Module Guide
for
Two and Three Dimensional Dynamic Model of Soil-Water-Structure Interaction

Prepared by:
Brandon Karchewski (karcheba@mcmaster.ca)
Ph.D. Candidate
Department of Civil Engineering

Prepared for:
Dr. Spencer Smith
CES 741 - Development of Scientific Computing Software
Department of Computational Engineering and Science

McMaster University
Hamilton, Ontario, Canada

Ver. DynSWS-MG-1.0
March 5, 2012

Table of Contents

Table of Symbols

Table 0.1: Document list prefixes

Symbol	Description
AC	Anticipated Change
G	Goal Statement
NFR	Non-functional Requirement
UC	Unlikely Change
UG	User Group

Acronyms and Abbreviations

2-D/3-D: Two-dimensional/three-dimensional; refers to the dimension of the coordinate system used to solve the problem.

BC: Boundary condition; includes kinematic (Dirichlet) and natural (Neumann) boundary conditions.

DynSWS: Dynamic model of Soil-Water-Structure interaction; the software product described herein.

MG: Module guide; the name attributed to this document.

PDE: Partial differential equation. Models of physical phenomena are typically stated mathematically as systems of this type of equation that must be integrated in order to obtain the solution. Initial and/or boundary conditions are also required for a given problem.

SRS: Software requirements specification; the document that specifies the requirements for a software product (see DynSWS-SRS-1.0).

Quick Reference Tables

Table 0.2: Index of User Groups

Table 0.3: Index of Anticipated Changes

Table 0.4: Index of Unlikely Changes

Table 0.5: Index of Leaf Modules

Table 0.6: Index of Goal Statements (see reference [1] for details)

Item	Description
G1	Compute displacement, velocity, and acceleration field
G2	Compute stress and strain field

Table 0.7: Index of Non-Functional Requirements (see reference [1] for details)

Item	Description
NFR1	Sensitivity for test cases
NFR2	Approximation error
NFR3	Convergence iterations
NFR4	List of test cases
NFR5	Validation against laboratory results
NFR6	Validation against field measurements
NFR7	Single executable file
NFR8	Interaction through input-output files
NFR9	Performance for test cases
NFR10	Abstraction of 2-D and 3-D versions of code
NFR11	Insulation of input-output file format
NFR12	Insulation of calculation data structure
NFR13	Run on Windows and MacOS
NFR14	Protection of intellectual property

1 Introduction

This section introduces the MG for the DynSWS software product. Readers not yet acquainted with DynSWS should see reference [1], which is the software requirements specification for the software product described herein. Section 1.1 describes the purpose of this MG. Section 1.2 explains how the requirements for DynSWS have guided the high level design of the modules. Section 1.3 relates the scope of DynSWS presented in this document to the scope of the software requirements specification (if there is any difference). Section 1.4 identifies the intended audience for this document. Section 1.5 outlines the organization of the MG.

1.1 Purpose

This document presents the high level design of the modules that compose DynSWS. The modularization is based on the principle of information hiding, a concept that was introduced (although not named as such) by Parnas [2]. The key to this form of modularization is that each module typically has one secret (such as the algorithm that is used in the implementation) and typically provides one service (such calculation of a value). This framework has the goal of isolating any one design decision to a single module. The benefits of this framework are that modules can often be worked on in parallel (provided that the interface between modules is known), tests of isolated portions of the code are easier to design and implement, and future changes are more easily accommodated since only one module need be changed. Interested readers should see reference [3] for much more information on information hiding and other important concepts in software engineering.

1.2 Bridge Between Requirements and Design

It is important to note that, while the SRS for DynSWS documents the requirements of the software product, it says nothing of how the requirements are to be achieved. This MG presents the high-level design, which is the first stage of the actual design of DynSWS. The overriding design principle used is that of information hiding. Design according to this principle begins by examining the specifications and identifying the aspects of the software product that are anticipated to change and those that are unlikely to change. Each anticipated change is then associated with a single module that keeps the "secret" of that change. Finally, related modules are grouped together into larger modules according to task or purpose with the highest level being the standard modules for behaviour hiding, software decision hiding, and machine hiding (see Section 3.1 for a more detailed description of these top-level modules). Note that not all of the modules presented in the MG will necessarily be implemented in DynSWS as some of the services are provided by, for example, the programming language. However, all modules that provide a service that is needed for DynSWS to meet its requirements are listed to ensure that the design is transparent.

