Development of a Web-Based Modeling System Using Metadata Concepts and Databases

A Thesis

Submitted to the Faculty

of

Drexel University

by

Akm Saiful Islam

in partial fulfillment of the

requirements of the degree

of

Doctor of Philosophy

September 2004

DEDICATIONS

To my wonderful parents:

My father Serajul Islam

And

My mother Shahana Islam

... With love

ACKNOWLEDGEMENTS

I am delighted to express my profound gratitude to Dr. Michael Piasecki for his supervision, continuous guidance and encouragements during all this years. Indeed his help to me has been indispensable to the completion of this work.

I would like to thank to National Oceanographic Partnership Program (NOPP) and National Science Foundation (NSF) for their financial support under grant numbers NAG13-0040 and 0412904 respectively.

I would like to express my appreciation to the other members of my doctoral committee: Dr. Richard Weggel, Dr. Xia Lin, Dr. Agami Reddy and Dr. Alen C.W. Lau for their support and encouragements during all these years.

I would like to thank Luis Bermudez for his generous help, valuable suggestions and friendship and good wishes over the years.

Very special thanks go to Bora Beran for many constructive discussions, friendly help and enthusiasm throughout the research work.

Other members of the computational Hydraulics Laboratory at Drexel University, namely Volkan Yargici, Kutay Celebioglu, Yoori Choi and Zafer Defne are also acknowledged for their help and friendship.

I am sincerely thankful to my friends, Mashfiqus Salehin, Shah Alam Khan, Nasir Uddin, Azizul Islam, Lei Lou, Songtao Liao, Abu Syed Nasim and many others for their love and respect.

I appreciated useful discussions and comments from Stephane Fellah of PCI Geomatics Inc. Hull, QC, Canada during the creation of ontologies.

I would also like to thank Dr. Holger Knublauch of Stanford University who is the developer of Protégé OWL plug-in for his valuable suggestions.

Finally, my deepest and sincerest regards go to my parents, brothers and sisters, who provided me moral support as always. Thanks to Almighty for giving me strength to overcome all the difficulties and problems I faced during the course of this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Development of a Web-Based Modeling System
Using Metadata Concepts and Databases
Akm Saiful Islam
Michael Piasecki, Ph.D.

The execution of hydrodynamic models typically requires the management of large amounts of data and also utilizes considerable computational resources. Powerful and robust servers with extensive storage capabilities are therefore desirable for rapid execution of numerical simulations. Unfortunately, it is not always possible for an individual to afford the necessary facilities whereas a powerful central computer system can be the viable alternative to serve many clients. The simplest way for a client to communicate with the central simulation server is via the internet and through a web browser. This kind of simulation has been classified as web based simulation, or WBS. The main advantages of web based simulation include platform independent access and easy access from virtually anywhere. In this study the formal steps that need to be taken for adapting a legacy hydrodynamic code such that it can be used for large scale applications in a WBS environment is investigated. Standardized description of the hydrodynamic model data (metadata) that has been created using geographical information metadata, e.g. the ISO 19115:2003 standard is introduced. A formal specification of the simulation domain or ontology has been developed to share and retrieve this information unambiguously. Ontologies have been successfully applied in many fields requiring intensive data retrieval, efficient searching, or analyzing the domain knowledge. A simulation ontology is developed, which can be applied for

analyses and future reuse of the simulation domain knowledge. The interface of the WBS environment has been developed based on the commonly used standard *Model-View-Controller (MVC)* architecture, which separates business logic from its presentation.

CHAPTER 1: INTRODUCTION

Demands on the modeling expertise of scientists and engineers in governmental units and consulting companies have increased over recent years as the modeling tasks have become more complex. Much of this increase is founded on the contention that surrounds the detail of the modeling approach, i.e. whether a simple or a more involved detailed approach needs to be chosen to address the (mostly environmental) problem at hand. The contention develops from the potential costs that more stringent water quality control measures demand as a result of higher water quality standards and the increased awareness and demand for a healthy environment. The stakeholder process for both the National Pollution Discharge Elimination System (NPDES) and the Total Maximum Daily Load Program (TMDL) are typical example frameworks within which the modeling approaches are often contested to address a specific water quality problem. As a result, simplified approaches, requiring little modeling expertise, are often not sufficient because, for instance, the chosen model must withstand challenges in court that are posed by the affected stakeholder(s).

For example, many of the state and local agencies find themselves in a position to have to conduct studies based on modeling approaches for which they can supply little or no expertise. Due to funding limitations many of these modeling needs cannot always be subcontracted out to (the few) consulting firms that do have the expertise. As a consequence, many entities find themselves in a position of needing modeling expertise but not being able to sub-contract it out due to limited funds resulting in a disparity between the need for detailed level modeling and expertise available.

To fill some of the gaps and to help alleviate the aforementioned difficulties, almost all modeling packages now contain graphical user interfaces (GUI) to aid the user in operating the numerical model and analyzing modeling results. One of the more popular modeling packages is BASINS (a modeling environment that links surface hydrology, subsurface hydrology, and stream flow models) that, in addition to a regular GUI, provides a multipurpose modeling environment as it allows for inclusion of a number of common modeling engines with data format conversions [1]. While this is a first step towards more integrated modeling environments, it still requires the development of numerical grids and modification (editing) of model specific input files. Also, no modules exist that provide a user with a "smart" interface that eliminates much of the expertise required to operate and run numerical models. As a result, the current status-quo of integrated modeling systems is not sufficient to tackle the disparity between needed ease-of-operation and required expertise.

Simulation models need input data (numerical grid, initial, and boundary conditions) to operate. Yet, each numerical model currently available demands its own specific data format making the interchange and analysis of model inputs and results a tedious task that requires enormous data format conversions. Therefore, a standardized framework to describe input and output data for the model is essential. Metadata, commonly known as "data about data", promises to provide a means to overcome the heterogeneity and inconsistency that exists between models. Metadata frameworks have been established worldwide to provide a solution to significantly eliminate the difficulties that commonly surround the collection, interchange and analysis of data sets. A metadata framework that is specific for operating a numerical model would eliminate the need for

data file format conversions, individual descriptions of numerical model data sets, and allow a large number of users to exchange their model data files freely and without hindrance. Hence, it is hypothesized that it must be possible to identify a meta-standard for the operation of numerical models. Consequently, the first goal of this research is to develop a unified and hierarchical meta-standard for input data for numerical models. Because of the complexity and breadth of numerical models and the vast scope of numerical modeling efforts, this study will restrict to the application of hydrodynamic models as a test-bed for the proposed work.

