The CRANE Framework for Simulation Model Workflows

by

Daniel Princz

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Civil Engineering

Waterloo, Ontario, Canada, 2016

AUTHOR'S DECLARATION

Abstract

CRANE is presented as a flexible framework for linking simulation models and model support tools to form integrated modelling systems for engineering and scientific applications, evaluated using the scientific workflow approach. CRANE was written using an object-oriented programming language; the separation of its core processing component from its user interface; support for plugins that can be updated and enhanced independent of the framework; and with intuitive user-friendly features and human-readable configuration files. Its strength is its ability to connect to legacy simulation models, whose code cannot be modified, through structured and/or free-format text files. The framework contains an engine that interprets the requirements of simulation models and modelling support tools, and facilitates the flow of data between these components in a simulation workflow. In addition, a user interface provides a familiar graphic interface through which the engine can be configured and monitored during the evaluation of the simulation workflow.

A case study was undertaken to demonstrate the ability of CRANE to wrap around, configure, and evaluate two versions of a hydrologic simulation model. Using the default parameter configuration, both versions of the model failed to capture the hydrologic regime of the basin; the modified version of the model only marginally improved the results by redistributing excess meltwater in a presumably more physically based way. The modified version of the model allowed excess meltwater to contribute to ponded storage and infiltrate into soil. By contrast, the original version of the model increased the evaporation rate to account for the excess meltwater. Given the poor overall performance of the model in this particular modelling scenario, the contribution of the modification could not be definitively commented upon. It was concluded that further assessment would be improved by better parameterization of the model. CRANE was used to configure the input files for the model, as well as to execute a simple simulation workflow. Unfortunately, the relative simplicity of the case study did not highlight the more advanced features of the framework. As this is a preliminary introduction of the framework, additional and different types of case studies are recommended, the results from which would identify areas where the framework can continue to be developed and enhanced.

Acknowledgements

The author expresses sincere gratitude and thanks to Dr. E. D. Soulis, Dr. J. R. Craig, Dr. L. S. Matott, and B. Davison, the examining committee, and faculty and students of the Department of Civil and Environmental Engineering, with special acknowledgement of V. Tolton. In addition, the author gratefully acknowledges the continuous support of his family and friends throughout this endeavour.

The research presented herein was funded in part by the Research Affiliate Program of Environment Canada; B. Davison and A. Pietroniro are thanked for their encouragement in this capacity.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

Simulation modelling is an important component of environmental risk assessment, as well as in supporting the determination of environmental policies (Babendreier & Castleton 2005). Environmental simulation models help represent, understand, and explore natural and engineered systems (Argent 2004). Simulation models sometimes facilitate forecasting and prediction, and are increasingly found in integrated modelling systems (Kouwen 1988; Pietroniro et al. 2007; Soulis et al. 2000; Voinov & Cerco 2010).

An integrated modelling system contains a number of individual models, modules, or other components that are linked together and operated in a collective way (Altintas et al. 2004; Argent 2004; Berkley et al. 2005). The components of an integrated modelling system might include tools to prepare, process, and visualize data; perform such tasks as a calibration on parameter values; or perform sensitivity, uncertainty, or other statistical analyses on the results from a simulation model. In some instances, the combination of these components might provide a more robust analysis in representing a complex environmental or scientific system, than could be obtained from a single simulation model alone (Matott, Babendreier & Purucker 2009).

A number of frameworks exist to link scientific process modules to form simulation models (e.g., Gregersen, Gijsbers & Westen 2007; Hillyer et al. 2003; Banta et al. 2006; Branger et al. 2010; and Hinkel 2005). However, not as many frameworks exist to link simulation models with other simulation models or modelling tools, such as utilities for calibration or data translation and/or visualization (Babendreier & Castleton 2005; Matott, Babendreier & Purucker 2009; Whelan et al. 1997). A few limitations exist in the frameworks that exist for this purpose, which are discussed in this thesis. In brief, many of them require modifications to the scientific programming code of the model, cannot arbitrarily link to a model by its input or output files, and/or lack intuitive user-friendly features. Perhaps the inability to link to a scientific model, unless by modifying its code, is the most limiting of these shortcomings as the code of some scientific models is considered to be of a proven quality (Whelan et al. 1997), which would be compromised by the invasive approach.

1.2 Objectives

The primary objective of this thesis is to present an intuitive software framework, CRANE, for building integrated modelling systems that does not require the modification of programming code to interact with existing scientific models. Its development is consistent with recent efforts in the engineering and scientific communities to produce modelling frameworks that promote interoperability between otherwise independently developed simulation models and modelling tools. Examples of these efforts include Altintas et al. (2004), Babendreier & Castleton (2005), Branger et al. (2010), Hinkel (2005), Oinn et al. (2004), and Whelan et al. (1997).

Differing from some of these efforts, CRANE demonstrates the ability to configure a simulation model by interacting with its input and output files using a unique abstraction approach. Among the requirements for the development of the framework are the ability to interact with the input and output files of the model; be applicable to more than a single case study, simulation model, or modelling scenario; use a relatively portable programming language; and be flexible in allowing the development of new features over time.

1.3 Thesis Organization

This thesis contains five chapters:

- an introduction to the thesis, the objectives of developing CRANE, and the organization of this thesis (Chapter 1);
- an overview of simulation modelling, inverse modelling, and integrated modelling systems, and a literature review of other frameworks that are similar to CRANE in either function or purpose (Chapter 2);
- design requirements and the programming interfaces of CRANE, as well as its constraints and limitations (Chapter 3);
- a case study of CRANE, which demonstrates its ability to wrap around a hydrologic simulation model (Chapter 4); and
- conclusions and a discussion of CRANE (Chapter 5).

Chapter 2
Literature Review

2.1 Parameterization of Simulation Models and Inverse Modelling

Civil engineers, geographers, earth scientists, geologists, meteorologists, and economists use simulation models to study the environment, climate change, sustainable development, human impacts, flood forecasting, or engineering design flows (Dehotin & Braud 2008; Matott, Babendreier & Purucker 2009). In general, simulation models use a system of mathematical equations to represent or conceptualize a natural or engineered system (Dingman 2008, p. 25; Villa, Athanasiadis & Rizzoli 2009). For example, a hydrologic model may contain a number of equations to represent hydrologic processes to predict river channel flows in response to a precipitation event.

In a very generic sense, simulation models use input data to produce output data. A specific type of input data, called parameters, allow a model to adapt to and be representative of more than one simulation scenario. For example, a parameter for hydraulic conductivity (K_h) is needed to calculate the flow of water through different types of soil columns without modifying the general physical description of the process (Dingman 2008, p. 231); variations between soil types are reflected by variations in K_h.

Often times, a precise value of the parameter that might be most representative of the physical system is unknown, especially if the parameter is tied to a hard-to-measure element in a natural system (Matott, Babendreier & Purucker 2009). Examples of such parameters include the porosity and hydraulic conductivity of a heterogeneous soil (Kouwen 1998). In this case, techniques such as inverse modelling can identify an appropriate value (McLaughlin & Townley 1996).

Inverse modelling adjusts the values of the parameters until a finite number of iterations has completed or until the output from the model satisfactorily matches historical (i.e., measured) data. Typically, the model is considered a sufficient representation of reality when the output from the model and the measured data are in agreement. Trial-and-error is a simple way of adjusting the parameters. Alternatively, algorithms, such as those proposed by Beven & Binley (1992) or Tolson & Shoemaker (2007), can be used to automatically traverse a set of possible parameter values in a more systematic way.

In the case of a parameter calibration by trial-and-error, a visual comparison of the output from the model to the measured data might be sufficient to assess the performance of the model. By contrast, in an automated case, a statistical metric of model quality, such as the Nash-Sutcliffe coefficient (Dingman 2008, p. 580) or the sum of squared errors, can guide the process of calibration. The Nash-Sutcliffe coefficient compares simulated to measured streamflow values and is often an appropriate metric for the inverse modelling of a hydrologic model (Dingman 2008).

Several utilities have been developed to automate the process of inverse modelling, such as implementations of the Generalized Likelihood Uncertainty Estimation (GLUE) method (Beven & Binley 1992), UCODE (Poeter & Hill 1998; Banta et al. 2008), PEST (Doherty 2005), or OSTRICH (Matott 2005). Some of these utilities, such as OSTRICH, offer the choice of various algorithms to traverse parameter space, multi-objective parameterization, the choice of various statistical metrics, and additional analyses to measure the sensitivity of the parameters to calibration and/or uncertainty in the parameter values. Matott et al. (2009) published an overview of a number of these utilities.

These types of utilities, as well as tools for additional statistical analyses, data assimilation, data translation, and visualization, provide additional insight into results from the modelling process. Their combination into an integrated modelling system can enhance the utility of the information provided by a simulation model. This is particularly useful when it is important to quantify the uncertainty in results from the model, for objectives such as advising environmental policy (Babendreier & Castleton 2005; Matott, Babendreier & Purucker 2009).

2.2 Integrated Modelling Systems as Scientific Workflows

While the previous section focused on simulation modelling and individual components of an integrated modelling system, this section focuses on a methodology for linking these components into an integrated system. An integrated modelling system is composed of a number of individual components, which could include one or more simulation models or modelling tools for pre- and post-simulation data manipulation, translation, or another analysis.

In many instances, scientists and engineers use batch files or shell scripts to automate their modelling exercises and to process, transform, and analyze scientific data (Görlach et al. 2011; McPhillips et al. 2009). For example, a shell script to run an inverse modelling exercise might contain procedural commands to copy a number of pre-fabricated files containing parameter values, and execute a sequence of simulations, each of which is evaluated using one of those files as input data.

The fabrication and execution of the commands in a shell script are similar to the design and execution of a scientific workflow. The scientific workflow approach differentiates from imperative scripts (e.g., shell scripts) in that the workflow is driven by the availability of data and not only by procedural commands (Berkley et al. 2005; Ludäscher et al. 2006; McPhillips et al. 2009). For example, a shell script might continue to execute a sequence of tasks after the operation of one of those tasks has failed, whereas a properly implemented workflow will stop if data are unavailable from the failed task.

The scientific workflow approach is a method for orchestrating the evaluation of a scientific experiment, in which the execution of the current task only moves forward if the required data are available from prior tasks (Berkley et al. 2005; Görlach et al. 2011). A task is any process that transforms or analyzes data (Simmhan et al. 2009).

Scientific workflows also act as metadata to describe and archive scientific experiments, including modelling scenarios. Many shell scripts and batch files contain commands meant for a particular operating system. By contrast, a workflow written in a portable programming code, for example, is a transparent set of instructions that can be transferred between users and systems. One of the advantages of preserving workflows as metadata is that an exact configuration of a simulation model is replicable, re-evaluable, and reusable for other analyses.

2.3 Frameworks for building Integrated Modelling Systems

While the previous section introduced integrated modelling systems and the scientific workflow approach, this section focuses on frameworks that exist for building integrated modelling systems. Intergovernment agencies have begun to promote interoperability between modelling systems since more flexible and powerful computing has become available (Whelan et al. 1997).

Frameworks for building integrated modelling systems facilitate the transfer of data between linked tasks. They differ from programming interfaces for building simulation models in that they transfer data in between tasks, rather than continuously during the simulation. Specifically, where a simulation model might transfer data in a sub-hourly time-step, these types of frameworks may only transfer data at the end of a particular task, after an entire set of data have become available.

Because they operate at the task level, these types of frameworks can more loosely couple with simulation models than some programming interfaces, which can result in certain benefits. For example, a loosely coupled framework will generally not require the translation of data to alternate or proprietary data types between model time-steps, whereas a tightly coupled framework may require the translation of these data to an alternate format. ‘Casting’ refers to this type of translation or transfer of data, and it can add significant overhead (i.e., additional computational expense) to the simulation model (Lloyd et al. 2011). In addition, a loosely coupled framework may not require any modification to the programming code of the simulation model, and may instead interact with the model by its input and output files. This is particularly beneficial when interacting with older scientific models that have an established history, as modifications would compromise the legacy of their code (Whelan et al. 1997).

A summary of frameworks for building integrated modelling systems is presented in Table 1. They include Kepler (Altintas et al. 2004), COMAD (McPhillips et al. 2009), Freefluo and Taverna (Oinn et al. 2004; Oinn et al. 2006), Object Modeling System (David et al. 2010; Kralisch, Krause & David 2005), Object User Interface (Markstrom & Koczot 2008), HydroPlatform (Harou et al. 2010), HydroDesktop (Ames et al. 2012), and Hydrologists’ Workbench (Box 2010; Cuddy & Fitch 2010). The scientific workflow approach is driven by the availability of data, but an integrated modelling system does not necessarily have to execute tasks using this approach. The frameworks presented in Table 1 support this approach either by data management or within the modules of simulation models within the integrated system.

Table 1: List of frameworks for building integrated modelling systems.

NAME	PLATFORM	CAPABILITIES†					AVAILABILITY‡	
		DM	MD	SA+	GUI	BM	DISTRIBUTION	SOURCE CODE
Kepler	Java	X	X	V	X	X	X	X
Taverna	Java	X	X		X	X	X	X
HydroPlatform	Python	X			X		X	X
Hydrologists' Workbench	C#	X	X	X	X	X	A	A
Object Modeling System	Java		X	X	X		X	X
Object User Interface	Java	X	X		X		X	X
HydroDesktop	C#	X			X		X	X

†DM: data management; MD: modular development; SA: statistical analyses; GUI: graphical user interface or experience; BM: batch mode.
‡X: available; A: availability noted, but links have expired or are no longer publicly accessible.
+X: analyses; V: analyses and reporting tools (e.g., visualization).

The subsections that follow discuss the strengths and weaknesses of the frameworks presented in Table 1. Namely, each subsection reviews specific considerations: (1) data management; (2) integration with simulation models; (3) operating platform and programming language; and (4) available extensions and plugins. The final subsection provides a summary of these considerations.

2.3.1 Facilitating Data Management

Data management is a critical component of integrated simulation modelling. It might include data assimilation, filtering, analysis, or translation, as well as visualization or report building. Some of the frameworks provide the ability to interact with existing data services.

Kepler is a framework that has the ability to ingest and interpret both local and online data services using multiple languages, including XQuery, Ecological Metadata Language (EML), and Perl. The flexibility of Kepler is demonstrated in its use as part of the Science Environment for Ecological

Knowledge (SEEK) initiative (Michener et al. 2005), which ties it to the EcoGrid cyber-infrastructure of ecological data sources.

In a similar manner, Taverna (Oinn et al. 2004) links to a number of remote data services in the bioinformatics community, and allows scientists to save and share workflows with others in the scientific community in a proprietary, yet standardized format.

HydroDesktop exploits the Hydrologic Information System (HIS) (Ames et al. 2012) of the Consortium of Universities for the Advancement of Hydrologic Science Inc. (CUAHSI) to provide a large collection of hydrologic observation data to promote hydrologic modelling exercises. The collections offered by CUAHSI mostly contain data for the United States. For example, the National Water Information System (NWIS) from the United States Geological Survey (USGS) or the STORage and RETrieval (STORET) system from the U.S. Environmental Protection Agency (USEPA). However, the infrastructure of the system potentially allows the framework to include any number of international resources.

More akin to Kepler, HydroPlatform is able to ingest input data from local sources and transform them to a format that is compatible with its modelling components.

2.3.2 Integrating with Simulation Models

In addition to linking to a variety of data sources, many of the frameworks have the ability to link to a variety of types of simulation models. The Freefluo workflow enactment engine is the workflow orchestration tool used by Taverna, and allows any model to be built from data services that implement the Taverna interface or that are built using the “Simple conceptual unified flow language” (Scrufl). However, this type of model is largely limited to creating queries to retrieve data from online ontology services (i.e., not environmental models).