1.3 Scope

The scope of the design of DynSWS presented in this MG is the same as that of the SRS. That is, this document contains the complete design for DynSWS to meet its functional and non-functional requirements as presented in the SRS.

1.4 Intended Audience

The three main groups that the MG is intended for use by are:

UG1. Developers. Users in this group are involved in the actual implementation of the requirements of DynSWS. This will certainly include the author, but may include others in the future if the software product proves useful and the functionality continues to be extended over time. This group can use the MG as a reference to the high level design of DynSWS, which will aid in determining what services are already available and where new features should fit. If users from this group add to the module hierarchy, they must also update the MG to reflect these additions.

UG2. Maintainers. Users in this group maintain the software product over time. This may include activities such as performing tests, fixing bugs, and reorganizing the module hierarchy to reflect design modifications. Again, this will initially be just the author, but in the future may include others. If the design is modified by users in this group, changes should be documented in the MG.

UG3. Reviewers. Users in this group have the task of ensuring that DynSWS meets all requirements and that the results produced by the software product are correct (insofar as correctness can be determined). This includes the author, but also the author's supervisory committee as they will be responsible for verifying the correctness and accuracy of the model contained in DynSWS. The MG will be useful for this group in understanding how the implementation is organized so that it can be reviewed in a systematic manner.

It should be noted that the MG is not necessarily intended for end users of the software product. The MG presents a high-level design of the implementation without going into detail on the requirements (except to name them in relation to the modules) or the implementation (as this aspect comes later in the development process). Readers interested in the requirements specification for DynSWS should see reference [1].

1.5 Organization of the Document

The MG is organized in a top-down manner. Since one of the most important benefits of modularization through the principle of information hiding is the facilitation of changes to the design, Section 2 presents the changes that may influence the design. Section 3 shows the actual modules by first presenting the module hierarchy diagrammatically (Section 3.1), then describing the secrets and services of the modules (Section 3.2), and finally showing the interdependencies of the modules (Section 3.3). Section 4 contains a series of tables that

show how each of the modules relate to the requirements for DynSWS. The intention of organizing the MG in this fashion is to present the big picture of the design first and then delve progressively into more detail on each element of the design.

2 Potential Changes

This section lists changes that may occur in the design. It is important to consider potential changes at this stage since they will have an important influence on the module decomposition. In particular, Section 2.1 lists changes that are likely to occur and that the module decomposition will specifically aim to accommodate. Section 2.2 lists changes that are possible, but not very likely to occur; although this second list of changes will be kept in mind, the design will not specifically target the ability to easily make the changes that are deemed unlikely.

2.1 Anticipated Changes

This section lists changes that are likely to be made to DynSWS, which will guide its design, chiefly, the module decomposition. The first set of anticipated changes relate to the underlying theoretical model that was presented in the SRS document:

AC1. Consideration of large deformations and large strains. The first implementation of DynSWS will not account for large strains, but for analysis of conditions approaching and exceeding failure it is likely that this assumption will need to be modified.

AC2. The material model for the structural subdomain. Initially, the structural subdomain will be modelled as linear elastic. Materials such as concrete, of which the type of structures that DynSWS is intended to model are often constructed, only behave in this manner for small strains. Coinciding with **AC1**, the material model for the structural subdomain is likely to change.

AC3. The material model for the solid phase of the soil subdomain in the non-Newtonian regime. The assumed material model for soil at low porosity levels is linear elastic. It is well known that soil is not a linear elastic material, so this assumption is very likely to change.

AC4. The use of a Cartesian coordinate system. As mentioned in the SRS for DynSWS, certain types of geometry for soil-water-structure interaction problems are best represented in coordinate systems other than the Cartesian system (*e.g.* cylindrical coordinates).

The following are additional changes that are anticipated in the implementation of DynSWS:

AC5. The system of files that contain the input data. It is expected that a system of multiple files will be used to store the input data, but the number of files and the naming convention for the files is unknown and likely to change.

AC6. The format for storing domain geometry data in its associated input file. This data will change, for example, depending on whether a 2-D or 3-D model is used. As such, the file storage format is likely to change.

AC7. The format for storing boundary geometry data in its associated input file. This data is likely to change for reasons similar to those described in **AC6**.

AC8. The format for storing material property data in its associated input file. The number of material parameters is likely to change as the material model changes, which will result in changes to the file format.

AC9. The format for storing initial conditions of vector field variables in their associated input file. This includes displacement, velocity, and acceleration. Changes in the instanced model, for example, from 2-D to 3-D will affect the size of field variable data structures. Thus, the storage format for field variable data is likely to change.