Once this standard has been developed it can be utilized to actually develop a Meta-language. That is, the standard, which is purely semantic in nature, must be given a syntactic support frame so it becomes a (Meta)-language. One very popular instrument for doing this is the Extensible Markup Language (XML), which has been widely recognized as standard form to publish electronic data and metadata [2]. In essence, it is a platform-independent grammar that can be used to develop markup languages. In addition, it can be used to define a so-called Resource Description Framework (RDF), which will host the definitions and limitations of the semantics (metadata standard) [3]. Other related technologies such as Web Ontology Language (OWL), built on XML, proves to be a better choice for encoding the standard because they permit a much richer table of semantics as well as a more flexible definition of classes and their attributes when compared to XML schema [4].

The concept of metadata and the use of XML as an encoding language lend itself for expanding the approach one step further to include a database. Instead of storing data in an input file and having a plethora of files crowding the file system, a database could

serve as a data repository for all modeling efforts. In fact, it should be possible to utilize the suggested technology to entirely eliminate input files and have the numerical engine use the database directly for data feeds. This would not only make data file input and output formats obsolete, but also would allow the population of the database by users that do not necessarily partake in modeling efforts. For example, a person collecting data in the filed could deposit these data in the database, such that subsequent modeling efforts can directly use these datasets. The only requirement, besides Quality Control and Quality Assurance (QC/QA) would be that the database itself must have an ontology that maps directly to the metadata standard. Hence, this study proposes to incorporate a database system that is directly linked to the metadata standard and can be used to directly and seamlessly feed all necessary data items to the numerical engine.

To make the modeling environment operating platform independent all development will be done in Java language. Java, in conjunction with a web-browser, like Windows Explorer or Netscape, allows for a system that is portable without problems and can be given a level of functionality. Various classes that come with Java JDK 1.4, permit coding that can interface with legacy codes in C/C++ or FORTRAN (Java Native Language Interface, JNI). As a result, execution of models can be performed in various ways such as component based; client-server based, or distributed using Parallel and High Level Architecture (HLA). Java is also rich on classes that allow the development of high level graphical interfaces using the SWING classes (Java ^{TM}, 2002). In addition, the possibility to include Applets into a regular HTML page permits a high level of functionality between the user and a powerful server (for number crunching purposes). The flexibility and power of Java for internet-capable software development is richly

documented, in fact Java has become the globally accepted standard for applications that seek to utilize network capabilities. Therefore, all code development, in fact the entire modeling environment, will be done in Java.

Hence, the focus of this study is to develop a modeling system that is platform-independent that eliminates much of the time consuming effort that accompanies learning the numerical approach, setup and development of the model, as well as operation, and that automatically ensures reasonable results with a greatly reduced demand on the operating knowledge.

1.1. Problem statement

The recent development trend of the numerical models is ignoring some basic needs of the modeling user community. A summary of the problems of the currently available numerical models have are listed below:

1. Lack of sufficient description or metadata of the data sets used in the numerical models.
2. Lack of selection of a suitable metadata standard.
3. Lack of providing guidelines for extending or creating new metadata sets.
4. Lack of selection of suitable metadata publishing languages so that it can be understand by both humans and machines
5. Lack of a framework to publish these metadata which describes data sets of the numerical models.

6. Lack of a client-server based architecture in order to use the capability of powerful machines for simulating large scale model such as simulation of hydrodynamic models.
7. Lack of user friendly tools or graphical user interfaces which can be easily used from the client machine to communicate with server machines.

1.2. Motivation

Numerical models deals with huge amount of input and output data. The proper description of this data is essential to search and exchange of these data. Unfortunately, this essential part of data has been neglected in the numerical modeling community in the past. A very few cases has been found where numerical model uses metadata or supports metadata creation tools. Metadata should be based on certain standard so that it is possible for the users to understand and share metadata among the users. There are several metadata standards or initiatives currently exist. The selection of a certain metadata standard which better fit the need of the numerical model is necessary. At present, there are no such suitable guidelines exist to create metadata based on any particular standard for hydrodynamic models. Moreover, the extension of the metadata for a particular purpose is required and should be provided. This study will focus on the particular needs of hydrodynamic model to describe its data structure. It also focuses on the selection of a certain metadata standard suitable for hydrodynamic models.

One of the problems of traditional hydrodynamic model is that it has designed to run in a standalone machine. Sharing of numerical simulation results among users located different places is not possible for most of the commonly used hydrodynamic models. Numerical model should reside in a central place so that users can easily access

the model and its data. User access to the server can be accomplished by many different ways and the simplest way is to use a web browser. This study intent to develop a system such that users can initiate simulation from their machines and simulation executed in the server machines. Web browser will be the means of this simulation to simplify this client-server based simulation.

One of the useful applications of the web based simulation is to use powerful server machine. For example, world's fastest super computer "Earth Simulator" has more than 5000 processors and can run approximately 500 times faster than an ordinaty computer. It is beneficial to use supercomputer if simulation requires high speed and uses huge amount of data. This study has been motivated to develop a web-based simulation environment as a future generation of simulation to use computational resources of powerful server machines.

Another key problem of traditional modeling system is to access to distributed data sources during the simulation time. Hydrodynamic models use different kinds of data and it is not always possible to find all these datasets in one place or in one server. Using distributed data sources is often desirable and access to these data sources could be essential for real time simulation. However, web-based simulation can easily access to distributed web resources during the simulation. Moreover, it can also help to retrieve and display simulation results almost instantaneously to the remote users.

Finally, development of numerical models is always an expansive and tedious process for a single person or a group of people. On the other hand it can be beneficial if a certain community help on the development and debugging process. In recent years the concept of "community models" has been established to develop a set of modeling tools

which can server the whole community. Web based simulation can be ideal for sharing the model for a certain community placing it in a central powerful server. These community models can be beneficial using a Web-based simulation environment to share the model and data within the community.

1.3. Contribution of the thesis

This study focuses on the development of a web-based simulation system. The following can be listed as the major contributions of this study.

- Implementation of the Geographic Information – Metadata (ISO 19115:2003) standard using Web Ontology Language (OWL);
- Development of a metadata community profile for hydrodynamic models using the ISO 19115:2003 ontology.
- Development of a web-based simulation system for numerical models employing databases.
- Presentation of a case study of the Web Based Simulation (WBS) environment using a two dimensional finite element model for the upper Potomac estuary.

1.4. Organization of this dissertation

The thesis is organized as follows to provide an easier and streamlined reading, each chapter was written as a stand-alone paper with its own abstract, introduction and conclusion.

Chapter 2 presents the implementation of the geographic information metadata (ISO 19115:2003) norm using the Web Ontology Language (OWL). It also discusses the

shortcomings of XML schema and some critical issues that surround the conversion of a Unified Modeling Language (UML) model into an OWL ontology.