Kepler, like Taverna, allows integration with any model that conforms to its programming interface. Additionally, Kepler provides the ability to build different types of workflows, which can include nested, looped, and conditional branches that capture the intent, and perhaps a better likeness, of the conceptual model of a desired workflow structure (Ludäscher et al. 2006; McPhillips et al. 2009).

Both Object Modeling System (OMS) and Object User Interface (OUI) integrate with the Modular Modeling System (MMS) of Leavesley et al. (1996) to link to legacy modules to form simulation models. Additionally, OMS facilitates linking these models to other resource management tools to help address policy and management issues. However, they can only link to models that implement the programming interfaces of MMS. HydroPlatform links to any external model whose components can be split into links (e.g., river channels) and nodes (e.g., where two river channels meet). However, it is unclear if this extends to existing models, which may operate using links and nodes, but which cannot be separated into explicit components of links and nodes.

Similarly, Hydrologists' Workbench allows users to import existing models using scripts, or models that conform to the programming interfaces of TIME (Rahman et al. 2003). It can also interpret the configuration files of MODFLOW (Harbaugh 2005).

HydroPlatform and Hydrologists' Workbench are different from Taverna, Kepler, OMS, and OUI, as they do not provide or require programming interfaces to interact with a simulation model. In the case of HydroPlatform, this is by design to promote the independent developments of both the modelling task and framework (Harou et al. 2010). Although not restrictive, an implication to this approach is that the model must be executable from a batch file or similar type of script.

2.3.3 Platform and Programming Languages

The ability of the framework to link to both simulation models and to tools that help with simulation modelling relies, in part, on the programming languages supported by the framework. Taverna and Kepler use the Java programming language, but Kepler also supports other languages, such as Matlab and Python, through an extension to its interface. Similarly, OMS and OUI use Java. However, OMS and OUI can also execute native code (e.g., C, C++) by utilizing the Java Native Interface (JNI). Additionally, OUI supports the eXtensible Markup Language (XML).

HydroPlatform uses the Python programming language. Like OMS and OUI, it has mechanisms by which it can execute native code. HydroDesktop uses C# and supports modules created in any one of the .NET programming languages (e.g., Visual Basic .NET, C#). The capabilities of HydroDesktop are further enhanced by the HydroModeler plugin (Castronova, Goodall & Ercan 2013), which allows it to link to modules that implement the OpenMI (Gregersen, Gijsbers & Westen 2007) programming interface in C#.

2.3.4 Extensions and Plugins

The framework can make use of extensions and plugins to add new features or to enhance functionality. A benefit of using an extension or plugin is that they provide features without requiring significant, or any, new programming code. By contrast, a risk with this approach is the dependency of the framework on third party software. If there is a significant dependency, then this can complicate the development of the framework or jeopardize its longevity, particularly if the third party software is no longer compatible or available (i.e., obsolete). The importance of evaluating available extensions and plugins lies in the capability and capacity of the framework to accommodate changes or updates to the plugins, and vice versa.

Kepler uses the Vergil graphic user interface (GUI) of Ptolemy II (Brooks et al. 2007) to construct and orchestrate the execution of workflows. OMS uses the NetBeans software to provide a GUI. HydroPlatform relies on a considerable amount of open-source (i.e., freely and publicly accessible) code to support many of its capabilities, including the management and visualization of data. HydroDesktop uses the DotSpatial open-source GIS GUI. Hydrologists' Workbench uses the Trident software to provide a GUI, and to construct, visualize, and orchestrate the evaluation of workflows, as well as to provide collaborative and parallel computing capabilities.

In the cases of HydroDesktop and Kepler, the same group develops both the framework and the software that provides their GUIs (e.g., DotSpatial and Ptolemy II, respectively). In the cases of OMS, Hydrologists' Workbench, and HydroPlatform, a third party group develops some of this software.

Both HydroPlatform and Hydrologists' Workbench use plugins that provide core capabilities. In the former case, this includes methods to access, manage, and visualize data. In the latter case, this includes methods to construct, manage, and execute simulation workflows. In both cases, the plugins are open source software (i.e., their programming code is publicly accessible). However, in the case of Hydrologists' Workbench, specifically, development of the plugin has ceased in the public domain. In this case, it is unknown if the most recent version of the plugin works with the framework or if the developers of the framework maintain a modified version of its code.

2.3.5 Summary

A number of frameworks exist for building integrated modelling systems. HydroPlatform is unique in that it contains no programming interfaces to develop new simulation models. By design, it will only link to fully formed simulation models, by utilizing such means as batch files or shell scripts. HydroDesktop and Hydrologists' Workbench provide this capability through extensions to their code.

Kepler and Taverna are both sophisticated and can create and publish workflows to online services. However, at least in the case of Taverna, this use has been limited to the discipline of bioinformatics for ontological models.

OMS and OUI both provide support for MMS, a legacy programming interface for developing modular simulation models. OUI, in particular, also provides a robust user experience, which includes a GIS. However, these frameworks cannot integrate with models that do not implement MMS.

Kepler, OMS and OUI are all Java-based and require the installation of additional software to run on the Microsoft Windows operating system. Moreover, although they do support the execution of native code and fully formed simulation models, modifications are required to the programming code of the simulation models to work with these frameworks.

Hydrologists' Workbench is different in this regard, as it is not invasive and uses an intuitive interface, instead of code, to link to fully formed models. However, it seems that newer versions of the framework are no longer publicly accessible. Furthermore, the framework requires the installation of the Trident software, which is no longer under active development, at least not in the public domain.

All of these frameworks are subject to the dependencies that are inherent to the use of modern programming languages, such as to a particular operating platform, which may limit the distribution of the framework to a handful of operating systems. There is a benefit to developing a cross-platform solution. OpenMI, for example, has concurrent code in both Java and C#, and supports both Linux and Microsoft Windows operating systems (Gregersen, Gijsbers & Westen 2007). However, OpenMI is a programming interface for building simulation models and does not support building integrated modelling systems.

CRANE, a new framework for building integrated modelling systems, is introduced in the next Chapter. Similar to Kepler, Taverna, and HydroDesktop, it provides interfaces to connect with data sources, which includes Microsoft Access and Excel files, SQLite databases, and data stored in text files. Like HydroPlatform, it can connect to fully formed models via their input and output files; and similar to Kepler, Taverna, OMS, and OUI, it can also connect to models directly through their code if implemented in one of the .NET programming languages, which includes Visual Basic .NET and C#.

In addition, like HydroDesktop, CRANE provides programming interfaces to build new extensions and plugins for the framework. However, this extends beyond analytical applications to logistical and management applications that, for example, allow CRANE to interact with the input and output files of a simulation model or modelling tool without being invasive to its programming code.

Finally, and similar to Kepler, the core functionality of CRANE is separated from advanced user-friendly features, such as a GUI, to improve portability and to minimize the dependencies that could limit the longevity of the framework. In addition, like OUI, CRANE also uses transparent and human-readable XML files for its configuration. It further extends upon this flexibility by using a proper XML schema definition that, with an XML editor, can automatically formulate the required structure and elements of the file with minimal effort on the part of the end-user.

Chapter 3
The CRANE Framework

In the context of environmental simulation modelling, CRANE is a framework for building integrated modelling systems to help mitigate environmental issues, facilitate the design of new policies and regulations, and help fulfill other management criteria. For example, CRANE might provide the means to link an environmental simulation model to an economic model to determine best management practices for agriculture.

An objective of developing this framework was to provide a software with the ability to link to an arbitrary model or set of models through input and output files. This link then allows the model to link to other simulation models or modelling tools via the framework. These types of links can facilitate such modelling tasks as ensemble forecasting, parameter calibration, and/or pre- and post-simulation data analysis. The scope of this chapter is to present the mechanisms by which CRANE links to simulation models. Programming requirements and concepts are also discussed.

3.1 Programming Requirements for CRANE

Programming requirements were determined prior to developing CRANE. Based on existing frameworks, outlined in Chapter 2, and the example of Whelan et al. (1997), the requirements included:

1. use of the Object-Oriented Programming (OOP) technique;
2. separation of the framework from modelling code;
3. a sustainable programming approach that provides flexibility to accommodate future updates without compromising existing configurations of the framework or its programming code; and
4. user-friendly design.

The following subsections explain these requirements in detail.

3.1.1 Object-Oriented Programming

Object-oriented programming (OOP) is a type of computer programming that defines programming code as a series of interconnected objects. Each of these objects can contain information or perform a particular task.

A key concept of this style of programming is the ability to define objects as abstract structures, which do not contain any programming code. These structures contain a list of the attributes (i.e., properties) that the object contains, and a list of methods (i.e., actions) that the object can perform. These types of structures are referred to as programming ‘interfaces’.

Similar to programming interfaces are ‘classes’, which can ‘implement’ an interface to inherit its lists of attributes and methods. Unlike interfaces, classes can contain operable code. Not only can a class implement an interface, but it can also implement and extend upon the attributes and methods from another class. This ability for the code to inherit information and tasks from other code is referred to as ‘subsumption’ (Villa, Athanasiadis & Rizzoli 2009), which is a powerful attribute of OOP. Subsumption allows for complex hierarchies to be made from simpler objects, and is a way of organizing, and sometimes isolating, programming code. In the context of an integrated modelling system, these techniques allow for structures to be made of code that correspond well with the structures found in conceptual models (Hillyer et al. 2003).

3.1.2 Separation of the Framework from Modelling Code

When the framework is required to interact with a simulation model or modelling tool, it can do so by integrating and directly interacting with the modelling code. A benefit to this approach is that it allows the framework to bypass reading and writing input and output files, the format and structure of which can change between succeeding versions of a simulation model. However, this type of integration must be necessarily invasive, as it requires code from the framework to be inserted to the simulation model, or vice versa.

For the case when the code of the simulation model cannot or should not be modified, an alternate means of communicating data between the simulation model and framework is via input and output files. This approach requires no modification of existing programming code. A benefit to this approach is that even models with very strict licensing terms, or whose code are completely removed from the public domain, can interact with the framework.

3.1.3 Sustainable Code Development

Introducing a level of separation between simulation models or modelling tools and the framework permits the framework to evolve and adapt as new types of tasks are determined. However, updates or enhancements to the framework should not break compatibility with existing code. Creating a sustainable framework not only means that it has the capacity to evolve and update, but also that the structure of its current core is generic enough to link to a broad scope of potentially yet-to-be-developed tasks (Whelan et al. 1997).

Another sustainable coding tactic pertains to the separation of the core code of the framework from what might be considered non-essential (i.e., accessory) code. For example, while a user interface (UI) is a familiar graphic means through which a user can interact with the framework, it is not necessarily required to evaluate the simulation workflow. A UI can also increase the computational requirements and memory used by the framework. The separation of the core code from non-essential code results in a framework comprised of at least two components: a primary component to determine the data requirements and evaluate the tasks of the workflow, and a secondary component to enhance the capabilities or user-friendliness of the framework. The primary component is self-reliant, while the secondary component is akin to an extension and relies on the primary component.

A benefit of this approach is that the primary component of the framework does not have to include libraries of programming code that might be required by the UI to provide user-interactive controls or to visualize data. The use of these libraries can limit the portability of the code. As a result, the primary component uses a generic or limited subset of the programming language, which allows it to transfer more easily to other operating systems.

3.1.4 User-friendly Design

The separation of core code from the UI in the framework potentially improves its longevity and portability. However, there are users who are accustomed to, prefer, or require a UI to interact with the framework. Some researchers and industry experts are less likely to use even the most state-of-the-art simulation model or analytical tool if it does not include user-friendly features and/or an intuitive UI (Harou et al. 2010).

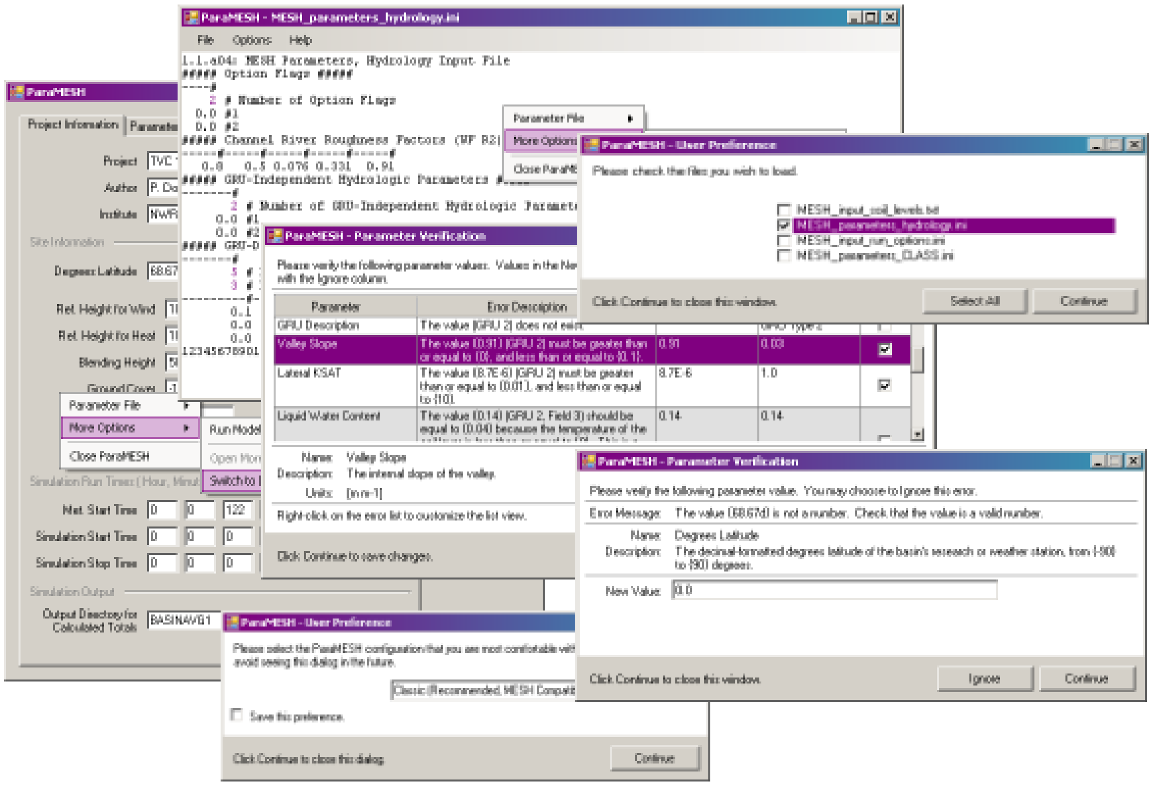
If the UI is not available or is incompatible with a particular system, then there are other ways that a framework can be user-friendly. For example, using structured, human-readable configuration files that adopt a schema which can validate the file and help prevent typographic errors; and by providing informative updates, status messages, or a log, during the execution of critical events or in the evaluation of the simulation workflow.

The next section elaborates on the programming concepts reviewed above, but specific to the development of CRANE. For example, the use of the C# programming language satisfies the OOP constraint; programming interfaces, which allow the creation of plugins, separate the framework from modelling code; the framework is split into two components, comprising an engine and user-interface (UI), for portability and to allow future updates; and structured, human-readable configuration files and an optional UI are used to provide user-friendly features. The next section also presents how CRANE is able to interact with simulation models and modelling support tools, as tasks in a simulation workflow.

3.2 Technical Components of CRANE

The development of the CRANE framework evolved from the same concepts and principals used in another software, ParaMESH (Princz & Matott 2010), but not the same code base. Its development was designed to satisfy the programming requirements from Section 3.1 (e.g., object-oriented programming, separation of the framework from modelling code, a sustainable programming approach, and user-friendly design) and also to improve upon existing frameworks for building integrated modelling systems (refer to Table 1, Section 2.3). Many of the features of CRANE focus on first-time, novice, and inexperienced users, particularly in the configuration of older text-based simulation models. This was also the basis for the ParaMESH software.