AC10. The format for storing initial conditions of tensor field variables in their associated input file. This includes stress, strain, strain rate, and pressure. The format of this data will change as a result of material model changes as well as the changes mentioned in **AC9**.

AC11. The format for storing kinematic boundary conditions in their associated input file. This type of boundary condition involves specified displacement, velocity, and acceleration data. This data will change for similar reasons to those mentioned in **AC9**, but boundary condition data will require a different storage format than initial condition data since the values may vary over time.

AC12. The format for storing natural boundary conditions in their associated input file. This type of boundary condition involves specified surface stresses. This will change for similar reasons to those described in **AC10** and **AC11**.

AC13. The format for storing body force data in its associated input file. This is likely to change for similar reasons to the other input file formats.

AC14. The system of files that contain the output data. This is likely to change in a similar manner and for similar reasons as those described in **AC5**.

AC15. The format for storing vector field data in its associated output file. This is likely to change in a similar manner to that described in **AC9** and **AC11**.

AC16. The format for storing tensor field data in its associated output file. This is likely to change in a similar manner to that described in **AC10** and **AC12**.

AC17. The system for reporting progress and error messages. There are a number of ways that the task of reporting success or failure of the computation and the implementation of this task is likely to change.

AC18. The format and content of progress and error messages. As the model develops, it is likely that progress and error messages will need to be added, modified, and deleted.

AC19. The values of system constants. The SRS for DynSWS lists a number of system constants defining items such as data constraints. As these values are not well known *a priori*, it is likely that they will be modified as DynSWS evolves.

AC20. The data structure for storing the geometry of the domain. This is likely to change in multiple ways: the manner in which the data is stored and/or the semantics of the contained data (see **AC4**). However, the interface to the data should remain constant.

AC21. The data structure for storing the geometry of the boundaries. This is likely to change in a similar manner to that described in **AC20**.

AC22. The data structure for storing material properties. As the required material properties will change depending on the material model, the data structure containing these properties is also likely to change. It is also possible that multiple versions of this data structure may exist for different subdomains.

AC23. The data structure for storing vector field data. This includes displacement, velocity, and acceleration.

AC24. The data structure for storing tensor field data. This includes stress, strain, strain rate, and pressure.

AC25. The algorithm for solving the systems of partial differential equations representing each subdomain. It is foreseeable that certain techniques for solving partial differential equations may be more amenable to one type of subdomain than others, or that changing the formulation from small strain to large strain (see **AC1**) may require a different algorithm.

2.2 Unlikely Changes

This section lists changes that are not considered likely to occur. The design of DynSWS will not necessarily ensure that these changes are easy to make. The first set of changes relates to fundamental aspects of the model that DynSWS will implement:

UC1. Time dependency of load input. Since static loading may be considered as a special case of dynamic loading (where the frequency of the loading is zero), there is no reason to modify the code specifically for the case of static loading.

UC2. The types of subdomain that make up the problem domain: structure, fluid, and soil. Although the details of the modelling of these subdomains may change, it is not expected that additional types of subdomain will need to be accomodated.

UC3. The functional goals, which are to compute the displacement, velocity, and acceleration response of the system and to compute the stress and strain fields. These are the basic goals for any model of the response of physical objects to dynamic loading.

UC4. The assumption that the domain is isothermal. Temperature gradients within the problem domain are not expected to have a significant influence on the model.

UC5. The assumption that there are no sources or sinks of material internal to the domain. DynSWS is intended to deal with problems where the materials are either present in the model or entering and exiting from the boundaries. It should not be difficult to construct any soil-water-structure interaction problem to accommodate this assumption.

UC6. The neglect of relativistic effects. It is inconceivable that the materials modelled using DynSWS would approach even a small fraction of the speed of light.

UC7. Continuum mechanics modelling framework. Molecular level interactions are not likely to influence the model.

UC8. The incompressibility of water. For practical purposes, this aspect of the material behaviour of water seems to be the case. This assumption also simplifies the formulation considerably.

UC9. Water is modelled as a Newtonian fluid. Again, this is a well accepted property of the behaviour of water.

UC10. Fluid flow is laminar. The flow velocity of fluids, especially those contained within the pores of two phase regions, is not expected to be elevated to turbulent levels. Regardless, this change would require modifications at the theoretical model level so it does not make sense to accommodate it at the module level.