Chapter 3 describes the development of a framework for hydrodynamic code and data descriptions using the International Standard Organization, ISO, “Geographic Information Metadata,19115:2003” standard.

Chapter 4 investigates design and development strategy of Web Based Simulation (WBS) environment for hydrodynamic processes. Developing tools and graphical user interfaces for different components of WBS such as metadata, data share and search, data storage of model I/O are also examined. An application of a web based simulation has been studied to simulate flow in the upper Potomac estuary using a two dimensional, vertically averaged finite element model.

CHAPTER 2: IMPLEMENTATION OF THE GEOGRAPHIC INFORMATION – METADATA (ISO 19115:2003) NORM USING THE WEB ONTOLOGY LANGUAGE (OWL)

2.1. Abstract

The International Organization for Standardization (ISO) has published the geographic information metadata norm, ISO 19115:2003, to provide a formal structure for describing digital geographic data. The standard is published using an abstract object model that is implemented via the Unified Modeling Language (UML) for conceptualizing the underlying structure of the norm. As a continuation for further enhancement ISO is currently working on a geographic information metadata implementation specification norm, ISO 19139.3 that seeks to find a new implementation for the UML-based abstract model for the ISO 19115:2003. The current implementation approach favors the use of a XML Schema to represent the 19115 norm in machine readable format. However, the use of XML Schema entails a number of disadvantages and as a result falls short in representing the full potential of the UML based ISO 19115:2003 standard. The use of the Web Ontology Language (OWL), built on XML, proves to be a better choice for encoding the standard because it permits a much richer table of semantics as well as a more flexible definition of classes and their attributes when compared to XML schema. In this study, the shortcomings of XML schema will be discussed. Some critical issues surrounded the conversion of a UML model into an OWL ontology will be pointed out and a number of mapping approaches to overcome some of the incompatibilities that exist between UML and OWL will be suggested.

2.2. Introduction

The International Organization for Standardization (ISO) has published the geographic information metadata standard, ISO 19115:2003, to provide a structure for describing digital geographic data [16]. Although this standard is primarily developed for digital datasets, its applicability can be extended to many other forms of geographic data such as maps, charts, textual documents, and general purpose data. As a continuation of interoperability enhancement among distributed geographic information, ISO is currently working on a geographic information metadata implementation specification, ISO 19139.3 [17]. This new implementation effort will specify a profile using Unified Modeling Language (UML) [18] interpretation and an Extensible Markup Language, XML, Schema [19] for machine readability.

Yet, there is evidence that the use of XML-schema, the Resource Description Framework, RDF [3], or RDF-schema is somewhat insufficient to translate the concepts provided in UML into a fully compatible machine understandable form. These shortcomings will be examined in more detail in the following sections.

XML Schema provides the syntax and grammar to validate documents, which in itself is a desirable feature for automated handling of documents, but the ability of XML Schema to represent semantics for validation and to link concepts across the World Wide Web (WWW) is quite poor [20]. For example, a well formed XML document, that is a document that strictly adheres to the rules of XML, could have a <*watershed*> element which contains elements like <*name*>, <*area*>, <*operatedBy*> and <*outletLocation*>. Using an XML Schema can also specify that a <*watershed*> can have only one <*outletLocation*>. If any <*watershed*> element contains more than one

<*outletLocation*>, a validation check of this XML Schema will yield that this information is syntactically wrong. However, if two instances of a watershed have the same values or entries for all its elements, except for <*outletLocation*> (this could be two different names or two different longitude-latitude coordinates), an XML Schema validation will not reveal that one of the instances is wrong, solely based on the fact that only one <*outletLocation*> is allowed for any watershed. Moreover, it is also difficult to state more facts about a watershed, for example that it is a geographic area or that a watershed is similar to a hydrologic unit. Hence, although XML Schema provides a means to validate the structures of a XML document, it does not permit to capture the semantic aspect (correctness of an entry) of the web document [21] prompting the need to seek for alternatives that better describe inherent semantic relationships.

Another model to better represent semantics is the Resource Description Framework (RDF), which was designed to improve the representation of information about web resources [22]. RDF is a data model that contains both properties and statements. A property can be a specific characteristic, attribute or relation that describes a web resource. A statement consists of three parts: a subject, a predicate and an object. The subject represents a resource which has properties, while a predicate represents the property and, an object the value of that property. A property value can bse either a resource or a literal, e.g. free text. The main difference between a literal and a resource is that literals cannot be the subject or the predicate of a statement. For example, a journal article is a resource, which has a property <*author*> whose values are literals (Figure 2.1). That paper also has a property <*publisher*> which would be this journal. A journal is considered a resource type value because it may have, or be related to, other elements

such as <*subscriber*>, <*number of copies published*>, <*cost of publication*>, etc, which does not permit it to be a literal.

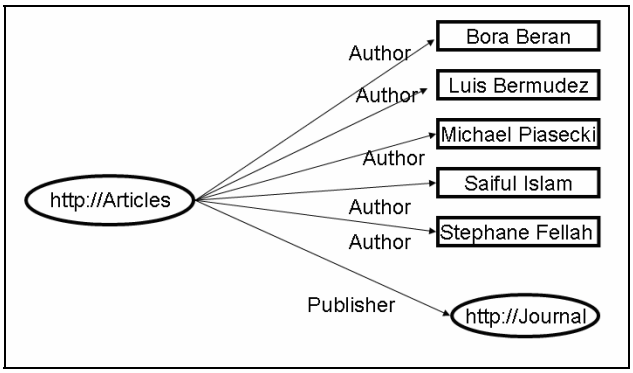


Figure 2.1. RDF graph to describe a journal paper which has five authors and a publisher

As shown in the above example, RDF can provide a model to represent information, however, it provides no mechanism for describing the relationships between these properties and other resources [23]. For example, in the above RDF-outline need that the value of the property <*author*> must reference a person and not a car or a flower. However, the above example clearly demonstrates that <*author*> is a property about which very little is said or demanded, i.e. that its range of acceptable values can only include persons.

The previously outlined shortcomings can be partially overcome by an RDF Schema, which provides a mechanism to describe properties and the relationship between those properties and resources. The use of RDF Schema permits the addition of information about the resource leading to better understanding of what it actually references. It allows the division of resources in to classes and the instance of classes, and

defines the property as an instance of a class [24]. As such, RDF Schema represents a semantic extension of RDF by providing mechanisms to describe relationships between resources. However, it does not provide all the descriptions that are useful for representing sensible meanings among RDF classes and their properties. For example, for a journal paper, it is needed to specify that each author of the paper is a distinct individual, a demand that RDF Schema can not accommodate because it does not permit inclusion of a restriction of this type on the property <*author*>.