ParaMESH, shown in Figure 1, was developed for teaching environments, in which a requirement to configure complex simulation models existed. The configuration that is sometimes required by some models can overshadow and distract a novice user from understanding the mechanics of the modelled process. The main contribution of ParaMESH was its rich UI that provided a graphic interface for console-based models. The interface provided context sensitive help, which displayed ‘on-the-fly’ information, including a description and units of parameter values. It also provided an inverse modelling solution through its integration with third-party utilities for the purpose of calibration (Princz 2009), such as the OSTRICH optimization toolkit (Matott 2005).



```
1.1.a04: MESH Parameters, Hydrology Input File
##### Option Flags #####
----#
    2 # Number of Option Flags
  0.0 #1
  0.0 #2
##### Channel River Roughness Factors (WF R2)
-----#-----#-----#-----#-----#
  0.0   0.5 0.076 0.331  0.91
##### GRU-Independent Hydrologic Parameters #
--------#
       2 # Number of GRU-Independent Hydrologic Paramet
    0.0 #1
    0.0 #2
##### GRU-D
--------#
       5 #
       3 #
-----------#-
      0.1
      0.0
      0.0
12345678901
```

Project: TVC
Author: P. Do
Institute: NWR
Site Information
Degrees Latitude: 68.67
Ref. Height for Wind: 1
Ref. Height for Heat: 1
Blending Height: 5
Ground Cover: -1
Simulation Run Times (Hour, Minut
Met. Start Time: 0 | 0 | 122
Simulation Start Time: 0 | 0 | 0
Simulation Stop Time: 0 | 0 | 0
Simulation Output
Output Directory for Calculated Totals: BASINAVG1

Please check the files you wish to load.

[ ] MESH_input_soil_levels.txt
[✓] MESH_parameters_hydrology.ini
[ ] MESH_input_run_options.ini
[ ] MESH_parameters_CLASS.ini

Click Continue to close this window.

Please verify the following parameter values. Values in the Ne
with the Ignore column.

Parameter	Error Description			
GRU Description	The value [GRU 2] does not exist		GRU Type 2	
Valley Slope	The value (0.91) [GRU 2] must be greater than or equal to [0], and less than or equal to (0.1).	0.91	0.03	✓
Lateral KSAT	The value (8.7E-6) [GRU 2] must be greater than or equal to (0.01), and less than or equal to (10).	8.7E-6	1.0	✓
Liquid Water Content	The value (0.14) [GRU 2, Field 3] should be equal to (0.04) because the temperature of the	0.14	0.14	

Name: Valley Slope
Description: The internal slope of the valley.
Units: [m m-1]

Right-click on the error list to customize the list view.

Click Continue to save changes.

Please verify the following parameter value. You may choose to Ignore this error.

Error Message: The value (68.67d) is not a number. Check that the value is a valid number.

Name: Degrees Latitude
Description: The decimal-formatted degrees latitude of the basin's research or weather station, from [-90] to [90] degrees.

New Value: 0.0

Click Continue to close this window.

Please select the ParaMESH configuration that you are most comfortable with
avoid seeing this dialog in the future.

Classic (Recommended, MESH Compatib

[ ] Save this preference.

Click Continue to close this dialog.

Figure 1: ParaMESH is a software that provided a rich graphic UI for the configuration of console-based models. This software inspires many of the user-friendly features of CRANE.

ParaMESH is the basis of many of the user-friendly features of CRANE. The largest difference that differentiates the two is that CRANE uses the C# programming language and the Windows Presentation Foundation (WPF) for visualization, whereas ParaMESH used Visual Basic .NET and Windows Forms. CRANE also extends upon the capabilities of ParaMESH by its management of many more types of text-based files and in its explicit use of the scientific workflow approach. In addition, CRANE adopts the eXtensible Markup Language (XML) in its own configuration files, whereas ParaMESH used simple text files.

One limitation of ParaMESH was that its robust UI and core functionality to parse text files were integrated into the same code. A benefit of the .NET programming framework is that it has implementations on other operating systems than Microsoft Windows, including Linux. However, these implementations do not all support the same features of the code. This is particularly true in their limited support of libraries to create certain user-interactive controls and for the visualization of data. Consequently, no part of ParaMESH was portable to other operating systems.

CRANE does not have this same limitation as its UI is separated from its main processing component, also referred to as its ‘engine’. The engine translates data into formats compatible with the tasks of the simulation workflow. The UI provides visualization and a way to configure and communicate with the engine. The engine has no dependency on the UI, whereas the UI does have a dependency on the engine. For the case when the engine can be ported to a system on which the UI is incompatible, it is possible that an alternate UI could be developed using an alternate application framework or set of controls. Because the dependency between the UI and engine is one way, developing a new UI would not require any modification to the engine.

3.2.1 The CRANE Engine

CRANE contains a core collection of code with no UI requirements, referred to as its engine. The engine interprets the requirements of the active tasks in a simulation workflow, facilitating the flow of data between tasks. It also orchestrates the evaluation of tasks in the simulation workflow, using the scientific workflow approach.

The engine contains a number of interfaces to represent tasks and data in the workflow. Every interface that represents a task or data implements the ‘IRegisteredObject’ interface so that every task and all data can be indexed and retrievable by the ‘RegisteredName’ attribute of the interface.

A group of interfaces identified as ‘OperableComponents’ represent tasks of the simulation workflow. All of the interfaces in this group extend the ‘IOperableComponent’ interface. The properties of OOP, which are inherent to the framework by its programming language, allow structures and hierarchies to be made from the interfaces that correspond with the structures of conceptual models and/or integrated modelling systems. As interfaces are structures that cannot directly store data or contain operable code, the engine also contains a number of classes that implement the interfaces. Additional classes can be made to represent new types of tasks by extending an existing class or by creating a new class, which implements one or more of the interfaces.

A second group of interfaces identified as ‘RegisteredObjects’ represent various forms of data, which are analyzed and/or translated by the tasks. Collections of these objects are stored by, or are accessible to, the interfaces of the ‘OperableComponents’ to facilitate the flow of data.

The interfaces of CRANE are intentionally simple, and contain only a few methods, which gives flexibility to developers and reduces risks of compatibility issues by future updates or enhancements to the framework. This also permits a degree of separation between the code of the engine and the code of the tasks, so that both can continue to evolve independently.

In the case where an existing task was written in C#, such as a simulation model built using either the TIME or OpenMI modelling frameworks, CRANE can interact with these because they are all built using the same programming framework. Similar to ModCom, which can interact with any application that is COM-enabled, .NET projects can be compiled and then referenced by other .NET projects. By the same principle, other software can directly interact with and use the interfaces of CRANE. For example, if a GIS were developed using .NET, such as HydroDesktop, it could interact with any geospatial simulation model developed using the interfaces of CRANE.

A simple example of a task to represent a simulation model is illustrated in Figure 2. The ‘ExecutableModel’ class, which can contain operable code, implements the ‘IExecutableComponent’ interface. ‘IExecutableComponent’ extends the ‘IOperableComponent’ interface, such that ‘ExecutableModel’ is recognized by the engine as a task. The base ‘IRegisteredObject’ interface contains a method that returns the ‘RegisteredName’ of the task, so that it can be registered with the engine and retrievable by other tasks.

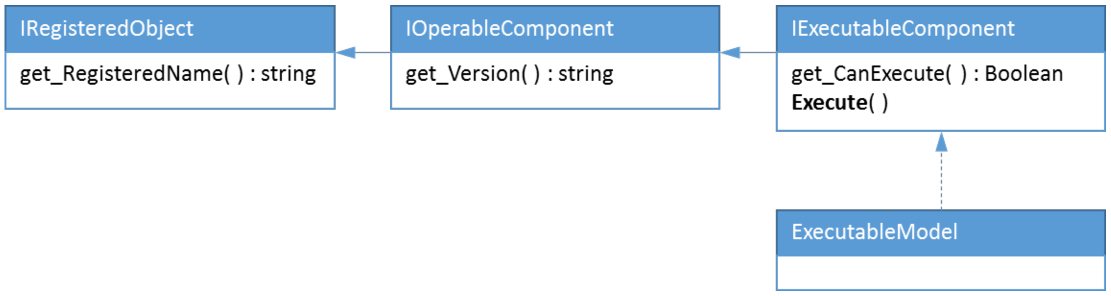


Figure 2: A sample implementation of the ExecutableModel class is derived from a chain of interfaces. ExecutableModel implements the IExecutableComponent interface. The IExecutableComponent interface inherits and extends the IOperableComponent interface, and so forth up the chain to the IRegisteredObject interface.

The ‘ExecutableModel’ class must contain an implementation of the ‘Execute’ method from the ‘IExecutableComponent’ interface, which can contain any operable code. The operable code can be code of the simulation model or code that calls a shell script or batch file to evaluate a model that is external to the engine. A further extension of this task might be the ‘TextFileBasedExecutableModel’ class, which could not only extend the base class but also implement the ‘ITextBasedComponent’ and ‘IFieldValuesComponent’ interfaces to read and write parameter values before or after the execution of the model.

CRANE has the ability to interact with complex input and output file formats or structures with plugins. A plugin is an external file of compiled code that interacts with the framework through an interface, but can be updated independent of the framework and its code. The plugin approach is rather flexible, as it allows interaction with new types of simulation models or modelling tools without requiring updates to the core code of the framework. It also allows components and the framework to continue to develop and evolve independently. For example, a plugin allows the framework to access Microsoft Access and Excel files, and SQLite databases, in addition to text-based files.

For the case where data are stored in structured text files, CRANE supports configuration files written in XML to describe the format of the file. This way, the engine can interact with parameter files, for example, of a simulation model without requiring additional code to parse those files. XML is a broadly accepted standard and flexible scripting language compatible with a variety of other programming languages. For example, once the configuration file has been made to describe the structure of a parameter file, then the XML can be parsed by a website or other software to document and describe the fields of the file for user support (e.g., to other users).

An XML Schema Definition (XSD) is used to enforce the structure of the configuration files. The schema definition is similar to a programming interface as it contains a list of structures and variables allowed or required in the XML file. With the combination of an XML editor and XSD, the allowed and/or required terms are automatically suggested or populated in the configuration file. This makes configuring CRANE for a simulation model relatively straightforward. Creating these files and then using CRANE to create text-based parameter files that use fixed-formatting is less prone to error than creating those files from scratch.

The sequencing of a simulation workflow is also created using XML. Tasks, such as saving parameter files, evaluating a simulation model, renaming, transferring, or translating results, or removing or moving directories or files, are defined and configured as nodes within the XML. Appendix A contains a sample configuration file for CRANE used for the case study in the Chapter 4.

The next section details the UI of CRANE. The UI provides an alternate way of configuring CRANE than by creating XML files, and is in some ways more intuitive and informative than XML to some users.

3.2.2 The CRANE User Interface

One of the design requirements of CRANE was to make it user-friendly. This is primarily achieved by its UI. The UI provides a familiar graphic interface through which the files interpreted by the engine can be configured.

The UI contains a programming interface that is not available in the engine, 'IBrowsableObject', which extends the 'IOperableComponent' interface that defines tasks in the engine. Tasks that implement 'IBrowsableObject' gain code to define a UI and an optional help file. The UI consists of pages that can be navigated to and from, and which contain user-interactive controls. The user-interactive controls are the mechanism by which users can communicate data to the engine, without writing configuration files in XML. The UI uses a style set that applies a universal presentation of headings, textboxes, labels, and many other controls, so that a consistent presentation of information is provided to the user.

The option exists to create a dedicated help page for a task; however, if a control is linked to a class that implements the 'IDisplayInformation' interface, then additional information, such as a description of the control, are automatically displayed to the user. This allows information or support documentation to be readily available to the user without loading the help page.

Different types of information can be displayed depending on the type of class that has been linked to the control. For example, controls that are linked to a class that implements the 'IOperableParameter' interface have the ability to display a description as well as the units of a parameter (e.g., hydraulic conductivity, $cm{\cdot}s^{-1}$). The 'IOperableParameter' interface also provides the ability for the UI to communicate if values in a control have not satisfied predefined constraints.

Constraints are rules that can be defined to ensure that a value (e.g., a parameter value) is within the acceptable operating limits of a task. When a control in the UI is linked to a class that implements this interface, the constraint and if a value in the control has violated that constraint, are both displayed to the user.

CRANE supports for the following rules:

1. the value must be of a particular data type, such as string, real, integer, Boolean, or character;
2. the value cannot be nothing or empty;
3. the value must be greater than or equal to a prescribed value;
4. the value must be less than or equal to a prescribed value;
5. the value must equal a prescribed value, which is sometimes required in a legacy model when a variable or flag must be assigned a specific value, or when a value is to be locked for a training seminar; and/or
6. the value must be one of a set of prescribed values.

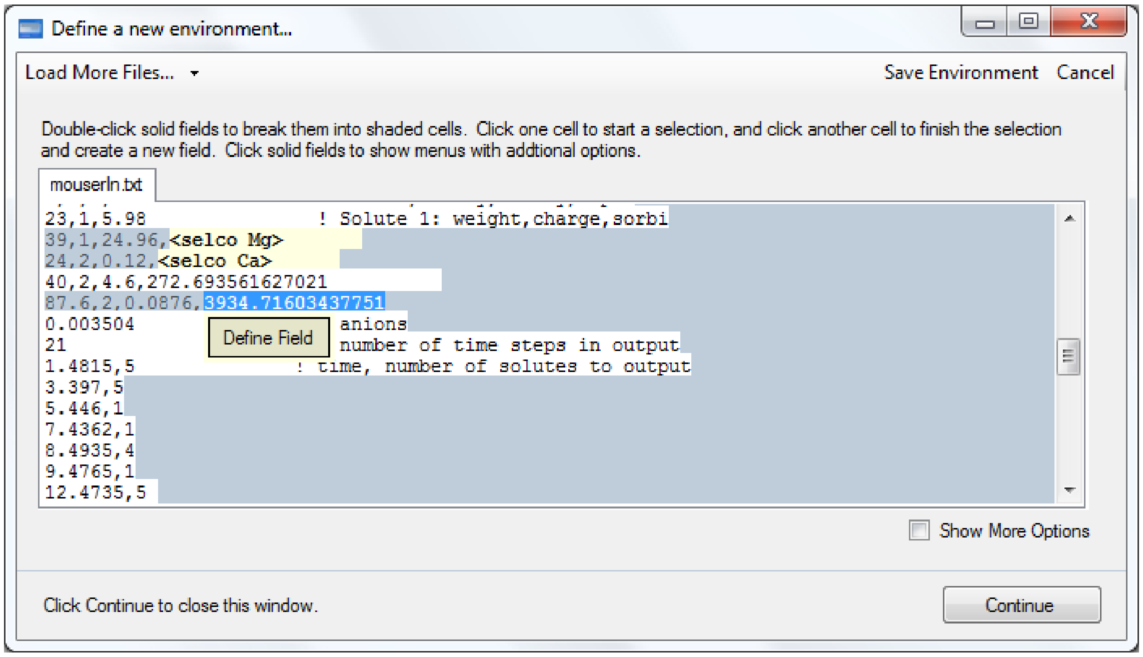
Some of the rules are only active with particular data types. For instance, Rules 3 and 4 can only be applied to values of numeric variables. Additionally, a constraint can be the combination of any number of rules. For example, considering the combination of Rules 3, 4, and 1, such that $\{4 \leq x \leq 6\}$ where x is defined as an integer; the constraint cannot be satisfied by $x = 5.1$, even if the value of x satisfies some of the rules (e.g., Rules 3 and 4), as the value does not satisfy all of the rules (e.g., Rule 1).

When a constraint is a simple rule or a combination of the rules, then it can be declared in XML in the configuration files of the engine or programmatically in the UI of the task. However, complex rules, such as when the value of one variable is tied to the value of another, cannot be configured in the configuration files and must be declared programmatically.