UC11. Soil grains are incompressible. Although the model will capture changes in the bulk density of two phase regions due to changes in porosity, the change in density of the actual solid particles is not expected to vary greatly.

UC12. Soil behaviour depends on the porosity level. The actual value of porosity at which the soil behaviour transitions from non-Newtonian to Newtonian may change, but the fact that there is a transition should not.

The following is an additional list of unlikely changes that relate to the requirements of the system:

UC13. The system behaviour model is: read input, perform calculation, generate output. It is not expected that users of DynSWS would need to interact with the calculation in real-time. Accommodating this feature would likely add to much complication to the implementation.

UC14. The compiled version should be contained in a single executable file or dynamic library. This assumption makes distribution and operation of the model simpler.

UC15. Interaction with the program will be through input and output files. Given the quantity of input and output data that will be involved in the model, this is likely the only input-output model that makes sense. This also simplifies interaction with pre- and post-processor programs.

3 Module Specification

This section presents the modular decomposition of the DynSWS system. Section 3.1 shows a high-level view of the decomposition in tabular form. Section 3.2 lists each module and provides the secret, the service, and (optionally) the prefix for each of the modules at the lowest level.

3.1 Module Hierarchy

Typical of modular decomposition based on the principle of information hiding are three modules at the highest level: machine hiding, behaviour hiding, and software decision hiding. The machine hiding module involves the interaction between the virtual realm of software and the physical realm of hardware; Table 3.1 shows the machine hiding module decomposition for DynSWS. The behaviour hiding module is concerned with items such as input formatting and text messages; Table 3.2 shows the behaviour hiding module decomposition for DynSWS. The software decision hiding module includes items such as internal data structures and important algorithms; Table 3.3 shows the software decision hiding module decomposition for DynSWS. Note that the services of some of the modules (particularly in the machine hiding module) may not be implemented in DynSWS as they are provided by the programming language or the operating system, but they are listed here nonetheless for completeness as well as awareness of the dependencies of DynSWS on outside systems.

Table 3.1: Decomposition of the machine hiding module of the DynSWS system

Level 1	Level 2	Level 3
Machine Hiding	File Reading and Writing	
	Physical Data Operations	Storage Access
		Integer Operations
		Floating Point Operations

Table 3.2: Decomposition of the behaviour hiding module of the DynSWS system

Level 1	Level 2	Level 3	Level 4
Behaviour Hiding	Master Control		
	Input File Reading	Input File Control	
		Domain File Reader	
		Boundary File Reader	
		Material File Reader	
		Initial Condition Readers	Initial Vector Field Reader
			Initial Tensor Field Reader
		Boundary Condition Readers	Kinematic BC Reader
			Natural BC Reader
		Body Force Reader	
	Output File Writing	Output File Control	
		Vector Field Writer	
		Tensor Field Writer	
	Log Message Handling	Log Message Control	
		Log Messages	

Table 3.3: Decomposition of the software decision hiding module of the DynSWS system

Level 1	Level 2	Level 3	Level 4
Software Decision Hiding	System Constants		
	Data Structures	Domain Geometry Data	
		Boundary Geometry Data	
		Material Property Data	
		Vector Field Data	
		Tensor Field Data	
	PDE Solver		

3.2 Module Decomposition

This section details each of the lowest level modules ("leaf" modules) in the design of DynSWS. In accordance with the design principle of information hiding, each leaf module has one secret and provides one service. The goal is to keep the scope of each leaf module relatively small and self-contained so that each can be viewed as a work assignment. The fact that each module maintains a secret allows different modules to be worked on in parallel, provided that the interface to the module is specified. That is, the implementation details of the module's service are isolated. This type of design also facilitates future changes to the software product as an individual change is ideally isolated to a single leaf module (provided that it comes from the list of anticipated changes in Section 2.1). Finally, some leaf modules are assigned a naming convention prefix to avoid naming conflicts in the implementation.

3.2.1 Machine Hiding

3.2.1.1 File Reading and Writing

Secret: The data structure of files in physical memory.

Service: Access file data from physical memory.

Prefix: N/A

3.2.1.2 Physical Data Operations

3.2.1.2.1 Storage Access

Secret: The data structure of program data on the physical storage media.

Service: Access program data from physical storage.

Prefix: N/A

3.2.1.2.2 Integer Operations

Secret: The algorithms for mathematical operations on integers in hardware.

Service: Perform mathematical operations on integers.

Prefix: N/A

3.2.1.2.3 Floating Point Operations

Secret: The algorithms for mathematical operations on floating point numbers in hardware.