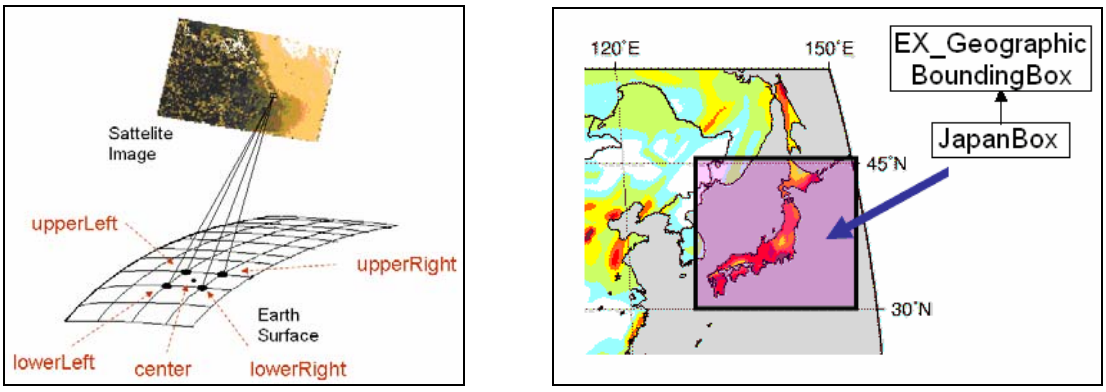
It is evident from the previous discussion that neither XML Schema, RDF, nor RDF Schema fully satisfies the specific needs of a structure to resolve foreseeable semantic problems emerging with the ISO 19115:2003 metadata norm. On the other hand, the advent of the Web Ontology Language (OWL) promises to introduce a tool to overcome the se shortcomings and problems [25]. OWL is part of the ongoing effort of the World Wide Web Consortium (W3C) to work towards the Semantic Web which is defined as a future vision of the web by building machine understandable meaningful form of web resources [26]. OWL enhances RDF Schema by adding more vocabulary to describe properties and classes. For example, the *owl:differentFrom* property of OWL can define that two authors of a journal paper are distinct individuals. Moreover, other terms such as the maximum cardinality of the <*publisher*> property that can be set to unity to indicate that an article can not be published in more than one journal. Hence, the use of OWL to formally specify in an ontology the relationship between elements (classes) and their properties (attributes) promises to provide adequate means through which the full potential of the ISO 19115 :2003 norm can be utilized.

In this study an ontology has been created for geographic metadata that implements the norm in a machine-understandable format. In the next section, the application of a geographic ontology is discussed with useful examples. Section 2.3, discusses the similarities and dissimilarities when attempting to map a UML model into an OWL ontology. Section 2.4 describes in more detail the mapping process and conventions necessary when mapping the ISO 19115:2003 UML model to the OWL ontology.

2.3. Ontology Concept

Ontologies can be applied in many ways, e.g. for detecting inconsistencies, performing validations, extending metadata, ensuring interoperability, and for validation of information integrity and hierarchy [44]. To better understand how ontologies in OWL can achieve the above objectives, some small examples are outlined in the following paragraphs.

A simple example how OWL can be used to detect possible inconsistencies, consider a small subset of geographic metadata. For example, as defined in ISO 19115:2003, the orientation of a point in a pixel is permitted in only five possible ways: center, lower left, lower right, upper right, and upper left, Figure 2.2 (a). This concept can be represented in an ontology by restricting the range of the property *pointInPixel* to a class, *MD_PixelOrientationCode*. The class *MD_PixelOrientationCode* can be enumerated by its five instances *center*, *lowerLeft*, *lowerRight*, *upperRight*, and *upperLeft*. If the *pointInPixel* has as value *bottom*, which is not inside the allowed range, a consistency error is reported.



(a) (b)

Figure 2.2. (a) Orientation of the point in a pixel corresponding to the earth location of the pixel for a regularly spaced grid, and (b) New metadata JapanBox is extended from EX_GeographicBoundingBox metadata to locate earthquakes around Japan.

A second example highlights the ability of ontologies to be easily extended to support the completion of geographic information if new metadata elements or vocabulary must be added. For example, a user is seeking to identify earthquakes that occur around Japanese islands can define a new metadata term *JapanBox*, which is an extension from the ISO 19115:2003 metadata element *EX_GeographicBoundingBox* whose property *westBoundLongitude* is 130^0, *eastBoundaLongitude* is 145^0, *southBoundLatitude is* 30^0, and *northBoundLatitude* is 45^0 as shown in Figure 2.2 (b). This newly defined term *JapanBox*, can be reused for similar queries by the user.

Third, ontologies can be used as a bridge for interoperability demands among different metadata standards. For example, ISO 19115:2003 has a metadata element *title* in the *CI_Citation* package, which is similar to the Dublin Core element *title*. Using OWL it is possible to declare an *owl:equivalentProperty* such that both these terms refer to the same thing while XML Schemas and RDF schemas do not provide such mapping

capabilities. One such effort uses a Dublin Core Metadata Ontology to provide meta information for other ontologies in the HealthCyberMap domain [30].

Fourth, ontologies can be useful to validate and verify geographic data. For example, the maximum occurrence of the ISO metadata element *title* in *CI_Citation* is unity. Let us suppose that an instance document of the metadata ontology has two different titles such as “water elevation” and “stage”. As the maximum cardinality restriction is unity for the *title* property, it will be inferred that both “water elevation” and “stage” represent the same thing. In contrast, an XML Schema would indicate a data inconsistency. The above example highlights a significant difference between XML Schema and OWL, i.e., XML Schema provides only syntactic information whereas OWL provides semantic information for a domain.

Fifth, ontologies can be used to define the basis for system integrity. For example, the ISO 19115:2003 *CI_Address* class has properties *deliveryPoint*, *city*, *administrativeArea*, *postalCode*, *country* and *electronicMailAddress* that are intended to describe any contact address. However, the property *electronicMailAddress* has zero-infinite cardinality making it an optional property. In contrast, if it were defined as a new metadata class called *webGroupAddress*, it would demand that instances of this class must have *electronicMailAddress* for each contact, i.e. it is needed to change the cardinality to unity-infinite. OWL permits change of cardinality. The result is that it is now possible to restrict users so they are required to fill the email address while creating an instance of *webGroupAddress*.

