs:double" use="optional" />
      <xs:attribute name="NewDepthDitch" type="xs:double" use="optional" />
      <xs:attribute name="StabilityDesignMethod" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="OptimizedSlopeAndShoulderAdaption" />
            <xs:enumeration value="SlopeAdaptionBeforeShoulderAdaption" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="WaternetOptions">
    <xs:complexType>
      <xs:attribute name="PhreaticLineCreationMethod" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="ExpertKnowledgeRRD" />
            <xs:enumeration value="ExpertKnowledgeLinearInDike" />
            <xs:enumeration value="GaugesWithFallbackToExpertKnowledgeRRD" />
            <xs:enumeration value="Sensors" />
            <xs:enumeration value="None" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="PolderLevel" type="xs:double" use="required" />
      <xs:attribute name="HeadPl2" type="xs:double" use="optional" />
      <xs:attribute name="DampingFactorPl3" type="xs:double" use="required" />
      <xs:attribute name="DampingFactorPl4" type="xs:double" use="required" />
      <xs:attribute name="PenetrationLength" type="xs:double" use="required" />
      <xs:attribute name="SlopeDampingFactor" type="xs:double" use="required" />
      <xs:attribute name="IntrusionVerticalWaterPressure" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Standard" />
            <xs:enumeration value="Linear" />
            <xs:enumeration value="FullHydroStatic" />
            <xs:enumeration value="HydroStatic" />
            <xs:enumeration value="SemiTimeDependent" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="DikeSoilScenario" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="ClayDikeOnClay" />
            <xs:enumeration value="SandDikeOnClay" />
            <xs:enumeration value="ClayDikeOnSand" />
            <xs:enumeration value="SandDikeOnSand" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="General">
    <xs:complexType>
      <xs:attribute name="Description" type="xs:string" />
      <xs:attribute name="HeadPL2" type="xs:double" use="optional" />
      <xs:attribute name="HeadPL3" type="xs:double" use="optional" />
      <xs:attribute name="HeadPL4" type="xs:double" use="optional" />
    </xs:complexType>
  </xs:element>
  <xs:element name="DesignScenarios">
```