Service: Perform mathematical operations on floating point numbers.

Prefix: N/A

3.2.2 Behaviour Hiding

3.2.2.1 Master Control

Secret: The overall algorithm of the software product.

Service: Manage the sequence of tasks required to meet all requirements.

Prefix: N/A

3.2.2.2 Input File Reading

3.2.2.2.1 Input File Control

Secret: The system of files used for input data.

Service: Put data from input file readers into internal data structures.

Prefix: in_

3.2.2.2.2 Domain File Reader

Secret: The format of the domain geometry data file.

Service: Read domain geometry data from the associated data file.

Prefix: N/A

3.2.2.2.3 Boundary File Reader

Secret: The format of the boundary geometry data file.

Service: Read boundary geometry data from the associated data file.

Prefix: N/A

3.2.2.2.4 Material File Reader

Secret: The format of the material property data file.

Service: Read material property data from the associated data file.

Prefix: N/A

3.2.2.2.5 Initial Condition Readers

3.2.2.2.5.1 Initial Vector Field Reader

Secret: The format of the initial conditions data file for displacement, velocity, and acceleration.

Service: Read initial conditions data for displacement, velocity, and acceleration from the associated data file.

Prefix: N/A

3.2.2.2.5.2 Initial Tensor Field Reader

Secret: The format of the initial conditions data file for stress, strain, strain rate, and pressure.

Service: Read initial conditions data for stress, strain, strain rate, and pressure from the associated data file.

Prefix: N/A

3.2.2.2.6 Boundary Condition Readers

3.2.2.2.6.1 Kinematic BC Reader

Secret: The format of a kinematic boundary conditions data file.

Service: Read kinematic boundary conditions from the associated data file.

Prefix: N/A

3.2.2.2.6.2 Natural BC Reader

Secret: The format of a natural boundary conditions data file.

Service: Read natural boundary conditions from the associated data file.

Prefix: N/A

3.2.2.2.7 Body Force Reader

Secret: The format of the data file for applied body forces.

Service: Read data for applied body forces from the associated data file.

Prefix: N/A

3.2.2.3 Output File Writing

3.2.2.3.1 Output File Control

Secret: The system of files used for output data.

Service: Pass output data from internal data structures onto the appropriate output file writer.

Prefix: out_

3.2.2.3.2 Vector Field Writer

Secret: The format of the output data file for displacement, velocity, and acceleration.

Service: Write output data for displacement, velocity, and acceleration to the associated data file.

Prefix: N/A

3.2.2.3.3 Tensor Field Writer

Secret: The format of the output data file for stress, strain, strain rate, and pressure.

Service: Write output data for stress, strain, strain rate, and pressure to the associated data file.

Prefix: N/A

3.2.2.4 Log Message Handling

3.2.2.4.1 Log Message Control

Secret: The system for reporting progress and error messages.

Service: Log progress and error messages corresponding to activated codes.

Prefix: log_

3.2.2.4.2 Log Messages

Secret: The formatted text messages corresponding to progress and error codes.

Service: Translate from a progress or error code into a human readable text message.

Prefix: msg_

3.2.3 Software Decision Hiding

3.2.3.1 System Constants

Secret: The values of constants used by the system (*i.e.* values that are known at compile time, but are likely to change).

Service: Make the system constants available to other modules that make use of them.

Prefix: CNST_

3.2.3.2 Data Structures

3.2.3.2.1 Domain Geometry Data

Secret: The data structure for storing domain geometry data.

Service: Provide access to domain geometry data through an intuitive interface.

Prefix: dmn_

3.2.3.2.2 Boundary Geometry Data

Secret: The data structure for storing boundary geometry data.

Service: Provide access to boundary geometry data through an intuitive interface.

Prefix: bnd_

3.2.3.2.3 Material Property Data

Secret: The data structure for storing material property data.

Service: Provide access to material property data through an intuitive interface.

Prefix: mtl_

3.2.3.2.4 Vector Field Data

Secret: The data structure for storing displacement, velocity, and acceleration data.

Service: Provide access to displacement, velocity, and acceleration data through an intuitive interface.

Prefix: vec_

3.2.3.2.5 Tensor Field Data

Secret: The data structure for storing stress, strain, strain rate, and pressure data.

Service: Provide access to stress, strain, strain rate, and pressure data through an intuitive interface.

Prefix: tns_

3.2.3.3 PDE Solver

Secret: The algorithm for solving a system of partial differential equations.