Finally, ontologies can improve query results inferring over the hierarchies of contained terms. For example, a geographic information ontology can classify natural

water bodies as stream, river, lake, pond or ocean. Lake can be further classified into saltwater lake and fresh water lake based on salinity. For someone looking for a lake for fishing, an ontology based search could not only provide the list of lakes but present a classification of fresh water lakes and salt water lakes.

The above discussion outlines possible difficulties when developing community-specific metadata sets based on the ISO 19115:2003 and the role OWL could play in overcoming these difficulties. The conceptualization of the ISO 19115:2003 (which is provided in UML) and its conversion into a machine readable framework is not straightforward but fraught with shortcomings and potential pitfalls. Among the three languages described, OWL is the most successful in avoiding these problems. In the following section, some of the problems that persist when attempting to map UML into OWL will be pointed out. Some of the newer concepts employed to date are reviewed.

2.4. Mapping of the UML model and OWL ontology

The basic difference between the concepts of ontology and object oriented modeling is in their motivation [45]. The intent of object oriented modeling is to capture sufficient knowledge for a specific purpose whereas the goal of an ontology is to capture facts or knowledge about a domain. Therefore, an object oriented model should always be an abstraction of an ontology. Object oriented models are typically visualized using UML diagrams that demonstrate the static application structure of the model, the different aspects of dynamic behavior, and the organization and management of the application modules. Because of its popularity and wide-spread use, the idea of using UML as an ontology representation language has recently received more attention by researchers.

There are a number of initiatives addressing domain knowledge in UML. Cranefield [46] attempts to represent a RDF Schema as a UML class and an object diagram using StyleSheet Language for XML Transformations [47]. Basic work converting UML to OWL such as *owl:Class*, *owl:SubClassOf*, *owl:DatatypeProperty* and *owl:ObjectProperty* has been carried out by using XSLT [48]. Kogut et al. [49] suggested that UML provides excellent notation for ontologies by considering: (1) UML as a graphical notation based on many years of experience, (2) UML as an open standard maintained by Object Management Group (OMG), (3) UML has been widely adopted in the software industry, and (4) UML has a standard mechanism for defining extensions for specific application such as ontologies. They pointed out that the Object Management Group (OMG) has built a Meta Object Facility (MOF) to provide metadata and semantics of the UML model. The idea behind the MOF is to provide an abstract language and framework to define a metadata layer for models [50]. Guizzardi et al. [51], proposed to extend UML to represent a knowledge language for a conceptual model. One such extension is *Powertype* a special class whose instances are classes in the UML. Finally, Knublauch [52] demonstrated a concept to map the Protégé (open source knowledge-modeling platform) metadata model into a UML meta-model. The basic idea is to create a metadata model for the Protégé's underlying object-model and then to convert this meta model into XML Metadata Interchange (XMI) [53] using Java Metadata Interface (JMI) [54].

Despite these recent efforts, no standard specification currently exists to express an OWL ontology as a UML model, and vice versa. The OMG group recently sought proposals for a specification consisting of three components: (1) an MOF metamodel for

ontology, defined as ontology definition metamodel (ODM), (2) a UML profile for ontology, and (3) mapping between ODM with UML and OWL [55]. At present, OMG is selecting from four initial submissions. It remains unclear when the recommendation for a specification of a MOF metamodel for OWL can be expected. Hence, similarities and dissimilarities between UML and OWL will be indicated and resolved by suggesting workable definitions and then manually mapping UML into OWL.

2.4.1. Similarities and Differences of UML and OWL Concepts

No reference available that points out the similarities between UML and OWL. Baclawski [56] demonstrated mapping similarities between UML and the Darpa Agent Markup Language (DAML). DAML is a language that can be considered the predecessor of OWL that is based on RDF and RDF Schema to represent ontologies. Hence, this comparison lends as a base to infer similarities for UML and OWL. For example, UML *class* can be expressed in a similar concept using *owl:Class*, UML *association* can be mapped into *owl:ObjectProperty* and UML *attribute* can be expressed as *owl:DatatypeProperty*. On other side, there are some concepts in OWL that cannot be straight forwardly mapped into UML. For example, there is no equivalent concept in UML to express the *owl:unionOf* concept of OWL. A summary of the similarities between UML and OWL concepts has presented in Table 2-1, while a list of significant differences is given in Table 2-2.

Table 2-1: Similarities between UML and OWL concepts

UML concept	**OWL concept**
Package	Ontology
Class	Class
Attributes, Associations	Property
Generalization Relation	Hierarchy
Data Type	Data Type
Object	Instance
Multiplicity Constrains	Restrictions

Table 2-2: Major Incompatibilities of conversion from UML to OWL concepts

Incompatible UML Concept	**OWL Concept**
Associations as second class concept	Property as first class concept
Closed world	Open world (monotonic)
Modularity	Not supported
Behavioral specialization/generalization	Set theoretic subclass/ superclass
Multiple meta levels	Single meta level
Abstract class	Not supported
Scope – private , public and protected	Not supported

Before starting to translate (or map) the metadata UML abstract model into an OWL-based ontology, it is worthwhile to further discuss some of the dissimilarities, i.e. potentially difficult points when mapping UML into OWL.

a. Incompatibility of properties

In OWL properties are defined as a *first-class* concept whereas UML *attributes* and *associations* are not first-class. What this means is that in OWL a property can exist without defining its range and domain, while in UML attributes need to have a range and domain definition. Therefore, in an ontology, every property must have a different name as they are declared independently from classes. On the other hand, different UML attributes and association with same name can exists in a single UML model in different classes. For example, the *language* attribute of ISO 19115:2003 can be found in the *MD_Metadata*, *MD_DataIdentification*, *MD_FeatureCatalogDescription* classes to represent metadata language, data language and feature catalog language, respectively. In contrast it is not possible to create three OWL owl:D*atatypeProperty* in a single ontology with the same name. This must be carefully when considered when mapping UML into OWL and some mapping guidelines will be provided to resolve this issue in the next section.

b. Monotonic worlds

OWL and other knowledge representation systems are monotonic in nature, which means that adding a new fact does not affect the correctness of a previously declared fact [56]. In contrast, UML assumes a closed world, which means that if something is absent or is newly added, it will be assumed as false or non-existent. For example, every metadata instance document can only have a single metadata documentation creation date, which is defined as *dateStamp* element. If any particular instance has two *dateStamp* properties, the UML model will consider this situation as a violation of the maximum cardinality requirements. In contrast, monotonic logic does not imply the same

conclusion, because it is possible that both metadata creation dates are the same, which in turn would establish equality of the dates and as such be consistent.