Values in violation of a constraint are not only communicated at the user-control level, but also by the engine. CRANE implements a separate thread dedicated to communicating status messages to the user. Status messages are passed to the user at different stages in the configuration of tasks and in the evaluation of a simulation workflow.

A status message is assigned a severity, which can stop an active task, prompt the user, or log the message and move forward with the process. In the UI, the user can interact with the status message through a dialog or prompt. However, when the engine evaluates a workflow without the UI, no user intervention is permitted. This means that when the engine encounters a message of any moderate severity, the evaluation of the workflow stops. A log of the status message is still available to the user and can provide valuable insight as to why the evaluation of a particular workflow might have failed or if perhaps, a data flow dependency was not fulfilled between two connected tasks.

When a task to represent a simulation model does not implement 'IBrowsableObject' and has no UI, but can be configured using text-based files, then a separate UI in CRANE can be used to parse its input files, as illustrated in Figure 3. This process creates configuration files for CRANE without requiring the user to write XML. It also allows the model then to interact with additional tasks in CRANE. For example, the model can be added to workflows; and if its parameters have been defined, can link to calibration tools. Once configuration files for the model have been made for CRANE, additional benefits include that a history can be saved of the parameter configurations used to run the model; that a simulation workflow that contains the model can be saved; and that the configuration files themselves can be shared with collaborators and other users.



Double-click solid fields to break them into shaded cells. Click one cell to start a selection, and click another cell to finish the selection and create a new field. Click solid fields to show menus with addtional options.

```
23,1,5.98                ! Solute 1: weight,charge,sorbi
39,1,24.96,<selco Mg>
24,2,0.12,<selco Ca>
40,2,4.6,272.693561627021
87.6,2,0.0876,3934.71603437751
0.003504                   anions
21                         number of time steps in output
1.4815,5                 ! time, number of solutes to output
3.397,5
5.446,1
7.4362,1
8.4935,4
9.4765,1
12.4735,5
```

Click Continue to close this window.

Figure 3: CRANE can link to some types of simulation models by an intuitive user interface, which does not require the user to write programming code.

In summary, programming requirements were determined for CRANE to make it a flexible framework that would allow tasks, such as simulation models and modelling support tools, to form an integrated modelling system, which would be evaluated using the scientific workflow approach. To satisfy these requirements, CRANE was written using an object-oriented programming language, with its core processing component separated from its user interface, support for plugins that can be updated and enhanced independent of the framework, and with intuitive user-friendly features, an interface, and human-readable configuration files.

The flexibility of the framework allows it to theoretically interact with a variety of simulation models and modelling support tools. Perhaps its strength is its ability to connect to legacy simulation models, whose code cannot be modified, through text files, supporting both free- and fixed-formatting. It can also do this with its own UI, which requires no additional programming code on the part of the user. Lastly, its ability to store configuration information in XML means that workflows and model configuration information can be saved and distributed in a transparent format to other users.

The next chapter presents a case study that demonstrates the ability of CRANE to wrap around, configure, and evaluate two versions of a hydrologic simulation model.

Chapter 4
Case Study

A case study is a way of demonstrating the capabilities of CRANE. This case study demonstrates using CRANE to configure the parameter input files of a simulation model; to modify these files for an alternate modelling scenario; manage the modelling workspace; and record the simulation workflow as a session, which contains information such as the parameters modified, as well as their original values.

The case study is a comparative analysis of two configurations of the semi-distributed hydrologic model MESH (Pietroniro et al. 2007). The first is the default configuration of the model, and the second enables an extra scientific module in the model. The objective of the study was to assess if including a new scientific module improved the results of the simulation.

4.1 Introduction

Cold regions are geographic areas that experience a colder climate, cold winters, and whose water storage is strongly tied to seasonal precipitation (Pomeroy et al. 2007). These regions experience widespread snow cover (Fang et al. 2010). The melting of a continuous layer of snow, called the snowpack, that covers soil during the spring freshet may result in one of the largest and most significant flow events of the year (Gray et al. 2001; Burn, Fan & Bell 2008). The substantial runoff generated from the melted snow can replenish reservoirs, lakes, and rivers, and has a significant impact on agriculture, irrigation, and on local drinking water supplies.

Recent climactic trends show that climate change is affecting seasonal precipitation, thus altering meltwater and streamflow levels, as well as the timing of flow events (Burn, Fan & Bell 2008; Poitras et al. 2011). Simulation modelling can help assess the potential impact of these types of changes. However, this practice is only useful if the simulation model can provide a good representation of the physical processes that dominate these types of regions.

Previous studies by Granger et al. (1984), Gray et al. (2001), and Zhao & Gray (1999) have improved our understanding of the physical process of the infiltration of meltwater into frozen soils; however, it continues to be poorly represented in many hydrologic simulation models (Spence et al. 2013). The primary objective of this study was to enhance the physical basis of the land-surface hydrologic model MESH (Pietroniro et al. 2007) by improving its representation of this process. Data from a cold region basin, the Assiniboine Headwaters in Central Canada, were used as inputs to the model. A second objective of this study was to demonstrate the ability to CRANE to help facilitate this type of comparative analysis.

4.2 Background

In some regions, meltwater is an essential component of the water cycle as it replenishes reservoirs for drinking water and for crop irrigation. Therefore, it is important to reliably determine the amount of meltwater ($MELT$) from the snowpack that will infiltrate (INF) a frozen, thawing, or refreezing soil, and the component that will runoff (OVR) and directly contribute to the spring freshet.

The separation of $MELT$ into the components of INF and OVR depends on the infiltrability of the soil. Infiltrability is a qualitative measure of the ability of a soil to allow infiltration (Granger, Gray & Dyck 1984; Gray et al. 2001). The Division of Hydrology at the University of Saskatchewan (Granger, Gray & Dyck 1984) has defined three classes of soil infiltrability for freezing and frozen soils, outlined below and illustrated in Figure 4.

Classes of soil infiltrability:

1. 'Unlimited infiltration' occurs in soils that have surface-connected, air-filled cracks. Gravity forces primarily govern the flow of the meltwater into and throughout the soil matrix. The amount of $MELT$ that infiltrates to the point of saturation is $INF = MELT_{inf}$. The excess, which cannot infiltrate the saturated soil, is $OVR = MELT - MELT_{inf}$.
2. 'Restricted infiltration' occurs in soils where an impervious boundary blocks access to its surface-connected, air-filled cracks. The amount of $MELT$ that infiltrates is negligible, such that $INF = 0$. Thus, all of the available $MELT$ will runoff, such that $OVR = MELT$.
3. 'Limited infiltration' occurs in soils where only some of the surface-connected, air-filled cracks are accessible. An impervious boundary blocks access to the remaining fraction. Capillary forces primarily govern the flow of the meltwater through the soil matrix. The amount of $MELT$ that infiltrates is a function of the temperature (T), and of the liquid (θ_l) and frozen (θ_f) water contents of the soil, such that $INF = MELT_{inf}(T, \theta_f, \theta_l)$. The excess, which does not infiltrate the soil, is $OVR = MELT - MELT_{inf}(T, \theta_f, \theta_l)$.

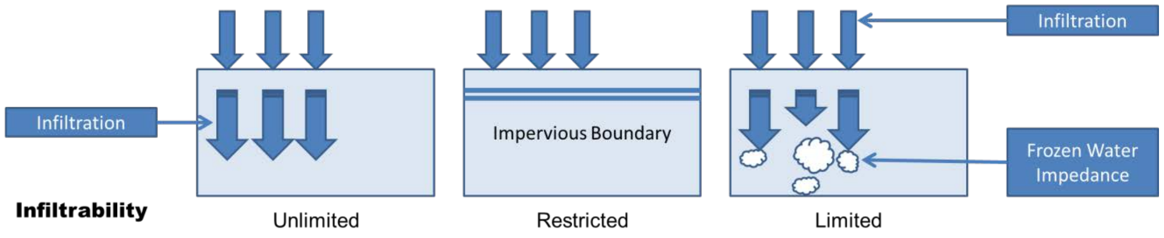


Figure 4: Classes of soil infiltrability. Unlimited infiltration occurs when the entire soil matrix is permeable, restricted infiltration occurs when the soil is blocked an ice lens and is not permeable, limited infiltration occurs when the soil matrix is partially permeable.

The three classes of infiltrability were determined based on a series of observations collected in the Canadian Prairies (Granger, Gray & Dyck 1984; Zhao & Gray 1999). Later studies have enhanced our understanding of limited infiltrability. Pomeroy et al. (2007) showed that, on average, the meltwater only infiltrates the top ~260 mm of the soil profile, as a result of the meltwater refreezing when cooler temperatures are encountered at lower depths in the soil. Furthermore, Zhang et al. (2010) determined a correlation between the amount of meltwater that infiltrates, and the existing liquid and frozen water contents of the soil.

Zhao & Gray (1999) derived an empirical relationship to estimate the total potential amount of $MELT$ that could infiltrate a soil of limited infiltrability, $MELT_{inf}$ (mm), given by Eq. 1:

$$MELT_{inf} = C \cdot {S_0}^{2.92}(1 - S_I)^{1.64} \left[\frac{273.15 - T_I}{273.15}\right]^{-0.45} {t_0}^{0.44} \qquad \textbf{Eq. 1}$$

C is a soil-dependent coefficient, S_0 is the moisture content at the surface of the soil (mm^3·mm^{-3}), S_I is the average saturation of the liquid and frozen water contents of the infiltrable top 260 mm of the soil (mm^3·mm^{-3}), T_I is the average temperature of the infiltrable layer of the soil (K), $T_I \geq 273.15$, and t_0 is the time that has passed from when the snow began melting, called the opportunity time.

The relationship assumes quasi-steady state flow, a homogeneous soil matrix, and a constant temperature and moisture distribution in the soil profile. The equation implies that the amount of water that infiltrates the soil is proportional to its available air-filled pore space. However, the existing saturation of the soil limits the amount that will actually infiltrate.

The opportunity time, t_0 (h), in Eq. 1 is a function of the snow water equivalent of the snowpack, SWE (mm), and can be determined by Eq. 2:

$$t_0 = 0.65 \cdot SWE - 5 \qquad \textbf{Eq. 2}$$

Eq. 2 is an empirical relationship derived from observations from the Canadian Prairies. The equation may not be applicable to all sites because of the site specificity of the data used to derive the relationship.

4.3 Modelling Strategy

Many hydrologic simulation models have poor or completely lack representation of the infiltration of meltwater into frozen, thawing, or refreezing frozen soils (Zhang et al. 2010; Spence et al. 2013). Recent studies have aimed to improve this in certain models, such as in the Cold Regions Hydrological Model (CRHM) (e.g., Ellis et al. 2010, Fang et al. 2010, Pomeroy et al. 2007). This study aimed to update this representation in the land-surface hydrologic model MESH.

MESH (Pietroniro et al. 2007) is a semi-distributed hydrologic simulation model that, among other variables, simulates streamflow. The model couples the Canadian Land Surface Scheme (CLASS) (Verseghy 2000) with a river channel routing scheme adapted from the WATFLOOD model (Kouwen 1988). Detailed descriptions of MESH are available in publications by Verseghy (2008), Soulis et al. (2005), Davison et al. (2006), and Pietroniro et al. (2007). In general, CLASS is run using the data of seven atmospheric inputs to calculate energy and water fluxes at the ground surface (Verseghy 2000). The river channel routing scheme takes the runoff from these calculations to simulate streamflow through a network of connected grid cells (Pietroniro et al. 2007; Davison et al. 2006).

The melting of the snowpack occurs in the CLASS portion of the model. Meltwater forms when there is sufficient energy to melt the snowpack. In the original version of the model, the meltwater contributes to the liquid water content of the snowpack. It begins to contribute to the precipitation incident at the ground surface when it exceeds the retention capacity of the snowpack (Verseghy 2008, pp. 140-141). In this case, the precipitation rate includes the original precipitation rate, meltwater, as well as the excess of liquid water content from the snowpack (Verseghy 2008, pp. 143-144).

A modification to the model allowed it to calculate the amount of meltwater that could potentially infiltrate a frozen, thawing, or refreezing soil, using Eq. 1 and Eq. 2. A configuration flag is used to enable this option, and additional parameters are available to override the calculation of the opportunity time for cold regions that might not be characterized by Eq. 2. In this case, the meltwater in excess, which would not be able to infiltrate the soil, contributes to ponded water storage at the surface of the soil, instead of to the precipitation incident at the ground surface. The ponded water storage does not immediately contribute to runoff or to overland flow, which allows the opportunity for the water to potentially infiltrate in a later time-step. However, if this storage exceeds a fixed ponding limit, then it directly contributes to runoff and to overland flow.

The modification does not change the total water storage in the system. However, it could affect the energy distribution of the system, as the transfer of the meltwater modifies the temperatures of the precipitation incident and ponded water storage at the surface of the soil.

4.4 Site Description

The catchment used for this study was the Assiniboine Headwaters basin (Figure 5). The study area is approximately 3,340 km^2 and forms part of the Upper Assiniboine River basin. The Assiniboine River drains from eastern Saskatchewan into western Manitoba, Canada, and terminates at Red River in Winnipeg, Manitoba. The surface elevations of the basin range from 436 m to 633 m above sea level.

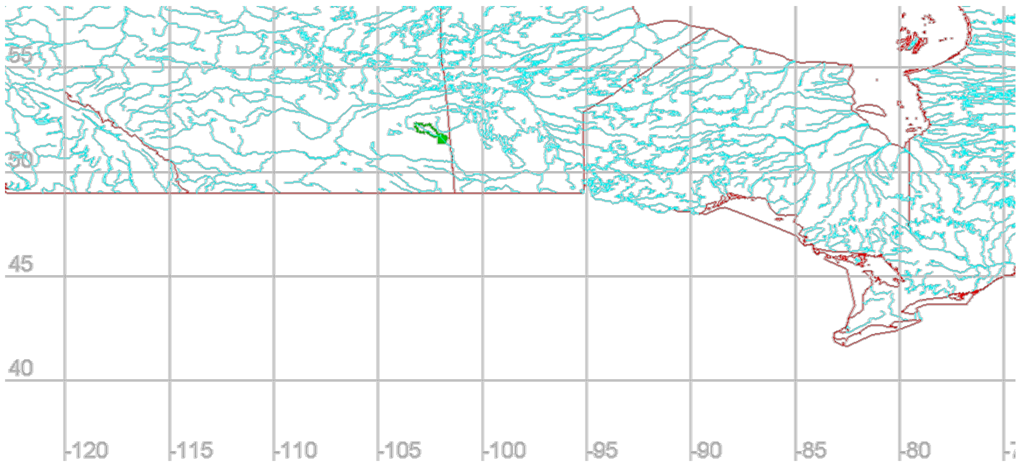


Figure 5: Location of the Assiniboine Headwaters basin in the Canadian Prairies with provincial boundaries and national rivers and lakes shown.

The basin is designated a cold region within Canada and experiences long cold winters and short warm summers. Snowmelt accounts for more than 80% of the total annual surface runoff, despite the fact that snowfall only accounts for approximately one third of the average 450 mm of annual precipitation (Manitoba Conservation, Saskatchewan Water Corporation & Environment Canada 2000, p. 13; Shretha, Dibike & Prowse 2012).

The National Ecological Framework Report (Marshall & Schut 1999) delineates the basin into six ecodistricts, four of which exhibit distinct soils and land use classification. The remaining two differ by name, but otherwise share identical attributes. The six ecodistricts are presented in Table 2.

Table 2: Ecodistricts of the Assiniboine Headwaters.