Service: Compute the solution to a system of partial differential equations.

Prefix: pde_

3.3 Uses Hierarchy

This section shows how the various modules in DynSWS are interrelated. Figure 3.1 shows the uses hierarchy for the DynSWS system. Note that only leaf modules are shown as these are the only modules that will actually be implemented (or used from an external source). Also, note that while the uses hierarchy implies the control flow of the program, it does not explicitly display the order in which the modules are called; the uses hierarchy simply shows which modules use other modules. From a high level, the system originates in Master Control, which controls the overall algorithm of the software product, and ends in low level operations such as Storage Access and Integer/Floating Point Operations, which are typically provided by the programming language or operating system. It is evident that there are no closed loops in the uses hierarchy of DynSWS, which is important for facilitating the incremental implementation of the software product. It is also clear that the most important modules to the control flow are Master Control, Input File Control, PDE Solver, Output File Control, and Log Message Control; this is in keeping with the typical control flow for scientific computing software.

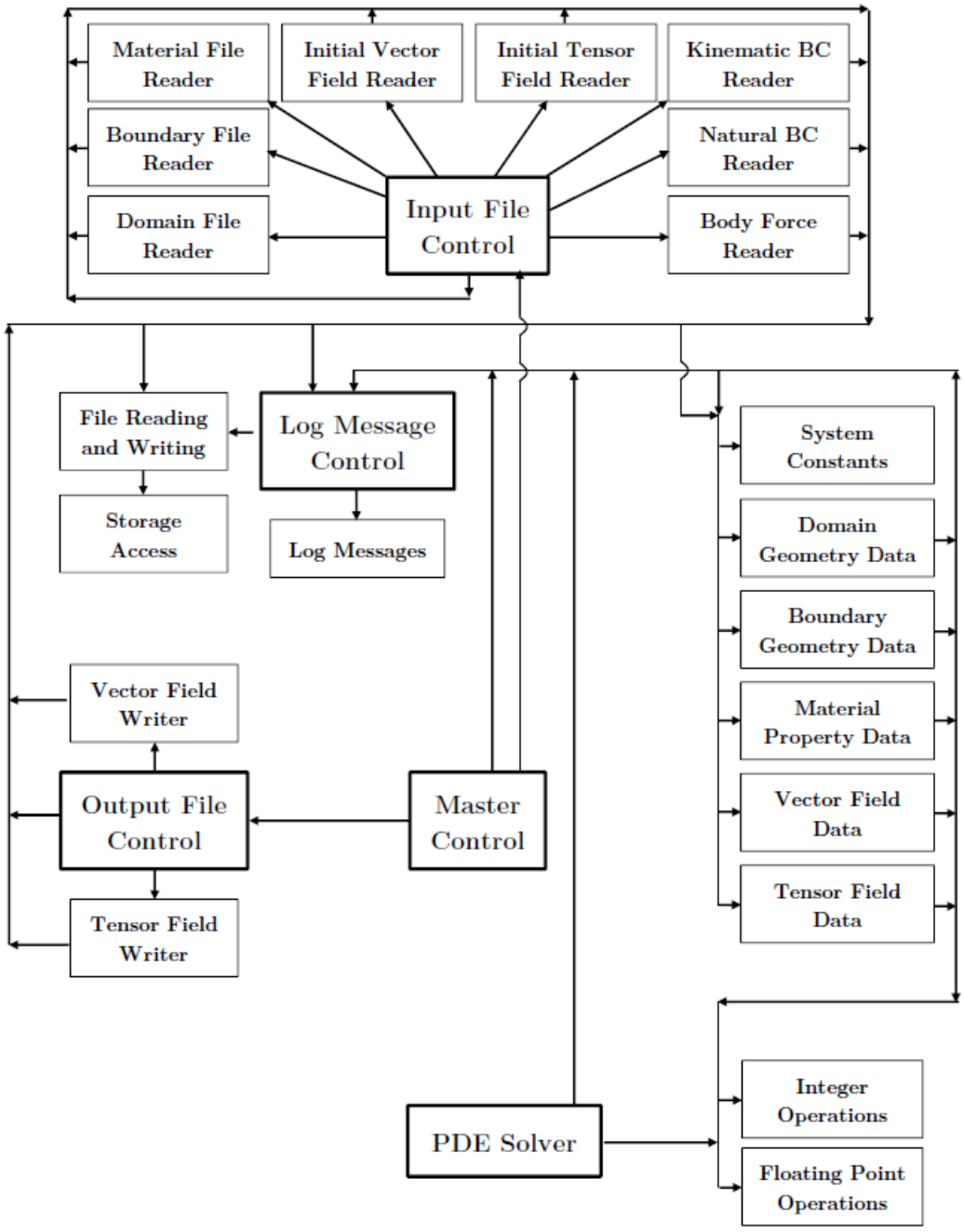

Figure 3.1: Uses hierarchy for modular decomposition of DynSWS

4 Traceability Matrices

Figure 4.1: Traceability matrix for anticipated changes, part 1 of 3

Top-Level Module	Leaf Module	Anticipated Change								
		AC1	**AC2**	**AC3**	**AC4**	**AC5**	**AC6**	**AC7**	**AC8**	**AC9**
	1.1									
1 (Machine Hiding)	1.2.1									
	1.2.2									
	1.2.3									
	2.1									
	2.2.1					X				
	2.2.2						X			
	2.2.3							X		
	2.2.4								X	
	2.2.5.1									X
2 (Behaviour Hiding)	2.2.5.2									
	2.2.6.1									
	2.2.6.2									
	2.2.7									
	2.3.1									
	2.3.2									
	2.3.3									
	2.4.1									
	2.4.2									
	3.1									
	3.2.1									
3 (Software Decision Hiding)	3.2.2									
	3.2.3									