c. Modularization

The ISO presents metadata in different UML packages aimed at reducing the maintenance effort. It is easier to handle, reuse and maintain a UML model if its different analogous components are bundled into different packages. OWL on the other hand, does not support the package-concept as UML does. Different ontologies can only interact using the OWL *owl:import* element and XML "namespaces", which provide unique names. But unlike UML packages, the *owl:import* element provides no semantic explanation about the use of resources within the imported ontology. To avoid this problem when using OWL, the ontology was not modularized into different packages rather to keep it as a single file.

d. Generalization

While the UML *generalization* and *specialization* relationship can be expressed in similar fashion in OWL using the *sub-class* and *super-class* relationships, there are semantic differences between the *generalization* relationship of UML and the *subclass* relationship of OWL. The *generalization* relationship in UML is *behavioral*, which means that *specialization* can add new methods or attributes thereby overriding the method of the *generalized* class. On the other hand, the *sub-class* relationship of OWL is defined and set as *theoretic*, which means that *sub-class* can restrict the *super-class* without adding any attributes or methods [56]. Despite these semantic differences mapping is important as these two concepts identify and represent parent and children relationship between two classes.

e. Abstract class

UML defines classes that are considered *abstract* for which an instance cannot be created. On the other side, OWL does not support the concept of an *abstract* class, which means that every class in OWL is available for creating a new instance. This incompatibility of mapping cannot be resolved directly and an annotation property can be provided to tag this type of class as *abstract*.

There are several other incompatibilities, all of which will be addressed in the following section, including a suggestion or mechanism for resolving them.

2.5. General rules for transforming the ISO 19115:2003 UML model to OWL ontology

This section will explain in detail the conventions adopted to translate the ISO 19115:2003 UML model into an OWL ontology. Section 3.1 to 3.8 addresses naming conventions, data types, and restriction conditions between the two concepts. In section 3.9 to 3.15, the conversion of different UML stereotypes described in ISO 19115:2003 into OWL ontology will be addressed.

2.5.1. Name / role name

Each metadata entity or metadata element has a name or label which is derived from single or multiple concatenated words. Although metadata element names are typically unique within the entire specification, a metadata element may not be globally unique as other entities (sub-portions of the specification) can contain the same element name. This could be a problem if name was used as identifiers (e.g. *rdf:ID*), which

should be unique throughout the namespace. Fortunately, there are very few elements which have identical names. The following criteria will be used to resolve this problem.

If duplicate elements have the same range, only one element is declared without defining its domain. A class that uses this property could refer to it by creating a restriction.

If duplicate elements have different range, create a new element using its short name. If duplicate element is a code list element, an underscore ("_") is added before its name. As only pairs of duplicate elements exist in the norm, this approach can resolve the conflict successfully.

For example, the metadata element *name* appears in seven metadata entities (ISO 19115:2003 classes), as presented in Table 2-3. To resolve this problem, a data type property *name* was created with range *xs:string* with an open domain for four identical range elements.

```
<owl:DatatypeProperty rdf:ID="name">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

Table 2-3: Conversion of duplicate metadata element name

ISO 19115:2003 Class	***owl:domain***	***owl:range***	***rdf:ID***
MD_ExtendedElement Information	-	*xs:string*	*name*
CI_Series	-	*xs:string*	*name*
RS_ReferenceSystem	*RS_ReferenceSystem*	*RS_Identifier*	*refSysName*
MD_Format	-	*xs:string*	*name*
MD_Application SchemaInformation	*MD_ApplicationSchema Information*	*CI_Citation*	*asName*
MD_Medium	*MD_Medium*	*MD_Medium Name Code*	*medName*
CI_OnlineResource	-	*xs:string*	name

Three *owl:ObjectProperty*: *refSysName, asName, medName* were created for different range elements using their short names. The ranges of the elements are kept and remain the same as that of their associated class.

Codelist element point appears in both code lists MD_CellGeometryCode and MD_GeometricObjectTypeCode. Here, rdf:ID=”point” in MD_CellGeometryCode and rdf:ID =“_point” in MD_GeometricObjectTypeCode is used.

2.5.2. Short name and domain code

The possibility to use short names is a unique feature of the ISO 19115:2003 standard which also suggests its use with the XML or SGML language. On the other

hand, the ISO 19139.3 norm uses the metadata element name as a unique identifier for the XML Schema to increase human readability. An approach that is similar to the ISO 19139.3 specification has suggested to use the metadata element name as *rdf:ID*. However, in order to allow the use of short names as permitted in the UML conceptualization of the ISO 19115:2003, an annotation property is created *_shortname* to document this specific feature of ISO 19115:2003.

<owl:AnnotationProperty rdf:ID="_shortName" />

<owl:Class rdf:ID="MD_Metadata">

<iso:shortName>Metadata</iso:shortName>

</owl:Class>

ISO 19115:2003 defines a special data type class *enumeration* (see also 4.12) whose instances form a list of literal values. *Enumeration* gives a list of well understand values by assuming that all the values of the list are known and that this list cannot be extended in future. To provide support for a more flexible list to which more values can be added in future, ISO 19115:2003 also defines *codelist* which is an open type of *enumeration*. Every element in *codelist* has a name and a domain code. Domain codes are comprised of a 3 digit unique number within the *codelist*. To represent this domain code feature, unlike a *_shortname*, an annotation property *_domainCode* has been created to provide a higher degree of readability for each *codelist* element.

<owl:AnnotationProperty rdf:ID="_domainCode" />

<iso:MD_RestrictionCode rdf:ID="copyright">

<iso:_domainCode>001</iso:_domainCode>

</iso:MD_RestrictionCode>

2.5.3. Definition

The ISO 19115:2003 norm provides the utility of a data directory within which descriptions for a metadata entity or element can be outlined to describe the UML abstract model. These descriptions are quite useful when the need arises to better understand what the element tries to describe. This feature can be mapped into OWL via the *rdfs:comment* property. For example, the definition for the *MD_Metadata* entity can be described as follows;

<owl:Class rdf:ID="MD_Metadata">

<rdfs:comment> root entity which defines metadata about a resource or resources

</rdfs:comment>

</owl:Class>

2.5.4. Obligation/condition

The ISO 19115:2003 defines three types of obligations for documentation of metadata entities or metadata elements which are classified as mandatory, conditional or optional. Mandatory and optional obligations are identified in UML as cardinality associations, that are mapped to an OWL ontology via cardinality restrictions. In OWL, a cardinality restriction is an anonymous class, which applies a restriction on a specific property and is declared as a super class of the class using that property.

2.5.5. Mandatory (M)

Mandatory metadata entities or metadata elements are those which must be documented, i.e. must receive a value if the norm is used for a data-set description. If a class defines a minimum or exact cardinality greater than one for the usage of a property, it is mandatory. A cardinality restriction in OWL is created with the value of that cardinality, and it is applied on the entity (property) and is declared super class of the entity (class) using the property.

If a metadata entity or metadata element has more than a single occurrence and also has the same maximum and minimum occurrences, the same cardinality restrictions will be used.

If the metadata entity or metadata element has different minimum and maximum occurrences, the minimum cardinality restriction will be used.

If the metadata entity or metadata element has a maximum occurrence equal to 1, it will be considered as an *owl:FunctionalProperty*.

2.5.6. Conditional (C)

A conditional element defines an element that is mandatory under certain conditions. This definition is quite difficult to map in OWL and at present any mechanism to map these conditional elements (or rather the conditions defining the mandatory use of this element) to an ontology in OWL is not known. This is one of the shortcomings of the present ontology and requires further investigation. For now, conditions are set as an annotation property *condition*.