ECODISTRICT ID	NAME	DOMINANT SOILS	DOMINANT LAND USE
707	Barrier River Upland	Loam	Cropland
709	Swan River Plain	Loam, Clay loam	Cropland
714	Porcupine Hills	Loam, Organic	Mixed forest
715	Duck Mountain	Clay loam, Loam, Clay	Broadleaf forest, Mixed forest
745	Quill Lake Plain	Loam	Cropland
749	Yorkton Plain	Loam	Cropland

Soils and land use classification contribute to the parameterization of the model. Soils in the area are predominantly loamy, with some of the sub-catchments exhibiting higher clay content. The Porcupine Hills sub-catchment is unique in that it has a significant percentage of organic soils.

As for land use, croplands dominate four of the six sub-catchments, and forests dominate the remainder. The Quill Lake Plain and Yorkton Plan sub-catchments share identical attributes, and can be merged to simplify the parameterization of the model.

4.5 Configuration of the Model

Certain data are required to configure the model, including data extracted from a Digital Elevation Model (DEM), from which a database of the drainage properties of the basin were extracted. The DEM was acquired from the Canadian Digital Elevation Data product.

The Green Kenue software was used to create a gridded cell representation of the basin from the DEM. The basin is captured by approximately 48 square grid cells contained within a mask that spans

15 cells in the x-direction and 9 cells in the y-direction. Each grid cell spans approximately 11 km in both the x- and y-directions, with the total drainage area of the basin approximating 3,340 km^2.

The model required hourly meteorological data for six variables: air temperature, wind speed, relative humidity, pressure, incoming shortwave radiation, and net longwave radiation. These variables were extracted from the Canadian Global Deterministic Prediction System (GEM) to match the spatial extent of the drainage database. The model also required hourly precipitation rate data, which were extracted from the Canadian Precipitation Analysis (CaPA). The set of meteorological data spanned from January 1, 2002, to December 31, 2014.

While the meteorological input data started from January 1, 2002, the simulation start date was set to June 1, 2002, to normalize the season-dependent initial conditions of the model. Doing this also allowed more time for the model to spin-up before the onset of the spring freshet. The spring freshet is one of the largest hydrologic events observed in this region, and was therefore the largest event simulated by the model.

The performance of the model was assessed by comparing the simulated to measured daily streamflow values, extracted from the National Water Data Archive from the Water Survey of Canada. Among the metrics calculated by the model are the Nash-Sutcliffe coefficient (R_{NS}^2) (Nash & Sutcliffe 1970) and the Nash-Sutcliffe coefficient of log-transformed streamflow values. The latter is particularly useful for this basin, where a low flow regime dominates most of the hydrologic year (Perrin, Michel & Andréassian 2003; Krause, Boyle & Bäse 2005). Values of these coefficients range from $-\infty$ to 1, where $R_{NS}^2 = 1$ would represent a perfect correlation between simulated and measured values.

Daily streamflow measurements were acquired for three streamflow gauges. The identification detail of the gauges is presented in Table 3, and the locations are illustrated in Figure 6.

Table 3: Streamflow gauge detail for the Assiniboine Headwaters.

GAUGE ID	NAME	DATA DESCRIPTION
05MC001	Assiniboine River at Sturgis	Seasonal data from 2002-2013
05MC002	Stony Creek near Stenen	Seasonal data from 2002-2014
05MC003	Lilian River near Lady Lake	Seasonal data from 2002-2014

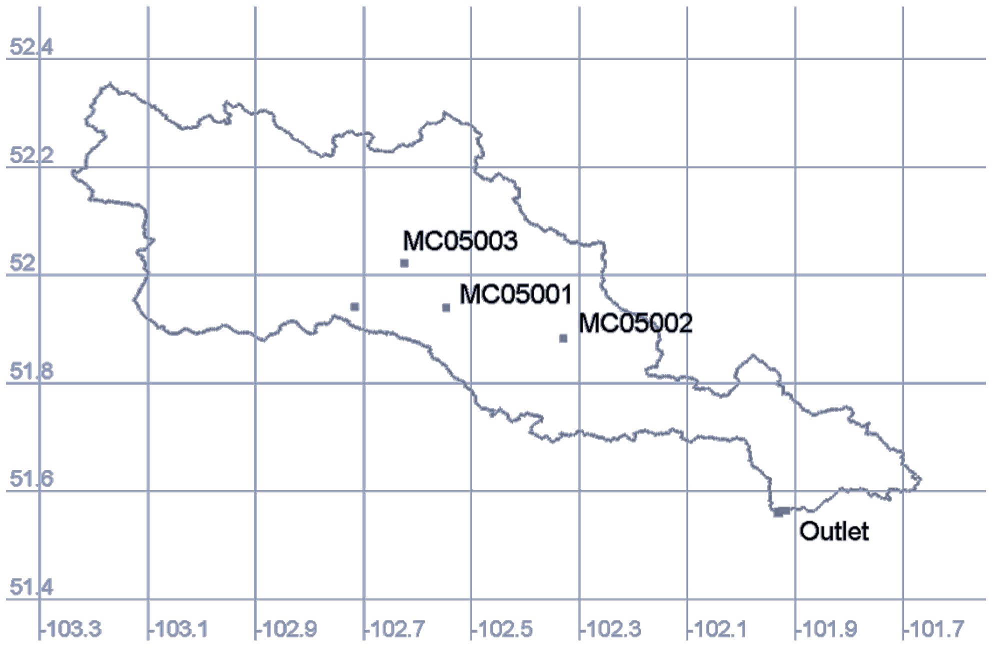


Figure 6: The locations of streamflow gauges in the Assiniboine Headwaters basin.

Finally, where possible, values for the parameters of the model were estimated according to data provided at the ecodistrict level from the National Ecological Framework. Soils information were based on the Soil Landscapes of Canada product (Soil Landscapes of Canada Working Group 2006). Values for the parameters of the CLASS portion of the model were extracted from Verseghy (2008). Values for hydraulic parameters were assigned from Dingman (2008). An XML file was created to describe the structure of a configuration file for the model for CRANE; this is presented in Appendix A.

The flow of the tasks of the simulation workflow is illustrated in Figure 7. Two identical series of tasks, one for the original and one for the modified version of the model, are shown. The difference between the two series is that the module for the infiltration of frozen meltwater into freezing, thawing, or refreezing soils has been activated in one of the configurations, but not in the other. The statistics, and output files for the measured versus simulated streamflow, and basin water balance are preserved at the end of each task.

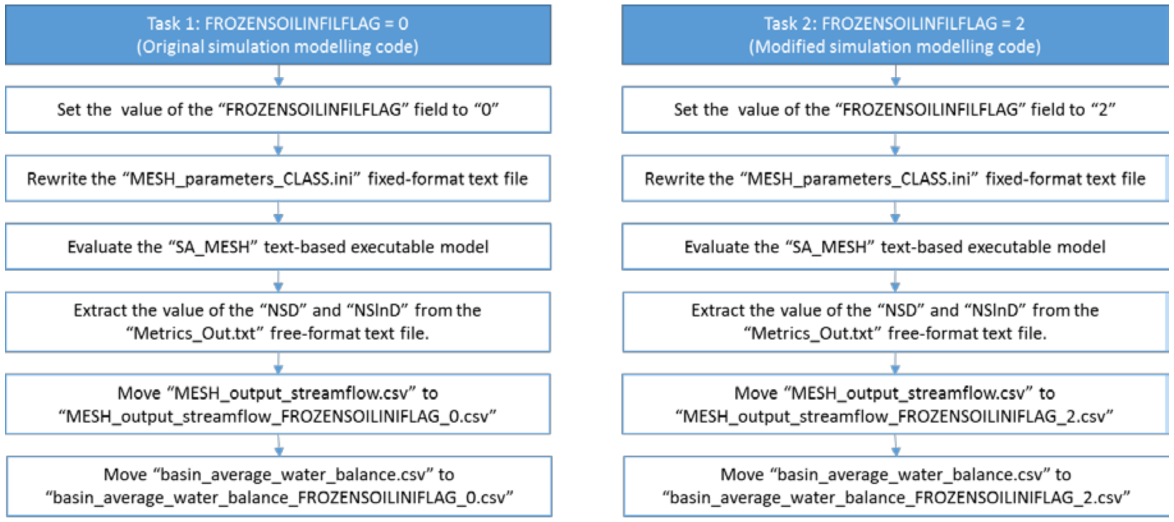

Task 1: FROZENSOILINFILFLAG = 0
(Original simulation modelling code)
Set the value of the "FROZENSOILINFILFLAG" field to "0"
Rewrite the "MESH_parameters_CLASS.ini" fixed-format text file
Evaluate the "SA_MESH" text-based executable model
Extract the value of the "NSD" and "NSlnD" from the "Metrics_Out.txt" free-format text file.
Move "MESH_output_streamflow.csv" to "MESH_output_streamflow_FROZENSOILINIFLAG_0.csv"
Move "basin_average_water_balance.csv" to "basin_average_water_balance_FROZENSOILINIFLAG_0.csv"
Task 2: FROZENSOILINFILFLAG = 2
(Modified simulation modelling code)
Set the value of the "FROZENSOILINFILFLAG" field to "2"
Rewrite the "MESH_parameters_CLASS.ini" fixed-format text file
Evaluate the "SA_MESH" text-based executable model
Extract the value of the "NSD" and "NSlnD" from the "Metrics_Out.txt" free-format text file.
Move "MESH_output_streamflow.csv" to "MESH_output_streamflow_FROZENSOILINIFLAG_2.csv"
Move "basin_average_water_balance.csv" to "basin_average_water_balance_FROZENSOILINIFLAG_2.csv"

Figure 7: Flow of two tasks in the simulation workflow, one for the original and the other for the modified version of the simulation modelling code. In the first case, a flag that enables a coding modification is disabled. In the second case, the flag is enabled. Output is preserved in both cases.

4.6 Results

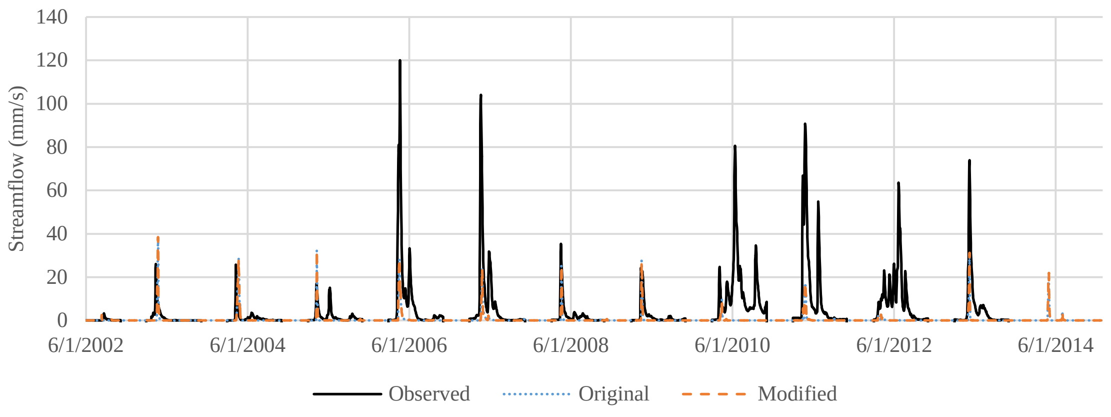
The original and modified versions of the model ran using the default parameterization scheme. The difference between the two configurations is that the modified version has an extra module for the infiltration of frozen meltwater into freezing, thawing, or refreezing soils. Both configurations ran for the full length of the meteorological record, from June 1, 2002, to December 31, 2014. Streamflow results from the runs are illustrated in Figure 8 for the three gauge locations. The peaks of the hydrographs correspond to the ephemeral streams of the basin responding to an input of water to the system. Peaks occurring on or around the beginning of April primarily correspond to snowmelt-related events, and peaks occurring later in the spring and into the summer correspond to rainfall events.

With the default parameter scheme, the model does not capture the hydrologic response or behavior of the basin. The timing of the peaks is relatively correct; however, the magnitude of the peaks is nowhere near that of the observed values. This is reflected in the statistical metrics from the model. The Nash-Sutcliffe coefficients and the Nash-Sutcliffe of the log-transformed streamflow values are presented in Table 4.

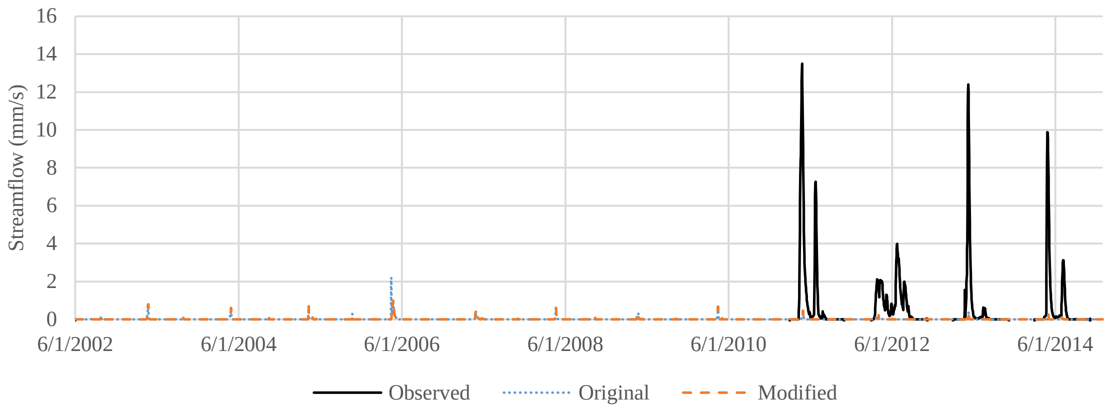
Table 4: Coefficients for Nash-Sutcliffe for the normal streamflow values (R_2^{NS}) and Nash-Sutcliffe of the log-transformed streamflow values (R_2^{NS} ln o_i) for three streamflow gauges.

GAUGE ID	ORIGINAL	MODIFIED	R_2^{NS}	R_2^{NS} ln o_i
05MC001	X		-9.283E-03	-6.413
		X	-9.092E-03	-6.414
05MC002	X		-0.1423	-0.3561
		X	-0.1419	-0.3526
05MC003	X		-4.404E-02	-8.398
		X	-4.399E-02	-8.398

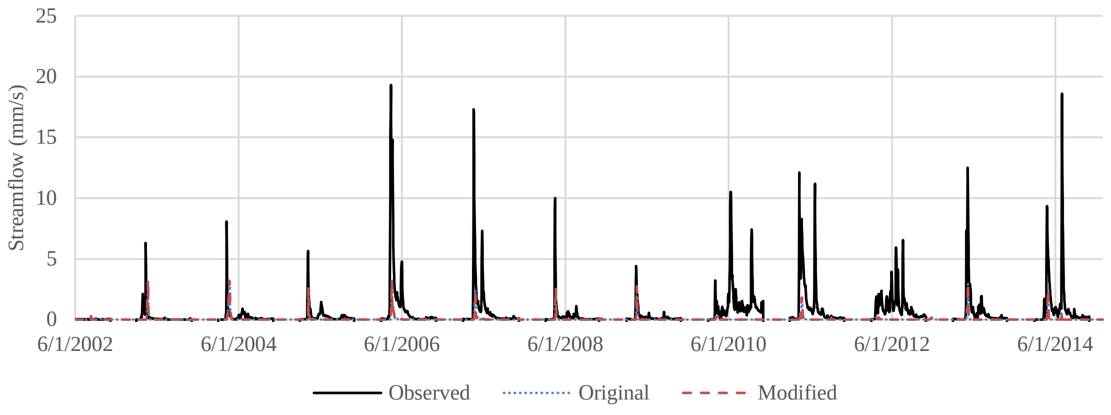
The values of the coefficients are near identical between both configurations of the model. Interestingly, the statistics improve in the modified version of the model; however, the improvement is marginal. The results were achieved using the default parameterization scheme of the model, and would likely improve with a calibration exercise of the parameters of the model (Refsgaard 1997; He, Bárdossy & Zehe 2011).