	3.2.4									
	3.2.5									
	3.3	X	X	X	X					

Figure 4.2: Traceability matrix for anticipated changes, part 2 of 3

Top-Level Module	Leaf Module	Anticipated Change							
		AC10	**AC11**	**AC12**	**AC13**	**AC14**	**AC15**	**AC16**	**AC17**
	1.1								
1 (Machine Hiding)	1.2.1								
	1.2.2								
	1.2.3								
	2.1								
	2.2.1								
	2.2.2								
	2.2.3								
	2.2.4								
	2.2.5.1								
2	2.2.5.2	X							
(Behaviour	2.2.6.1		X						
Hiding)	2.2.6.2			X					
	2.2.7				X				
	2.3.1					X			
	2.3.2						X		
	2.3.3							X	
	2.4.1								X
	2.4.2								
	3.1								
	3.2.1								
3 (Software	3.2.2								
Decision	3.2.3								
Hiding)	3.2.4								
	3.2.5								
	3.3								

Figure 4.3: Traceability matrix for anticipated changes, part 3 of 3

Top-Level Module	Leaf Module	Anticipated Change							
		AC18	**AC19**	**AC20**	**AC21**	**AC22**	**AC23**	**AC24**	**AC25**
1 (Machine Hiding)	1.1								
	1.2.1								
	1.2.2								
	1.2.3								
2 (Behaviour Hiding)	2.1								
	2.2.1								
	2.2.2								
	2.2.3								
	2.2.4								
	2.2.5.1								
	2.2.5.2								
	2.2.6.1								
	2.2.6.2								
	2.2.7								
	2.3.1								
	2.3.2								
	2.3.3								
	2.4.1								
	2.4.2	X							
3 (Software Decision Hiding)	3.1		X						
	3.2.1			X					
	3.2.2				X				
	3.2.3					X			
	3.2.4						X		
	3.2.5							X	
	3.3								X

Figure 4.4: Traceability matrix for unlikely changes, part 1 of 2

Top-Level Module	Leaf Module	Unlikely Change							
		UC1	**UC2**	**UC3**	**UC4**	**UC5**	**UC6**	**UC7**	**UC8**
1 (Machine Hiding)	1.1								
	1.2.1								
	1.2.2								
	1.2.3								
2 (Behaviour Hiding)	2.1			X					
	2.2.1			X					
	2.2.2					X	X	X	
	2.2.3						X	X	
	2.2.4		X		X	X	X	X	X
	2.2.5.1			X	X	X	X	X	X
	2.2.5.2			X	X	X	X	X	X
	2.2.6.1	X		X	X		X	X	
	2.2.6.2	X		X	X		X	X	
	2.2.7	X					X	X	
	2.3.1			X					
	2.3.2	X		X	X	X	X	X	X
	2.3.3	X		X	X	X	X	X	X
	2.4.1								
	2.4.2								
3 (Software Decision Hiding)	3.1						X		X
	3.2.1					X		X	
	3.2.2							X	
	3.2.3				X	X	X	X	X
	3.2.4	X		X	X	X	X	X	X
	3.2.5	X		X	X	X	X	X	X
	3.3	X	X	X	X	X	X	X	X