```
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="1" maxOccurs="unbounded" name="DesignScenario">
            <xs:complexType>
              <xs:attribute name="Id" type="xs:string" use="required" />
              <xs:attribute name="RiverLevel" type="xs:double" use="required" />
              <xs:attribute name="RiverLevelLow" type="xs:double" use="optional" />
              <xs:attribute name="DikeTableHeight" type="xs:double" use="optional" />
              <xs:attribute name="PlLineOffsetBelowDikeTopAtRiver" type="xs:double" use="require
              <xs:attribute name="PlLineOffsetBelowDikeTopAtPolder" type="xs:double" use="requir
              <xs:attribute name="PlLineOffsetBelowShoulderBaseInside" type="xs:double" use="re
              <xs:attribute name="PlLineOffsetBelowDikeToeAtPolder" type="xs:double" use="requir
              <xs:attribute name="PlLineOffsetBelowDikeCrestMiddle" type="xs:double" use="optior
              <xs:attribute name="PlLineOffsetFactorBelowShoulderCrest" type="xs:double" use="o
              <xs:attribute name="HeadPl3" type="xs:double" use="optional" />
              <xs:attribute name="HeadPl4" type="xs:double" use="optional" />
              <xs:attribute name="UpliftCriterionStability" type="xs:double" use="required" />
              <xs:attribute name="UpliftCriterionPiping" type="xs:double" use="required" />
              <xs:attribute name="RequiredSafetyFactorStabilityInnerSlope" type="xs:double" use=
              <xs:attribute name="RequiredSafetyFactorStabilityOuterSlope" type="xs:double" use=
              <xs:attribute name="RequiredSafetyFactorPiping" type="xs:double" use="required" />
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" maxOccurs="1" name="StabilityOptions">
      <xs:complexType>
        <xs:attribute name="MapForSoilgeometries2D" type="xs:string" use="optional" />
        <xs:attribute name="SoilDatabaseName" type="xs:string" use="optional" />
        <xs:attribute name="ZoneType" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:int">
              <xs:enumeration value="NoZones" />
              <xs:enumeration value="ZoneAreas" />
              <xs:enumeration value="ForbiddenZones" />
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="ForbiddenZoneFactor" type="xs:double" use="optional" />
        <xs:attribute name="ZoneAreaRestSlopeCrestWidth" type="xs:double" use="optional" />
        <xs:attribute name="TrafficLoad" type="xs:double" use="optional" />
        <xs:attribute name="MinimumCircleDepth" type="xs:double" use="optional" />
        <xs:attribute name="TrafficLoadDegreeOfConsolidation" type="xs:double" use="optional" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="SurfaceLineName" type="xs:string" use="required" />
  <xs:attribute name="SegmentName" type="xs:string" use="required" />
  <xs:attribute name="Name" type="xs:string" use="required" />
  <xs:attribute name="DikeEmbankmentMaterial" type="xs:string" use="required" />
  <xs:attribute name="XSoilGeometry2DOrigin" type="xs:double" use="optional" />
  <xs:attribute name="DistanceToEntryPoint" type="xs:double" use="optional" />
 </xs:complexType>
</xs:schema>
```

A.3 DamSurfaceLine.xsd

This is the DamSurfaceLine XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="SurfaceLine">
    <xs:sequence>
      <xs:element name="Points">
```

```
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="2" maxOccurs="unbounded" name="Point">
            <xs:complexType>
              <xs:attribute name="X" type="xs:double" use="required" />
              <xs:attribute name="Z" type="xs:double" use="required" />
              <xs:attribute name="PointType" use="required">
                <xs:simpleType>
                  <xs:restriction base="xs:int">
                    <xs:enumeration value="None" />
                    <xs:enumeration value="SurfaceLevelOutside" />
                    <xs:enumeration value="DikeToeAtRiver" />
                    <xs:enumeration value="ShoulderTopOutside" />
                    <xs:enumeration value="ShoulderBaseOutside" />
                    <xs:enumeration value="DikeTopAtRiver" />
                    <xs:enumeration value="DikeLine" />
                    <xs:enumeration value="TrafficLoadOutside" />
                    <xs:enumeration value="TrafficLoadInside" />
                    <xs:enumeration value="DikeTopAtPolder" />
                    <xs:enumeration value="ShoulderBaseInside" />
                    <xs:enumeration value="ShoulderTopInside" />
                    <xs:enumeration value="DikeToeAtPolder" />
                    <xs:enumeration value="DitchDikeSide" />
                    <xs:enumeration value="BottomDitchDikeSide" />
                    <xs:enumeration value="BottomDitchPolderSide" />
                    <xs:enumeration value="DitchPolderSide" />
                    <xs:enumeration value="SurfaceLevelInside" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:attribute>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="Name" type="xs:string" use="required" />
 </xs:complexType>
</xs:schema>
```

A.4 DamSoil.xsd

This is the DamSoil XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="Soil">
    <xs:attribute name="Name" type="xs:string" use="required" />
    <xs:attribute name="BeddingAngle" type="xs:double" use="optional" />
    <xs:attribute name="DiameterD70" type="xs:double" use="optional" />
    <xs:attribute name="PermeabKx" type="xs:double" use="optional" />
    <xs:attribute name="WhitesConstant" type="xs:double" use="optional" />
    <xs:attribute name="DiameterD90" type="xs:double" use="optional" />
    <xs:attribute name="AbovePhreaticLevel" type="xs:double" use="optional" />
    <xs:attribute name="BelowPhreaticLevel" type="xs:double" use="optional" />
    <xs:attribute name="DryUnitWeight" type="xs:double" use="optional" />
    <xs:attribute name="ShearStrengthModel" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="None" />
          <xs:enumeration value="CPhi" />
          <xs:enumeration value="StressTable" />
          <xs:enumeration value="PseudoValues" />
          <xs:enumeration value="SuMeasured" />
          <xs:enumeration value="SuCalculated" />
```