Gauge 05MC001



Gauge 05MC002



Gauge 05MC003

Figure 8: Hydrographs for three gauges of measured to simulated streamflow from both the original and modified configurations of the model.

The total accumulated runoff and its components of overland flow, interflow, and baseflow are presented in Table 5. There is a discrepancy between the original and modified versions of the model in the overland flow and interflow components. The cumulative difference between the runoff from the original and modified versions of the model is illustrated in Figure 9.

Table 5: Total accumulated runoff and runoff components of overland flow, interflow, and baseflow at the basin outlet.

CONFIGURATION	TOTAL ACC. RUNOFF (MM)	FROM OVERLAND FLOW (MM)	FROM INTERFLOW (MM)	FROM BASEFLOW (MM)
ORIGINAL	5.69E+01	1.92E+00	5.50E+01	0.00E+00
MODIFIED	5.67E+01	1.68E+00	5.51E+01	0.00E+00

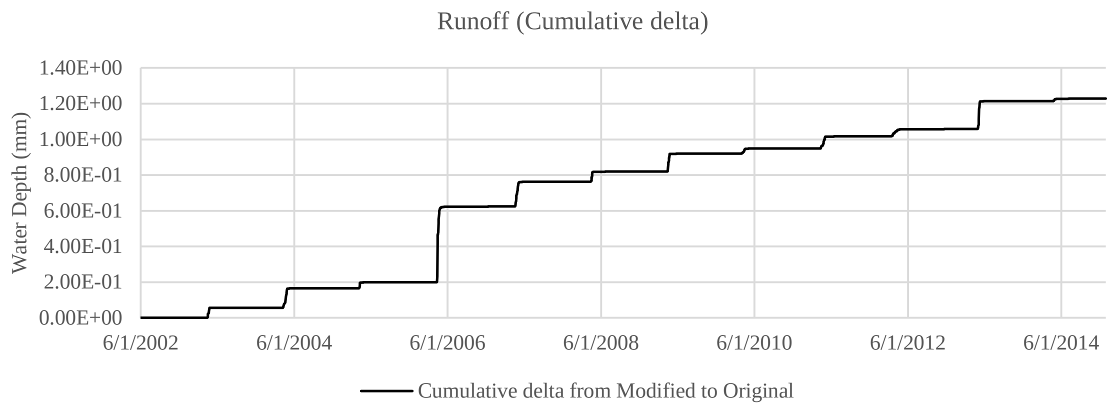


Figure 9: Difference in cumulative runoff between the original and modified configurations of the model at the basin outlet.

The discrepancy occurs during the spring freshet after snow has begun to melt. The water levels of the runoff, evaporation, ponded storage at the surface, and the liquid and frozen water contained in the soil profile, are illustrated in Figures 10-13, for a subset of the series.

This subset spans approximately two weeks during the peak of the spring freshet. The different distributions of water between the two versions of the model are evident from these figures. This discrepancy is the topic of discussion in the following section.

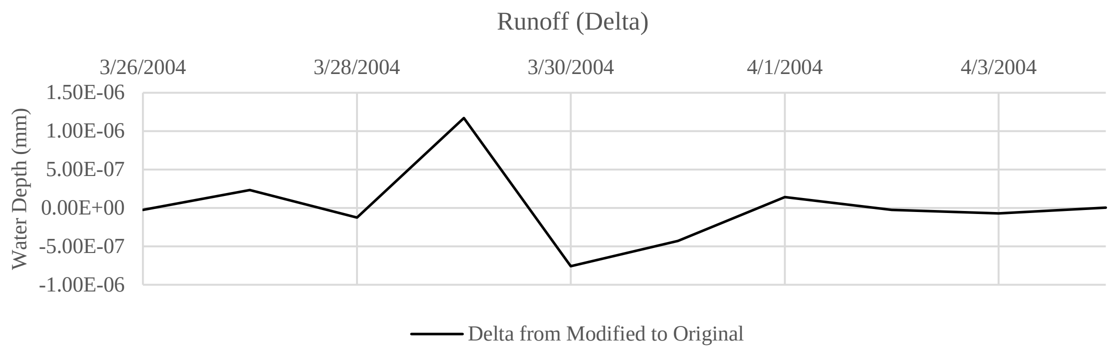


Figure 10: Difference in runoff (mm) simulated between the modified and original versions of the model.

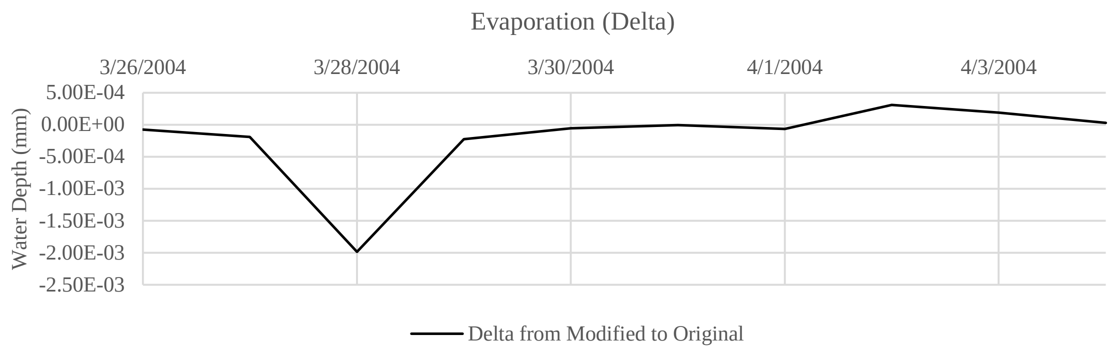


Figure 11: Difference in evaporation (mm) simulated between the modified and original versions of the model.

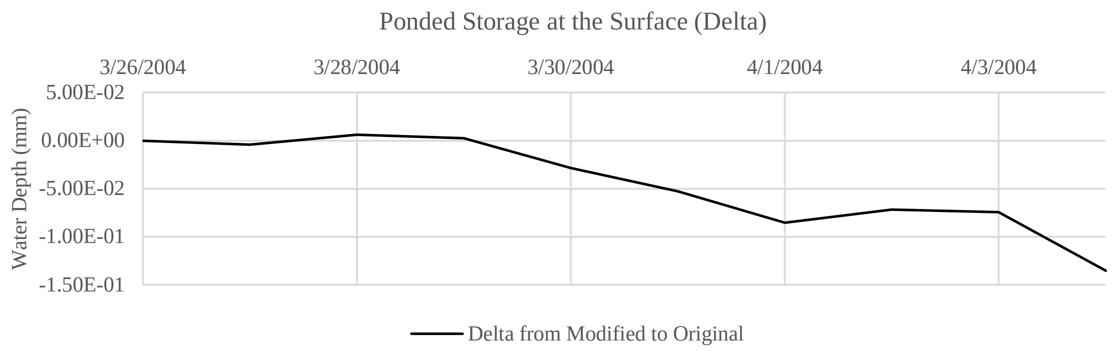


Figure 12: Difference in ponded storage at the surface (mm) simulated between the modified and original versions of the model.

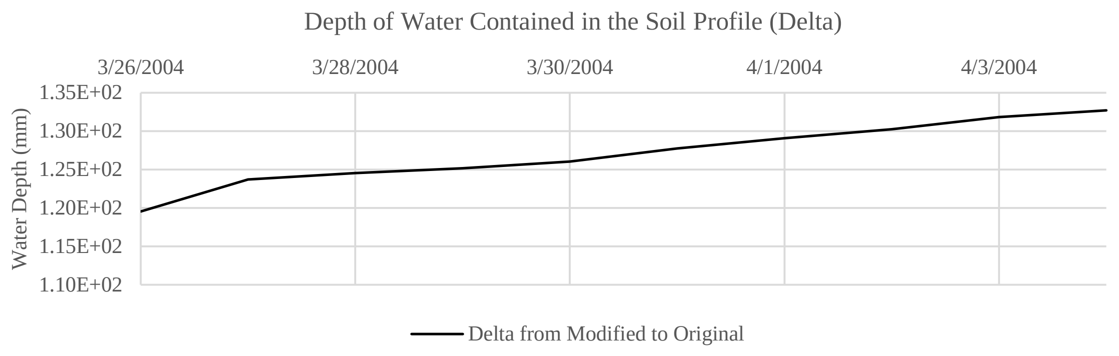


Figure 13: Difference in depth of water contained in the soil profile (mm) simulated between the modified and original versions of the model.

4.7 Discussion

This study saw the MESH land-surface hydrologic model (Pietroniro et al. 2007) modified to include a more physically based representation of the infiltration of meltwater into freezing, thawing, and refreezing soils. This process is an important component of the water cycle in cold regions, where snowmelt-related events contribute to one of the largest flow events of the year (Gray et al. 2001).

A more physically based representation of this process is particularly important in simulation models used for forecasting, as the current climate and the timing of snowmelt events continues to change (Burn, Fan & Bell 2008). A more physical representation of the process, which is based more so on physiographic and meteorological input than fitting coefficients, was hypothesized to provide a more consistent prediction for forecasts of flooding or drought in a changing climate.

Recent modifications to the model added two equations from Zhao & Gray (1999). The equations estimate the depth of meltwater that could infiltrate a frozen, thawing, or re-freezing porous soil. The actual amount that would infiltrate depends upon the availability of meltwater and the existing saturation of the soil. A further modification to the model contributed excess meltwater, which would not be able to infiltrate the soil, to ponded water storage at the surface of the soil instead of to the precipitation incident at the ground surface

The model ran with a default parameterization scheme, using values from Verseghy (2008), physiographic input data, and hydraulic values from Dingman (2008). Using the default parameterization scheme, the model did not capture a sufficient hydrologic representation of the basin. However, activating the meltwater infiltration algorithm marginally improved the results.

The results show that up until the spring freshet, the two configurations of the model generate identical runoff results. However, after the spring freshet, the results for simulated runoff diverge. In the modified configuration of the model, the amount of meltwater that might infiltrate the frozen soil is redistributed from runoff to ponded storage at the surface of the soil; this is evident on March 28, 2004, in Figure 10. In the original configuration of the model, the excess water is lost by evaporation.

The modified configuration of the model eventually allows the meltwater to infiltrate, as is evident by the upward trend in Figure 13. Overland flow rarely directly contributes to streamflow (Dingman 2008, p. 437), so land surface schemes sometimes rely on unrealistic rates of evaporation to compensate for a poor representation of surface hydrology (Soulis et al. 2000). In the modified version of the model, the amount that cannot infiltrate eventually contributes to runoff, as evident on March 29, 2004 in Figure 10.

This process occurs every freshet, which explains the difference in accumulated runoff between the two versions of the model, illustrated in Figure 9. It is extremely likely that the results of the model, and its ability to capture the hydrologic regime of the basin, would greatly improve with a calibrated parameter set. In such a case, it would be worthwhile to observe if activating the module would still only marginally improve the simulated results, between the two versions of the model. CRANE supports linking parameterized field values to a calibration utility, such as the OSTRICH optimization toolkit (Matott 2005). This study could benefit from this type of link to calibrate parameter values.

The design of the study incorporated only two configurations of the model; however, a benefit of using CRANE is that the framework was able to evaluate the model and manage the workspace, so that the two configurations of the model could be run, and their results could be saved and preserved, in the same folder. The meteorological input data are very large for this model, so it is inconvenient to maintain multiple copies of the files. Because the workflow is saved in a transparent format, the study could be more easily replicated and transferred between systems than if it was performed using a batch file or shell-script. A drawback to using CRANE for this type of modelling exercise is generating its configuration data to describe the input files for the model. This is a relatively laborious task; however, its use of an XML schema definition means that most of the fields required in the configuration files are generated automatically by an XML editor. In addition, once the files are created, they can be copied and used again in another study that would use the same model.

Table 6 compares a few means by which this study could be accomplished, with varying degrees of computer programming knowledge, scripting knowledge, or from a ‘new user’ perspective.

Table 6: Methods for evaluating a comparative modelling analysis with varying degrees of computer programming knowledge, scripting knowledge, or from a 'new user' perspective, with strengths highlighted. In contrast, batch scripting requires knowledge of scripting language(s); manual evaluation is especially prone to typographic error; versus communication of the syntax of files from CRANE requires an existing knowledge of the framework.

CAVEAT	BATCH SCRIPTING	MANUAL EVALUATION	CRANE
Programming Requirements	Scripting knowledge: At the basic level: MS-DOS commands, such as 'cd', 'xcopy', 'move'; At the intermediate level: 'mklink'; Alternatively, with a utility such as Cygwin, 'mpirun'. Programming knowledge: none.	None	**Scripting knowledge: Optionally, XML.** Programming knowledge: Optionally, C# or XML.
Expert Knowledge	Fixed-format text files with explicit spacing rules.	Fixed-format text files with explicit spacing rules.	Understanding of fixed-format text files (e.g., of field spacing).
Workspace Requirement	At the basic level, using folder replication: 2 replicates; At the intermediate level, using symbolic links: 2 replicates, 1 set of common input files.	2 replicates.	**1 replicate; the input and output files to be preserved are managed by CRANE.**
Effort	Writing input files with formatting constraints. Writing batch file or shell script in simple text editor. Debugging (i.e., solving errors in the script) by trial-and-error.	Writing input files with formatting constraints. **Copying and pasting files.** Higher level of folder management, and of keeping track of where alternate configurations or outputs have been stored.	Writing XML; with an XML editor, the required fields are generated automatically. OR— **No scripting or programming requirements using the user interface for the text-based simulation model.**
Portability	Commands in the batch file are transferrable to similar systems. Paths in the batch file may not be transferable.	**Can copy and paste files.** Paths in companion spreadsheets (e.g., post-analysis) are likely not transferable.	Can be parsed by CRANE on other systems. Can be potentially transferred to other programming platforms or operating systems.

Batch scripting is common in many engineering and scientific modelling analyses. A benefit of this approach is that an experiment can be automated using a script. However, batch scripts cannot always be transferred between systems, and are often tied to a particular user profile or user folder structure, and so sometimes cannot be transferred between users either. Moreover, at a basic level, the user still requires knowledge of scripting commands or terms, syntax, and structure. This approach still requires creating the parameter input files for the model; and in this particular case study, these files have strict formatting constraints, and so are especially prone to typographic errors.

Manual evaluation has the benefit of simplicity and relative transparency. The method is relatively transparent in the sense that all of the files and the folder structure are immediately obvious to an outside observer. However, it is potentially the least efficient in terms or workspace organization and efficiency. Duplicates of all files are required between simulations, to preserve the setup, and there is no obvious distinction between configurations without meticulous labelling of the folder structure or other files, or by the comparison of the folders or files using a third-party utility. Similar to the method of batch scripting, the approach still requires creating the parameter input files for the model, which in this case study, have strict formatting constraints. To a user who does not already have knowledge of the simulation model, or of the formatting constraints of these types of files, the spacing requirements would not be obvious. Moreover, in this particular case study, the model has no way of communicating to the user formatting violations or typographic errors. For example, if a user has accidentally used a tab to space a field in this type of file, the spacing constraint of the field would be in error, the model would crash with a generic error, and it is extremely unlikely that the user using a generic text editor could differentiate the tab from regular spacing to identify and remediate the error.

The last of these methods is CRANE, which provides a proxy to the strict formatting constraints of the input file of the simulation model either in the form of structured XML or its user interface. When using the interface, CRANE derives the necessary XML; or else with an XML editor, the CRANE XML schema definition validates code written by the user, preventing typographic error. CRANE also extracts and updates only the variables required to change between the configurations of the model, so that no duplication of the folder or other files is required to evaluate the experiment. However, a potential collaborator would be required to install the framework, or otherwise gain knowledge of the syntax and structure of its configuration files, to replicate or re-evaluate the experiment.