Figure 4.5: Traceability matrix for unlikely changes, part 2 of 2

Top-Level Module	Leaf Module	Unlikely Change						
		UC9	**UC10**	**UC11**	**UC12**	**UC13**	**UC14**	**UC15**
1 (Machine Hiding)	1.1							
	1.2.1							
	1.2.2							
	1.2.3							
2 (Behaviour Hiding)	2.1					X	X	X
	2.2.1							X
	2.2.2							X
	2.2.3							X
	2.2.4	X	X	X	X			X
	2.2.5.1	X		X	X			X
	2.2.5.2	X		X	X			X
	2.2.6.1	X		X	X			X
	2.2.6.2	X		X	X			X
	2.2.7	X		X				X
	2.3.1							X
	2.3.2	X		X	X			X
	2.3.3	X		X	X			X
	2.4.1							X
	2.4.2							
3 (Software Decision Hiding)	3.1							
	3.2.1							
	3.2.2							
	3.2.3	X	X	X	X			
	3.2.4	X		X	X			
	3.2.5	X		X	X			
	3.3	X	X	X	X			

Figure 4.6: Traceability matrix for goal statements

Top-Level Module	Leaf Module	Goal	
		G1	**G2**
1 (Machine Hiding)	1.1		
	1.2.1		
	1.2.2		
	1.2.3		
2 (Behaviour Hiding)	2.1		
	2.2.1		
	2.2.2		
	2.2.3		
	2.2.4		
	2.2.5.1		
	2.2.5.2		
	2.2.6.1		
	2.2.6.2		
	2.2.7		
	2.3.1		
	2.3.2		
	2.3.3		
	2.4.1		
	2.4.2		
3 (Software Decision Hiding)	3.1		
	3.2.1		
	3.2.2		
	3.2.3		
	3.2.4		
	3.2.5		
	3.3	X	X

Figure 4.7: Traceability matrix for non-functional requirements, part 1 of 2

Top-Level Module	Leaf Module	Non-Functional Requirement						
		NFR1	**NFR2**	**NFR3**	**NFR4**	**NFR5**	**NFR6**	**NFR7**
1 (Machine Hiding)	1.1							
	1.2.1							
	1.2.2							
	1.2.3							
2 (Behaviour Hiding)	2.1							X
	2.2.1							
	2.2.2							
	2.2.3							
	2.2.4							
	2.2.5.1							
	2.2.5.2							
	2.2.6.1							
	2.2.6.2							
	2.2.7							
	2.3.1							
	2.3.2							
	2.3.3							
	2.4.1							
	2.4.2							
3 (Software Decision Hiding)	3.1							
	3.2.1							
	3.2.2							
	3.2.3							
	3.2.4							
	3.2.5							
	3.3		X	X	X	X	X	

Figure 4.8: Traceability matrix for non-functional requirements, part 2 of 2

Top-Level Module	Leaf Module	Non-Functional Requirement						
		NFR8	**NFR9**	**NFR10**	**NFR11**	**NFR12**	**NFR13**	**NFR14**
1 (Machine Hiding)	1.1							
	1.2.1							
	1.2.2							
	1.2.3							
2 (Behaviour Hiding)	2.1	X						X
	2.2.1	X			X			
	2.2.2			X	X		X	
	2.2.3			X	X		X	
	2.2.4			X	X		X	
	2.2.5.1			X	X		X	
	2.2.5.2			X	X		X	
	2.2.6.1			X	X		X	
	2.2.6.2			X	X		X	
	2.2.7			X	X		X	
	2.3.1	X			X			
	2.3.2			X	X		X	
	2.3.3			X	X		X	
	2.4.1				X		X	
	2.4.2				X			
3 (Software Decision Hiding)	3.1							
	3.2.1			X		X		
	3.2.2			X		X		
	3.2.3			X		X		
	3.2.4			X		X		
	3.2.5			X		X		
	3.3		X	X				

References

[1] B. Karchewski, "Software requirements specification for two and three dimensional dynamic model of soil-water-structure interaction," Course Project - CES 741, McMaster University, Feb. 2012.

[2] D. Parnas, "On the criteria to be used in decomposing systems into modules," *Communications of the ACM*, vol. 15, no. 12, Dec.

[3] ——, *Software Fundamentals: Collected Papers by David L. Parnas*, D. Hoffman and D. Weiss, Eds. Toronto, ON: Addison-Wesley, 2001.