```
            <xs:enumeration value="SuGradient" />
            <xs:enumeration value="SuCalculatedWithYield" />
            <xs:enumeration value="CPhiOrSuCalculated" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="UseDefaultShearStrengthModel" type="xs:boolean" use="optional" />
      <xs:attribute name="Cohesion" type="xs:double" use="optional" />
      <xs:attribute name="FrictionAngle" type="xs:double" use="optional" />
      <xs:attribute name="Ocr" type="xs:double" use="optional" />
      <xs:attribute name="SlopeRestProfile" type="xs:double" use="optional" />
      <xs:attribute name="DilatancyType" use="optional">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Phi" />
            <xs:enumeration value="Zero" />
            <xs:enumeration value="MinusPhi" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
</xs:schema>
```

A.5 DamSegment.xsd

This is the DamSegment XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="Segment">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="SoilGeometryProbability">
        <xs:complexType>
          <xs:attribute name="SoilProfileName" type="xs:string" use="required" />
          <xs:attribute name="SoilProfileType" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:int">
                <xs:enumeration value="ProfileType1D" />
                <xs:enumeration value="ProfileType2D" />
                <xs:enumeration value="ProfileTypeStiFile" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="SegmentFailureMechanismType " use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:int">
                <xs:enumeration value="All" />
                <xs:enumeration value="Stability" />
                <xs:enumeration value="Piping" />
                <xs:enumeration value="Liquefaction" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="Probability" type="xs:double" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="Name" type="xs:string" use="required" />
  </xs:complexType>
</xs:schema>
```

A.6 DamSoilProfile1D.xsd

This is the Dam SoilProfile XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="SoilProfile1D">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="Layers1D">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="unbounded" name="Layer1D">
              <xs:complexType>
                <xs:attribute name="Name" type="xs:string" use="required" />
                <xs:attribute name="SoilName" type="xs:string" use="required" />
                <xs:attribute name="TopLevel" type="xs:double" use="required" />
                <xs:attribute name="IsAquifer" type="xs:boolean" use="required" />
                <xs:attribute name="WaterpressureInterpolationModel" use="required">
                  <xs:simpleType>
                    <xs:restriction base="xs:int">
                      <xs:enumeration value="Automatic" />
                      <xs:enumeration value="Hydrostatic" />
                    </xs:restriction>
                  </xs:simpleType>
                </xs:attribute>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="Name" type="xs:string" use="required" />
    <xs:attribute name="BottomLevel" type="xs:double" use="required" />
  </xs:complexType>
</xs:schema>
```

A.7 DamSoilProfile2D.xsd

This is the DamSoilProfile2D XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="SoilProfile2D">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="Layers2D">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="unbounded" name="Layer2D">
              <xs:complexType>
                <xs:sequence>
                  <xs:element minOccurs="1" maxOccurs="1" name="Surface">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element minOccurs="1" maxOccurs="1" name="OuterLoop">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element minOccurs="3" maxOccurs="unbounded" name="OuterPoint">
                                <xs:complexType>
                                  <xs:attribute name="X" type="xs:double" use="required" />
                                  <xs:attribute name="Z" type="xs:double" use="required" />
                                </xs:complexType>
                              </xs:element>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
```

```
                        <xs:element minOccurs="0" maxOccurs="1" name="Innerloop">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element minOccurs="3" maxOccurs="unbounded" name="InnerPoint">
                                <xs:complexType>
                                  <xs:attribute name="X" type="xs:double" use="required" />
                                  <xs:attribute name="Z" type="xs:double" use="required" />
                                </xs:complexType>
                              </xs:element>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="SoilName" type="xs:string" use="required" />
                <xs:attribute name="IsAquifer" type="xs:boolean" use="required" />
                <xs:attribute name="WaterpressureInterpolationModel" use="required">
                  <xs:simpleType>
                    <xs:restriction base="xs:int">
                      <xs:enumeration value="Automatic" />
                      <xs:enumeration value="Hydrostatic" />
                    </xs:restriction>
                  </xs:simpleType>
                </xs:attribute>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element minOccurs="0" maxOccurs="1" name="PreconsolidationStresses">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" name="PreconsolidationStress">
              <xs:complexType>
                <xs:attribute name="StressValue" type="xs:double" use="required" />
                <xs:attribute name="Name" type="xs:string" use="required" />
                <xs:attribute name="X" type="xs:double" use="required" />
                <xs:attribute name="Z" type="xs:double" use="required" />
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="Name" type="xs:string" use="required" />
  </xs:complexType>
</xs:schema>
```

A.8 DamStabilityParameters.xsd

This is the DamStabilityParameters XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="StabilityParameters">
    <xs:attribute name="SearchMethod" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:int">
          <xs:enumeration value="Calculationgrid" />
          <xs:enumeration value="GeneticAlgorithm" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
```