4.8 Conclusions

Modifications were made to include an updated representation of the infiltration of meltwater into frozen, thawing, or re-freezing soils in the MESH land-surface hydrologic model. Physiographic, land use, soils, and meteorological data were collected for the Assiniboine Headwaters basin. Two configurations of the model, the original and the modified version, ran with a default set of parameter values.

The simulated results from the model did not capture the hydrologic regime of the basin. The modified version of the model produced slightly improved results; however, the improvement across three streamflow gauges was marginal, if not negligible, and clouded by very poor initial performance.

The modified version of the model did, however, redistribute water during the spring freshet with an influx of meltwater in a presumably more physically based way. The original configuration of the model increased the evaporation rate to account for the excess of water. The modified configuration allowed the excess of meltwater to contribute to ponded storage at the surface of the soil, which instead infiltrated the soil and later contributed to runoff.

The objective of the study, to introduce and to use CRANE to test a more physically based representation of the infiltration of meltwater into freezing, thawing, or refreezing soils was generally met, based on preliminary results. However, the actual contribution of the algorithm cannot be definitively commented upon, given the poor overall performance of the model in this particular modelling scenario. An improved parameterization of the model would have to be made before any further assessment. Furthermore, an improved parameterization of the model could be achieved by linking the model to a calibration utility; this type of link could be facilitated by CRANE.

The benefit to using CRANE in this type of comparative analysis is its ability to manage the workspace, and its ability to use its own configuration files to create the input files for the model that have strict formatting requirements. Contrasting to an approach of copying and pasting files, CRANE updates the fields in an existing file, requiring no need to replicate the file or the entire folder structure. Contrasting to a batch file scripting approach, CRANE requires no knowledge of the scripting language. Without its user interface, knowledge of the framework might be required to understand its configurations files and their syntax; however, the elements in these files would be suggested, auto-completed, and validated, using an XML editor.

Chapter 5
Conclusions and Recommendations

The primary objective of this thesis was to present an intuitive software framework for building integrated modelling systems that would link to existing simulation models without requiring modification to their programming code. CRANE was presented as a flexible framework for linking simulation models and model support tools to form integrated modelling systems, evaluated using the scientific workflow approach.

CRANE demonstrated the ability to configure a simulation model by interacting with its text-based input files. Using a unique abstraction approach, it identifies fields and replaces values in the file, supporting parameter files that use either free- or fixed-formatting.

To develop CRANE, programming requirements were determined based upon existing frameworks and publications. The framework was required to use object-oriented programming techniques; provide separation between its own code and the code of the simulation models or modelling support tools; use sustainable programming techniques to allow its use beyond a single case study or simulation workflow; and provide user-friendly features. The framework was developed in a way that each of these requirements were satisfied.

Similar to Kepler, Taverna, and HydroDesktop, CRANE provides programming interfaces to connect with data in various formats, which includes Microsoft Access and Excel files, SQLite databases, and text files. Similar to HydroPlatform, it can connect to fully formed simulation models via their input and output files. Similar to Kepler, Taverna, OMS, and OUI, it can also connect to the programming code of models, if written in one of the .NET programming languages, which includes Visual Basic .NET and C#. In addition, like HydroDesktop, its programming interfaces can be used to build new plugins to enhance its own feature base and connect to new types of simulation models or data. Finally, and similar to Kepler, the core processing code in CRANE is separated from its user interface to minimize dependencies on code that could potentially limit its longevity, as well as to improve the portability of its core features to other operating systems. CRANE, similar to OUI, also uses structured, human-readable configuration files.

CRANE further improves upon this feature set with its use of a proper schema definition that, with an editor, can automatically formulate the required structure and elements of its configuration files with minimal effort on the part of the user. It also provides a user interface to graphically create elements of the configuration files, without requiring coding on the part of the user. This makes it a useful tool in teaching and training environments.

Its strength is its ability to connect to legacy simulation models, whose code cannot be modified, through input files. The flexibility of the framework allows it theoretically to connect to a variety of simulation models and modelling support tools across various scientific and engineering applications. Its ability to save simulation workflows and configuration information in a structured and transparent format means that whole experiments and simulation modelling scenarios can be saved and distributed among collaborators.

The framework consists of two components: an engine and a user interface. The engine interprets the requirements of the simulation models and modelling tools and facilitates the flow of data between the various components in the simulation workflow. It was written using an older specification of the C# programming language with lesser requirements so that it could be compiled with more distributions of the .NET framework, and potentially across operating systems. The user interface provides a familiar graphic interface through which the engine can be configured and monitored during the evaluation of the simulation workflow.

A case study demonstrated the basic use of the framework as a workspace management tool. Unfortunately, it did not highlight its more advanced features. However, because CRANE can extend beyond a single case study, simulation model, or modelling scenario, these features could possibly be demonstrated by future case studies.

The evaluation of a simulation workflow is a non-trivial task. It requires more than programming knowledge, as the text file requirements of many legacy models are substantial, and an efficient management of the workspace is required to translate files and communicate data between connected simulation models and other types of modelling tasks.

Traditionally, shell scripts might be used to facilitate this exchange. An experienced programmer or data analyst would be able to develop these types of scripts in a matter of days; however, more time might be required to find and fix oversights or errors. In addition, they would have to create code and/or additional scripts to translate and communicate data between the connected tasks, which would take more time, potentially upward of weeks. This type of setup might also include nested scripts, environment variables, or similar scripting mechanisms, that might not be portable, and would be hard for another user to follow. By contrast, a similarly experienced user would be able to setup and evaluate a simulation workflow using CRANE in less than a day, and would be able to create new tasks and programming code for the framework in a matter of days. Most important, code can be transferred to others and the transparent, structured syntax of the simulation workflow would be easy for another user to follow.

In summary, CRANE was presented as a flexible framework for building integrated modelling systems using the scientific workflow approach. It is able to connect to simulation models by a unique method of field abstraction on text files, supporting both free- and fixed-formatting. It adopts object-oriented programming techniques and is written using a relatively portable, modern, and flexible programming language. It supports plugins that can be updated and enhanced independent of the framework, and has an intuitive interface with many user-friendly features. Its flexibility and the generic structures of its programming interfaces, should allow it to interoperate with a variety of simulation models and modelling support tools.

While the framework is flexible, it has yet to be applied to a variety of case studies or compiled on other operating systems. As this is a preliminary introduction of the framework, additional and different types of case studies are recommended, the results from which would identify areas where the framework can continue to be developed and enhanced. The framework would benefit from the integration of more modelling support tools, so that they can be used in workflows with legacy simulation models. Nevertheless, CRANE is an excellent tool for first-time model users, and is particularly suited to teaching environments, training seminars, and automated workflows.

References

Altintas, I, Berkley, C, Jaeger, E, Jones, M, Ludäscher, B & Mock, S 2004, 'Kepler: an extensible system for design and execution of scientific workflows', *16th International Conference on Scientific and Statistical Database Management*, IEEE.

Ames, DP, Horsburgh, JS, Cao, Y, Kadlec, J, Whiteaker, T & Valentine, D 2012, 'HydroDesktop: Web services-based software for hydrologic data discovery, download, visualization, and analysis', *Environmental Modelling and Software*, vol 37, pp. 146-156.

Argent, RM 2004, 'An overview of model integration for environmental applications--components, frameworks and semantics', *Environmental Modelling & Software*, vol 19, pp. 219-234.

Babendreier, JE & Castleton, KJ 2005, 'Investigating uncertainty and sensitivity in integrated, multimedia environmental models: tools for FRAMES-3MRA', *Environmental Modelling and Software*, vol 20, pp. 1043-1055.

Banta, ER, Hill, MC, Poeter, E, Doherty, JE & Babendreier, J 2008, 'Building model analysis applications with the Joint Universal Parameter IdenTification and Evaluation of Reliability (JUPITER) API', *Computers & Geosciences*, vol 34, pp. 310-319.

Banta, ER, Poeter, EP, Doherty, JE & Hill, MC 2006, 'JUPITER: Joint Universal Parameter IdenTification and Evaluation of Reliability -- An Application Programming Interface (API) for Model Analysis', U.S. Geological Survey, Denver, Colorado.

Berkley, C, Bowers, S, Jones, M, Ludäscher, M & Tao, J 2005, 'Incorporating semantics in scientific workflow authoring', *Proceedings of the 17th international conference on Scientific and statistical database management*, Lawrence Berkeley Laboratory.

Beven, K & Binley, A 1992, 'The future of distributed models--Model calibration and uncertainty prediction', *Hydrological Processes*, vol 6, pp. 279-298.

Box, P 2010, 'Hydrologists workbench: A governance model for scientific workflow environments', *2010 International Congress on Environmental Modelling and Software Modelling for Environment's Sake*, International Environmental Modelling and Software Society (iEMSs), Ottawa, Canada.

Branger, F, Braud, I, Debionne, S, Viallet, P, Dehotin, J, Henine, H, Nedelec, Y & Anquetin, S 2010, 'Towards multi-scale integrated hydrological models using the LIQUID framework. Overview of concepts and first application examples.', *Environmental Modelling and Software*, vol 25, no. 12, pp. 1672-1681.

Brooks, C, Lee, EA, Liu, X, Neuendorffer, S, Zhao, Y & Zheng, H 2007, 'Heterogeneous Concurrent Modeling and Design in Java (Volume 1: Introduction to Ptolemy II)', Technical Report No. UCB/EECS-2007-7, Electrical Engineering and Computer Sciences, University of California at Berkeley.

Burn, DH, Fan, L & Bell, G 2008, 'Identification and quantification of streamflow trends on the Canadian Prairies', *Hydrological sciences*, vol 53, no. 3, pp. 538-549.

Castronova, AM, Goodall, JL & Ercan, MB 2013, 'Integrated modeling within a Hydrologic Information System: An OpenMI based approach', *Environmental Modelling and Software*, vol 39, pp. 263-273.

Cuddy, SM & Fitch, P 2010, 'Hydrologists workbench -- A hydrological domain workflow toolkit', *2010 International Congress on Environmental Modelling and Software Modelling for Environment's Sake*, International Environmental Modelling and Software Society (iEMSs), Ottawa, Canada.

David, O, Ascough II, JC, Leavesley, GH & Ahuja, L 2010, 'Rethinking modeling framework design: Object Modeling System 3.0', *2010 International Congress on Environmental Modelling and Software Modelling for Environment's Sake*, International Environmental Modelling and Software Society (iEMSs), Ottawa, Canada.

Davison, B, Pohl, S, Domes, P, Marsh, P, Pietroniro, A & MacKay, M 2006, 'Characterizing snowmelt variability in a land-surface-hydrologic model', *Atmosphere-Ocean*, vol 44, no. 3, pp. 271-287.

Dehotin, J & Braud, I 2008, 'Which spatial discretization for distributed hydrological models? Proposition of a methodology and illustration for medium to large-scale catchments', *Hydrology and Earth System Sciences*, vol 12, pp. 679-796.

Dingman, SL 2008, *Physical Hydrology*, 2nd edn, Waveland Press, Inc., Long Grove, IL.

Doherty, J 2005, 'PEST: Software for Model-Independent Parameter Estimation', *Watermark Numerical Computing*.

Ellis, CR, Pomeroy, JW, Brown, T & MacDonald, J 2010, 'Simulation of snow accumulation and melt in needleleaf forest environments', *Hydrology and Earth System Sciences*, vol 14, no. 6, pp. 925-940.

Fang, X, Pomeroy, JW, Westbrook, CJ, Guo, X, Minke, AG & Brown, T 2010, 'Prediction of snowmelt derived streamflow in a wetland dominated prairie basin', *Hydrology and Earth System Sciences Discussions*, vol 7, no. 1, pp. 1103-1141.

Görlach, K, Sonntag, M, Karastoyanova, D, Leymann, F & Reiter, M 2011, 'Conventional workflow technology for scientific simulation', *Guide to e-Science*, pp. 323-352.

Granger, RJ, Gray, DM & Dyck, GE 1984, 'Snowmelt infiltration to frozen Prairie soils', *Canadian Journal of Earth Sciences*, vol 21, no. 6, pp. 669-677.

Gray, DM, Toth, B, Zhao, L, Pomeroy, JW & Granger, RJ 2001, 'Estimating areal snowmelt infiltration into frozen soils', *Hydrological Processes*, vol 15, pp. 2095-3111.

Gregersen, J, Gijsbers, P & Westen, S 2007, 'OpenMI: Open modelling interface', *Journal of Hydroinformatics*, vol 9, no. 3, pp. 175-191.

Harbaugh, AW 2005, 'MODFLOW-2005, The U.S. Geological Survey modular ground-water model--The ground-water flow process', USGS Techniques and Methods 6-A16, U.S. Geological Survey, Reston, Virginia.

Harou, JJ, Pinte, D, Tilmant, A, Rosenberg, DE, Rheinheimer, DE, Hansen, K, Reed, PM, Reynaud, A, Medellin-Azuara, J, Pulido-Velazquez, M, Matrosov, E, Padula, S & Zhu, T 2010, 'An open-source model platform for water management that links models to a generic user-interface and data-manager', *2010 International Congress on Environmental Modelling and Software Modelling for Environment's Sake*, International Environmental Modelling and Software Society (iEMSs), Ottawa, Canada.

He, Y, Bárdossy, A & Zehe, E 2011, 'A review of regionalisation for continuous streamflow simulation', *Hydrology and Earth System Sciences*, vol 15, no. 11, pp. 3539-3553.

Hillyer, C, Bolte, J, van Evert, F & Lamaker, A 2003, 'The ModCom modular simulation system', *European Journal of Agronomy*, vol 18, no. 3, pp. 333-343.

Hinkel, J 2005, 'DIVA: an iterative method for building modular integrated models', *Advances in Geosciences*, vol 4, pp. 45-50.

Kouwen, N 1988, 'WATFLOOD: a Micro-Computer Based Flood Forecasting System Based on Real-Time Weather Radar', *Canadian Water Resources Journal*, vol 13, no. 1, pp. 62-77.

Kouwen, N 1998, 'WATFLOOD users manual', Water Resources Group, University of Waterloo, Waterloo, ON.

Kralisch, S, Krause, P & David, O 2005, 'Using the object modeling system for hydrological model development and application', *Advances in Geosciences*, vol 4, no. 4, pp. 75-81.

Krause, P, Boyle, DP & Bäse, F 2005, 'Comparison of different efficiency criteria for hydrological model assessment', *Advances in Geosciences*, vol 5, pp. 89-97.

Leavesley, GH, Markstrom, SL, Brewer, MS & Viger, RJ 1996, 'The modular modeling system (MMS)--The physical process modeling component of a database-centered decision support system for water and power management', *Water, Air, and Soil Pollution*, vol 90, no. 1, pp. 303-311.

Lloyd, W, David, O, Ascough II, JC, Rojas, KW, Carlson, JR, Leavesley, GH, Krause, P, Green, TR & Ahuja, LR 2011, 'Environmental modeling framework invasiveness: Analysis and implications', *Environmental Modelling and Software*, vol 26, no. 10, pp. 1240-1250.

Ludäscher, B, Altintas, I, Berkley, C, Higgins, D, Jaeger, E, Jones, M, Lee, EA, Tao, J & Zhao, Y 2006, 'Scientific workflow management and the Kepler system', *Concurrency and Computation: Practice and Experience*, vol 18, no. 10, pp. 1039-1065.

Manitoba Conservation, Saskatchewan Water Corporation & Environment Canada 2000, 'Upper Assiniboine River Basin Study', Main Report, Manitoba Conservation, Winnipeg, MB.

Markstrom, SL & Koczot, KM 2008, 'User's Manual for the Object User Interface (OUI): An Environmental Resource Modeling Framework', Open-File Report 2008-1120, U.S. Geological Survey, Reston, Virginia.

Marshall, IB & Schut, P 1999, 'A national ecological framework for Canada', Eastern Cereal and Oilseed Research Centre (ECORC), Research Branch, Agriculture and Agri-Food Canada.