```
    <xs:attribute name="GridDetermination" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:int">
          <xs:enumeration value="Automatic" />
          <xs:enumeration value="Specified" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="BishopTangentLinesDefinition" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:int">
          <xs:enumeration value="OnBoundaryLines" />
          <xs:enumeration value="Specified" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="BishopTangentLinesDistance" type="xs:double" use="optional" />
    <xs:attribute name="BishopGridVerticalPointsCount" type="xs:int" use="optional" />
    <xs:attribute name="BishopGridVerticalPointsDistance" type="xs:double" use="optional" />
    <xs:attribute name="BishopGridHorizontalPointsCount" type="xs:int" use="optional" />
    <xs:attribute name="BishopGridHorizontalPointsDistance" type="xs:double" use="optional" />
    <xs:attribute name="UpliftVanGridLeftVerticalPointsCount" type="xs:int" use="optional" />
    <xs:attribute name="UpliftVanGridLeftVerticalPointsDistance" type="xs:double" use="optional" />
    <xs:attribute name="UpliftVanGridLeftHorizontalPointsCount" type="xs:int" use="optional" />
    <xs:attribute name="UpliftVanGridLeftHorizontalPointsDistance" type="xs:double" use="optional" />
    <xs:attribute name="UpliftVanGridRightVerticalPointsCount" type="xs:int" use="optional" />
    <xs:attribute name="UpliftVanGridRightVerticalPointsDistance" type="xs:double" use="optional" />
    <xs:attribute name="UpliftVanGridRightHorizontalPointsCount" type="xs:int" use="optional" />
    <xs:attribute name="UpliftVanGridRightHorizontalPointsDistance" type="xs:double" use="optional" />
    <xs:attribute name="UpliftVanTangentLinesDefinition" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:int">
          <xs:enumeration value="OnBoundaryLines" />
          <xs:enumeration value="Specified" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="UpliftVanTangentLinesDistance" type="xs:double" use="optional" />
  </xs:complexType>
</xs:schema>
```

B Messages

These are the XSD's that apply to the messages XML of the DAM Engine.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="Message">
    <xs:attribute name="MessageType" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="Error" />
          <xs:enumeration value="Warning" />
          <xs:enumeration value="Info" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Message" type="xs:string" use="required" />
  </xs:complexType>
</xs:schema>
```

C DamOutput

These are the XSD's that apply to the output XML of the DAM Engine.

C.1 DamOutput.xsd

This is the DamOutput XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation=".\Message.xsd" />
  <xs:include schemaLocation=".\DamCalculationResults.xsd" />
  <xs:include schemaLocation=".\DamTimeSerie.xsd" />
  <xs:complexType name="Output">
    <xs:sequence>
      <xs:element name="Results">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="CalculationResults" type="CalculationResults" />
            <xs:element minOccurs="0" maxOccurs="1" name="OperationalOutputTimeSeries" nillable="
              <xs:complexType>
                <xs:sequence>
                  <xs:element minOccurs="0" maxOccurs="unbounded" name="TimeSerie" type="TimeSerie
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="CalculationMessages">
              <xs:complexType>
                <xs:sequence>
                  <xs:element minOccurs="0" maxOccurs="unbounded" name="Message" type="Message" />
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="ValidationResults">
        <xs:complexType>
          <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element minOccurs="0" maxOccurs="unbounded" name="ValidationMessages" type="Messag
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Output" type="Output" />
</xs:schema>
```

C.2 Message.xsd

This is the Message XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="Message">
    <xs:attribute name="MessageType" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="Error" />
          <xs:enumeration value="Warning" />
          <xs:enumeration value="Info" />
        </xs:restriction>
```

```
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Message" type="xs:string" use="required" />
  </xs:complexType>
</xs:schema>
```

C.3 DamCalculationResults.xsd

This is the DamCalculationResults XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation=".\DamTimeSerie.xsd" />
  <xs:include schemaLocation=".\DamRegionalScenarioResult.xsd" />
  <xs:include schemaLocation=".\DamDesignResult.xsd" />
  <xs:complexType name="CalculationResults">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="DesignResults" type="DesignResult" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```