Matott, LS 2005, *Ostrich: an Optimization Software Tool, Documentation and User's Guide, Version 1.6*, University at Buffalo, Buffalo, NY.

Matott, LS, Babendreier, JE & Purucker, ST 2009, 'Evaluating uncertainty in integrated environmental models: A review of concepts and tools', *Water Resources Research*, vol 45, no. 6.

McLaughlin, D & Townley, LR 1996, 'A reassessment of the groundwater inverse problem', *Water Resources Research*, vol 32, no. 5, pp. 1131-1161.

McPhillips, T, Bowers, S, Zinn, D & Ludäscher, B 2009, 'Scientific workflow design for mere mortals', *Future Generation Computer Systems*, vol 25, no. 5, pp. 541-551.

Michener, W, Beach, J, Bowers, S, Downey, L, Jones, M, Ludäscher, B, Pennington, D, Rajasekar, A, Romanello, S, Schildhauer, M, Vieglais, D & Zhang, J 2005, 'Data integration and workflow solutions for ecology', *Data integration in the life sciences*, pp. 321-324.

Nash, J & Sutcliffe, JV 1970, ' River flow forecasting through conceptual models part I—A discussion of principles', *Journal of Hydrology,* vol 10, no. 3, pp. 282-290.

Oinn, T, Addis, M, Ferris, J, Marvin, D, Senger, M, Greenwood, M, Carver, T, Glover, K, Pocock, MR, Wipat, A & Li, P 2004, 'Taverna: a tool for the composition and enactment of bioinformatics workflow', *Bioinformatics*, vol 20, no. 17, pp. 3045-3054.

Oinn, T, Greenwood, M, Addis, M, Alpdemir, MN, Ferris, J, Glover, K, Goble, C, Goderis, A, Hull, D, Marvin, D, Li, P, Lord, P, Pocock, MR, Senger, M, Stevens, R, Wipat, A & Wroe, C 2006, 'Taverna: lessons in creating a workflow environment for the life sciences', *Concurrency and Computation: Practice and Experience*, vol 18, no. 10, pp. 1067-1100.

Perrin, C, Michel, C & Andréassian, V 2003, 'Improvement of a parsimonious model for streamflow simulation', *Journal of Hydrology,* vol 279, no. 1, pp. 275-289.

Pietroniro, A, Fortin, V, Kouwen, N, Neal, C, Turcotte, R, Davison, B, Verseghy, D, Soulis, ED, Caldwell, R, Evora, N & Pellerin, P 2007, 'Development of the MESH modelling system for hydrological ensemble forecasting of the Laurentian Great Lakes at the regional scale', *Hydrology and Earth System Sciences,* vol 11, no. 4, pp. 1279-1294.

Poeter, EP & Hill, MC 1998, *Documentation of UCODE: A computer code for universal inverse modeling,* US Geological Survey.

Poitras, V, Sushama, L, Seglenieks, F, Khaliq, MN & Soulis, E 2011, 'Projected changes to streamflow characteristics over western Canada as simulated by the Canadian RCM', *Journal of Hydrometeorology*, vol 12, no. 6, pp. 1395-1413.

Pomeroy, JW, Gray, DM, Brown, T, Hedstrom, NR, Quinton, WL, Granger, RJ & Carey, SK 2007, 'The cold regions hydrological model: a platform for basing process representation and model structure on physical evidence', *Hydrological Processes*, vol 21, no. 19, pp. 2650-2667.

Princz, D 2009, 'Evaluating two optimization schemes and assessing the automated calibration process of ParaMESH on the Baker Creek watershed using MESH', Department of Civil and Environmental Engineering, University of Waterloo, University of Waterloo, Waterloo, Ontario, Canada.

Princz, D & Matott, LS 2010, 'Calibrating Environmental Models using ParaMESH', *2010 International Congress on Environmental Modelling and Software*, International Environmental Modelling and Software Society (iEMSs), Ottawa, Canada.

Rahman, JM, Seaton, SP, Perraud, JM, Hotham, H, Verrelli, DI & Coleman, JR 2003, 'It's TIME for a new environmental modelling framework', *MODSIM 2003 International Congress on Modelling and Simulation*, Modelling and Simulation Society of Australia and New Zealand, Townsville, Australia.

Refsgaard, JC 1997, 'Parameterisation, calibration and validation of distributed hydrological models', *Journal of Hydrology*, vol 198, no. 1-4, pp. 69-97.

Shretha, RR, Dibike, YB & Prowse, TD 2012, 'Modelling of climate-induced hydrologic changes in the Lake Winnipeg watershed', *Journal of Great Lakes Research*, vol 38, pp. 83-94.

Simmhan, Y, Barga, R, van Ingen, C, Lazowska, E & Szalay, A 2009, 'Building the trident scientific workflow workbench for data management in the cloud', *Third International Conference on Advanced Engineering Computing and Applications in Science*, IEEE.

Soil Landscapes of Canada Working Group 2006, 'Soil landscapes of Canada v3. 1', Agriculture and Agri-Food Canada, Ottawa, ON.

Soulis, ED, Kouwen, N, Pietroniro, A, Seglenieks, FR, Snelgrove, KR, Pellerin, P, Shaw, DW & Martz, LW 2005, 'A framework for hydrological modelling in MAGS', *Prediction in Ungauged Basins: Approaches for Canada's Cold Regions*, Canadian Water Resources Association, Cambridge, Ont., Papers presented at a conference held in Yellowknife, N.W.T., Mar. 6, 2004.

Soulis, ED, Snelgrove, KR, Kouwen, N, Seglenieks, F & Verseghy, DL 2000, 'Towards closing the vertical water balance in Canadian atmospheric models: Coupling of the land surface scheme CLASS with the distributed hydrological model WATFLOOD', *Atmosphere-Ocean*, vol 38, no. 1, pp. 251-269.

Spence, C, Burn, DH, Davison, B, Hutchinson, D, Ouarda, TBMJ, St-Hilaire, A, Weber, F & Whitfield, PH 2013, 'A Canadian viewpoint on data, information and uncertainty in the context of prediction in ungauged basins', *Hydrology Research*, vol 44, no. 3, pp. 419-429.

Tolson, BA & Shoemaker, CA 2007, 'Dynamically dimensioned search algorithm for computationally efficient watershed model calibration', *Water Resources Research*, vol 43, no. 1.

Verseghy, DL 2000, 'The Canadian land surface scheme (CLASS): Its history and future', *Atmosphere-Ocean*, vol 38, no. 1, pp. 1-13.

Verseghy, D 2008, 'The Canadian Land Surface Scheme: Technical Documentation, Version 3.4', Climate Research Division, Science and Technology Branch, Environment Canada, Environment Canada.

Villa, F, Athanasiadis, IN & Rizzoli, AE 2009, 'Modelling with knowledge: A review of emerging semantic approaches to environmental modelling', *Environmental Modelling and Software*, vol 24, no. 5, pp. 577-587.

Voinov, A & Cerco, C 2010, 'Model integration and the role of data', *Environmental Modelling and Software*, pp. 965-969.

Whelan, G, Castelton, KJ, Buck, JW, Hoopes, BL, Pelton, MA, Strenge, DL, Gelston, GM & Kickert, RN 1997, 'Concepts of a framework for risk analysis in multimedia environmental systems', Pacific Northwest National Laboratory, Richland, Wash.

Zhang, Y, Carey, SK, Quinton, WL, Janowicz, JR, Pomeroy, JW & Flerchinger, GN 2010, 'Comparison of algorithms and parameterisations for infiltration into organic-covered permafrost soils', *Hydrology and Earth System Sciences*, vol 14, no. 5, pp. 729-750.

Zhao, L & Gray, DM 1999, 'Estimating snowmelt infiltration into frozen soils', *Hydrological Processes*, vol 13, pp. 1827-1842.

Appendix A
Sample Configuration

CRANE supports configuration in the eXtensible Markup Language (XML) to define the structure of the input text files of a simulation model and/or to activate and configure the tasks of a simulation workflow. The structure and elements that are allowed in this configuration are enforced by an XML schema definition (XSD).

The root element of a configuration file for CRANE is the `<Crane>` element. This element contains a version number to remain compatible with future versions of CRANE in case the schema definition should change. The main children of this element are the `<RegisteredObjects>` and `<TaskWorkflow>` elements. Children of the `<RegisteredObjects>` element are any objects that correspond to classes that implement the programming interfaces of CRANE; these include the 'OperableComponents' described in Section 3.2.1. Children of the `<TaskWorkflow>` element are tasks to be executed in the evaluation of the simulation workflow.

The first configuration sample, below, corresponds to the definition of the parameter file modified by CRANE to run both versions of the simulation model in the case study presented in Chapter 4. The second configuration sample, which follows, corresponds to the simulation workflow of the same case study.

Sample XML for the ‘MESH_input_run_options.ini’ text file with a fixed-format field named ‘_FROZENSOILINFILFLAG_val’ tied to the ‘_FROZENSOILINFILFLAG_defn’ field definition. The field must contain an integer value that can only equal ‘0’ or ‘2’.

```
<?xml version="1.0" encoding="utf-8"?>
<Crane xmlns="http://www.eng.uwaterloo.ca/~dprincz/crane/xmlns/CraneNamespace_2012-v-1-1.xsd"
       version="1.1">
  <RegisteredObjects>
    <TextFile name="MESH_input_run_options_ini" path="MESH_input_run_options.ini">
      <RegisteredInformation name="_FROZENSOILINFILFLAG_defn">
        <VariableConstraint>
          <TypeOf type="Integer"/>
          <DefaultValue value="0"/>
          <DiscreteValue>
            <DiscreteValue value="0"/>
            <DiscreteValue value="2"/>
          </DiscreteValue>
        </VariableConstraint>
      </RegisteredInformation>
      <RegisteredInformation name="_OUTFILES_defn">
        <VariableConstraint>
          <TypeOf type="String"/>
          <DefaultValue value="."/>
        </VariableConstraint>
      </RegisteredInformation>
      <TextSection>
        <StaticTextField staticValue="MESH Input, Run Options" toLineEnd="true"/>
        <StaticTextField staticValue="##### Control Flags #####" toLineEnd="true"/>
        <StaticTextField staticValue="----#" toLineEnd="true"/>
        <StaticTextField staticValue="   14 # Number of Control Flags" toLineEnd="true"/>
        <StaticTextField staticValue="HOURLYFLAG            60" toLineEnd="true"/>
        <StaticTextField staticValue="BASINRAINFLAG          1 hf=60" toLineEnd="true"/>
        <StaticTextField staticValue="BASINLONGWAVEFLAG      1 hf=60" toLineEnd="true"/>
        <StaticTextField staticValue="BASINSHORTWAVEFLAG     1 hf=60" toLineEnd="true"/>
        <StaticTextField staticValue="BASINTEMPERATUREFLAG   1 hf=60" toLineEnd="true"/>
        <StaticTextField staticValue="BASINWINDFLAG          1 hf=60" toLineEnd="true"/>
        <StaticTextField staticValue="BASINPRESFLAG          1 hf=60" toLineEnd="true"/>
        <StaticTextField staticValue="BASINHUMIDITYFLAG      1 hf=60" toLineEnd="true"/>
        <StaticTextField staticValue="SHDFILEFLAG            1" toLineEnd="true"/>
        <StaticTextField staticValue="AUTOCALIBRATIONFLAG    1" toLineEnd="true"/>
        <NamedFixedTextField name="_FROZENSOILINFILFLAG_val"
                             fieldLength="4" labelLength="20" toLineEnd="true">
          <RegisteredInformation referId="_FROZENSOILINFILFLAG_defn"/>
        </NamedFixedTextField>
        <StaticTextField staticValue="##### Output Grid Points #####" toLineEnd="true"/>
        <StaticTextField staticValue="----#" toLineEnd="true"/>
        <StaticTextField staticValue="    0 # Number of CLASSOUT folders" toLineEnd="true"/>
        <StaticTextField staticValue="---------#" toLineEnd="true"/>
        <StaticTextField staticValue=" #1 Grid Number" toLineEnd="true"/>
        <StaticTextField staticValue=" #2 Output GRU" toLineEnd="true"/>
        <StaticTextField staticValue=" #3 Grid Point Output Directory" toLineEnd="true"/>
        <StaticTextField staticValue="##### Output Directory #####" toLineEnd="true"/>
        <StaticTextField staticValue="---------#" toLineEnd="true"/>
        <FixedTextField name="_OUTFILES_val" fieldLength="10">
          <RegisteredInformation referId="_OUTFILES_defn"/>
        </FixedTextField>
        <StaticTextField staticValue=" # Output folder for Basin Output" toLineEnd="true"/>
        <StaticTextField staticValue="##### Simulation Run Times #####" toLineEnd="true"/>
        <StaticTextField staticValue="---#---#---#---#" toLineEnd="true"/>
        <StaticTextField staticValue="2002 152   0   0 # Sim. Start" toLineEnd="true"/>
        <StaticTextField staticValue="2015   1   0   0 # Sim. Stop" toLineEnd="true"/>
      </TextSection>
    </TextFile>
  </RegisteredObjects>
</Crane>
```

Sample XML for a task workflow to run two configurations of a simulation model named 'MESH_1_3_mod'. Parameters are read from a file defined by 'MESH_input_run_options_ini', a parameter value named '_FROZENSOILINFILFLAG_val' is sought and its value changed to '0', the parameter file is saved, the model is run, and various output files are moved/renamed. This process repeats except the parameter value is changed to '2'.

```
<?xml version="1.0" encoding="utf-8"?>
<Crane xmlns="http://www.eng.uwaterloo.ca/~dprincz/crane/xmlns/CraneNamespace_2012-v-1-1.xsd"
       version="1.1">
  <RegisteredObjects>
    <TextFileBasedExecutableModel name="MESH_1_3_mod" executablePath="sa_mesh.exe">
      <TextFile name="MESH_input_run_options_ini" path="MESH_input_run_options.ini"/>
    </TextFileBasedExecutableModel>
  </RegisteredObjects>
  <TaskWorkflow>

    <!--FROZENSOILINFILFLAG_0-->
    <ParameterRead refer="MESH_input_run_options_ini"/>
    <ChangeTextField>
      <Component refer="MESH_input_run_options_ini">
        <TextField refer="_FROZENSOILINFILFLAG_val" value="0"/>
      </Component>
    </ChangeTextField>
    <ParameterSave refer="MESH_input_run_options_ini"/>
    <Execute refer="MESH_1_3_mod"/>
    <MoveFile refer="Metrics_Out.txt" to="Metrics_Out_FROZENSOILINFILFLAG_0.txt"/>
    <MoveFile refer="MESH_output_streamflow.csv"
              to="MESH_output_streamflow_FROZENSOILINFILFLAG_0.csv"/>
    <MoveFile refer="Basin_average_water_balance.csv"
              to="Basin_average_water_balance_FROZENSOILINFILFLAG_0.csv"/>

    <!--FROZENSOILINFILFLAG_2-->
    <ParameterRead refer="MESH_input_run_options_ini"/>
    <ChangeTextField>
      <Component refer="MESH_input_run_options_ini">
        <TextField refer="_FROZENSOILINFILFLAG_val" value="2"/>
      </Component>
    </ChangeTextField>
    <ParameterSave refer="MESH_input_run_options_ini"/>
    <Execute refer="MESH_1_3_mod"/>
    <MoveFile refer="Metrics_Out.txt" to="Metrics_Out_FROZENSOILINFILFLAG_2.txt"/>
    <MoveFile refer="MESH_output_streamflow.csv"
              to="MESH_output_streamflow_FROZENSOILINFILFLAG_2.csv"/>
    <MoveFile refer="Basin_average_water_balance.csv"
              to="Basin_average_water_balance_FROZENSOILINFILFLAG_2.csv"/>
  </TaskWorkflow>
</Crane>
```