```
<owl:AnnotationProperty rdf:ID="condition” />
<owl:ObjectProperty rdf:ID="characterSet”>
<iso:condition>ISO 10646-1 not used ?</iso:condition>
</owl:ObjectProperty>
```

2.5.7. Optional (O)

Optional metadata entities or metadata elements provide a means towards the full documentation of the data but are not necessary. This feature is relatively easy to map into OWL by setting the minimum cardinality restriction of the optional metadata element to zero.

```
<owl:Restriction>
<owl:onProperty>
<owl:DatatypeProperty rdf:about="#metadataStandardName"/>
</owl:onProperty>
<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0
</owl:minCardinality>
</owl:Restriction>
```

2.5.8. Maximum Occurrence

Maximum occurrence indicates the possible maximum number of values that can occur for a metadata element in a metadata entity. In OWL, this can be expressed as a local cardinality restriction on the class.

The rules are given as follows:

a. If the maximum and minimum occurrences of an metadata entity or metadata element are the same, maximum occurrence will be represented by the *owl:cardinality*
b. If a metadata entity or metadata element has a different maximum and minimum occurrence, maximum occurrence will be represented by the *owl:maxCardinality*, and the minimum as *owl:minCardinality* restriction.

```
<owl:Restriction>
<owl:onProperty>
<owl:DatatypeProperty rdf:about="#metadataStandardName"/>
</owl:onProperty>
<owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1
</owl:maxCardinality>
</owl:Restriction>
```

2.5.9. Data Type

The ISO 19115:2003 norm uses a number of data types which are defined in "Geographic Information Conceptual Schema Language (ISO 19103)" norm to provide distinct values for the metadata elements [57]. A similar approach as taken in the ISO 19139.3, i.e. to translate data types into XML base types is used. The conversion lists from the ISO 19115:2003 data type class into XML base types are given in Table 2-4.

Table 2-4: Data type Implementation

ISO 19103 Class	**XML base type**
Binary	xs:base64Binary
Boolean	xs:Boolean
Character	xs:string
CharacterString	xs:string
Date	xs:date
DateTime	xs:dateTime
Decimal	xs:decimal
Integer	xs:integer
Number	xs:decimal
Real	xs:decimal
Time	xs:time
URI	xs:anyURI

A Metadata element that has its domain as data type, is mapped as *owl:DatatypeProperty*. The range of the OWL property is then given by an XML base type. For example, the range of the metadata element “hoursOfService” is defined as follows.

<owl:DatatypeProperty rdf:ID="hoursOfService">

<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>

</owl:DatatypeProperty>

2.5.10. Domain and Range

In the ISO:19115:2003 norm the data directory *domain* represents the line numbers covered by the entity. *rdfs:range* is used to represent the domain defined in the ISO 19115:2003. OWL classifies properties into two basic groups *owl:DatatypeProperty* and *owl:ObjectProperty*. If the domain in the ISO 19115:2003 is *Free Text, Integer, Real, Boolean, Date, DateTime* , the property is mapped to *owl:DatatypeProperty*. If the domain of a property in ISO 19115:2003 is a class, *owl:ObjectProperty* is used as shown in Table 2-5.

Table 2-5: ISO 19115 Domain and OWL property mapping

ISO 19115:2003 Domain	**Property**
Free Text, Integer, Real, Boolean, Date, DateTime	*owl:DatatypeProperty*
Class or *Association*	*owl:ObjectProperty*

2.5.11. UML Model Stereotypes

A UML *stereotypes* is an extension mechanism for existing UML concepts. An annotation property "*stereotypes"* is used to provide this information. For example, the *stereotypes* for the class MD_Metadata is "*class"*.

<owl:AnnotationProperty rdf:ID="stereotypes" />

<owl:Class rdf:ID="MD_Metadata">

<iso:stereotypes>class</iso:stereotypes>

</owl:Class>

2.5.12. Abstract class

Abstract class is defined in ISO 19115:2003 as a class that cannot be directly instantiated, Since OWL, as mentioned before, does not support an abstract class concept, An annotation property *stereotype* is used with a value of *abstractClass*. An example of OWL stereotype for an abstract class is shown below.

```
<owl:Class rdf:ID="MD_Identification">
<iso:stereotypes>abstractClass</iso:stereotypes>
</owl:Class>
```

2.5.13. Generalization

Generalization is defined in ISO 19115:2003 as a relationship between a generalized super-class and a specified subclass, Figure 2.3.

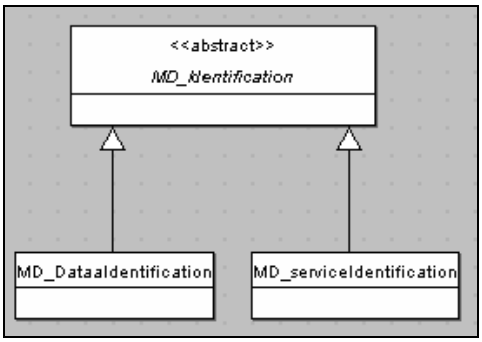


Figure 2.3. ML Generalization relationship in ISO 19115:2003

Generalization in OWL can be expressed as *owl:Class* and *owl:subClassOf*.

<owl:Class MD_Identifiation></owl:Class>

<owl:Class rdf:ID="MD_ServiceIdentification">

<rdfs:subClassOf rdf:resource="#MD_Identification"/>

</owl:Class>

<owl:Class rdf:ID="MD_DataIdentification">

<rdfs:subClassOf rdf:resource="#MD_Identification"/>

</owl:Class>

2.5.14. Enumeration

Enumeration is defined in ISO 19115:2003 as a data type whose instances come exclusively from a list of named literal values, as shown in Figure 2.4. Enumeration is treated through the *owl:oneOf* property in OWL. The process in OWL is to: 1) create an enumerated class; 2) create all the code list elements as the instances of the class, 3) link the enumerated class with an equivalent class, which is a collection of the instances of that class, and 4) the range of the property, which will hold the values of the enumeration, is restricted to allow values of the equivalent created class.

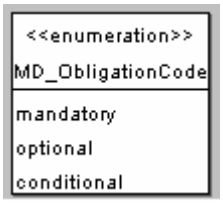


Figure 2.4. UML of Enumeration in ISO 19115:2003

For example, the enumerated class *MD_ObligationCode* is represented as a enumerated *owl:equivalentClass* using *owl:oneOf* collections:

```
<owl:Class rdf:ID="MD_ObligationCode">
<owl:equivalentClass>
<owl:Class>
<owl:oneOf rdf:parseType="Collection">
<iso:MD_ObligationCode rdf:ID="mandatory" />
<iso:MD_ObligationCode rdf:ID="optional" />
<iso:MD_ObligationCode rdf:ID="conditional" />
</owl:oneOf>
</owl:Class>
</owl:equivalentClass>
</owl:Class>
```

2.5.15. CodeList

CodeList are typically used to describe a long list of likely values, which makes it a more open or flexible enumeration. *CodeList* is expressed by creating instances of the class they belongs to, Figure 2.5. However, because *CodeList* needs to retain its ability to be extended in the future, its instances should not be enumerated using the *owl:oneOf* property. The stereotype annotation for CodeList is *codelist*.

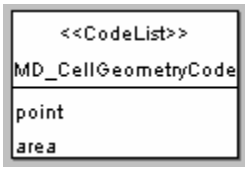


Figure 2.5. UML Code list in ISO 19115:2003

For example, the *CodeList* class *MD_CellGeometryCode* is represented as an enumerated *owl:equivalentClass* rather than using *owl:oneOf* collections.

```
<owl:Class rdf:ID="MD_CellGeometryCode">
<owl:equivalentClass>
</owl:Class>
<iso:MD_CellGeometryCode rdf:ID="point" />
<iso:MD_CellGeometryCode rdf:ID="area" />
```

2.5.16. DataType class

A *datatype* class is a class that describes a set of values representing attributes of a particular object. For example, *citation* is an attribute of MD_Identification. The values of citation are of type CI_Citation, where CI_Citation is a *datatype*. Datatype classes are composed of one or more primitive *datatypes* (e.g. number, string and date), as well as other *datatype* classes.

While primitive *datatypes* are mapped to XML Schema *datatypes*, classes with stereotype *datatype* are mapped to OWL classes. To keep the stereotype information from the ISO specification for the datatype classes, an annotation property *iso:stereotypes* with value *datatypeClass* is created in the OWL class. DQ_Result is an example of a *datatype* class, and its encoding in OWL is shown below:

```
<owl:Class rdf:ID="DQ_Result">
<iso:stereotypes>datatypeClass</iso:stereotypes>
</owl:Class>
```

2.5.17. Union

A union class allows an alternative selection (one of) of two or more classes. This class is used in the ISO specification so there is no need to create a common super type class. An union class is mapped to an OWL class, and its union definition is stored in the iso:stereotypes annotation property with a value of unionClass. An example for MD_ScopeDescription is shown below.

```
<owl:Class rdf:ID="MD_ScopeDescription">
<iso:stereotypes>unionClass</iso:stereotypes>
</owl:Class>
```

Finally, the complete ISO 19115:2003 norm has been mapped into OWL and is being made available to the general public at http://loki.cae.drexel.edu/~wbs/ontology/list.html . Several links have been created at other web-pages, one of which is hosted at the Protégé Ontology Library at Stanford University at http://protege.stanford.edu/ontologies/ontologies.html.

2.6. Summary

In this study, a mapping approach is developed for geographic information metadata, ISO norm 19115:2003, from its conceptualization in UML into an ontology using the Web Ontology Language, OWL. This work has been motivated by the realization that the current implementation approach utilized in the ISO 19139.3, i.e. the use of XML schema, falls well short of being able to fully represent the concepts inherent in the UML conceptualization of the standard. These shortcomings have been discussed

also including the alternative approaches using the Resources Description Framework schema. In contrast, the use of an ontology is shown, and its realization in OWL, has the potential of much better implementing the underlying UML concepts of the ISO 19115:2003. In addition, a step by step conversion process from ISO 19115:2003 UML model to OWL ontology is demonstrated addressing each of the items that, in our opinion, needed a specific mapping rule. While many of the dissimilarities have been resolved and a mapping rule has been derived, a 100% perfect mapping has not been achieved and that the present specification of UML does not permit a complete conversion of an object oriented concept into a knowledge-based concept. This work can help in establishing an automated conversion framework in future that would permit, through an ontology definition metamodel, to convert any object oriented model into knowledge based concept. The ISO 19115:2003 ontology is made available for public use and hope that it will be used and also tested with a number of extensive real world examples for future modification and corrections.

CHAPTER 3: A GENERIC HYDRODYNAMIC MODEL DATA DESCRIPTION FOR THE SEMANTIC WEB

3.1. Abstract.

Sharing of data sets between numerical models is considered an important and pressing issue in the modeling community, because i) the time consumed to convert data sets and ii) the need to connect different types of numerical codes to better map inter-connectedness of aquatic domains. One of the reasons of the data sharing problem arises from the lack of sufficient description of the data, or lack of metadata, which is due to the absence of a standardized framework for these metadata sets. This study describes the development of a framework for hydrodynamic code and data descriptions using the International Standard Organization, ISO, "Geographic Information Metadata,19115:2003" standard. This standard has been chosen not only because of its extent and adequacy to describe geospatial data, but also because of its widespread use and flexibility to extend the coverage. The latter is particularly important as further extensions of the metadata standard are needed to provide a comprehensive metadata representation of hydrodynamic codes and their data. In order to enable the community to share and reuse numerical code data sets, however, they need to be published in both human and machine understandable format. One such format is the Web Ontology language (OWL), whose syntax is compliant with the Extensible Markup Language (XML). OWL gives explicit meaning for machines to automatically extract and compile information available on the web. In this study, an extensive metadata profile using the available elements of the ISO 19115:2003 as well as its extension rules is presented. Based on the metadata profile, an explicit specification or ontology for the modeling

domain has been created using OWL. This ontology permits not only permits flexibility when extending the coverage but also, to share data sets as resources across the internet as part of the Semantic Web. The use of the framework using a two-dimensional finite element code and its associated data sets is demonstrated.

3.2. Introduction

Over the last decade, the accessibility of data through internet based systems has become more widespread with a plethora of web-portals to access real-time data sources, forecasting systems, or direct access to holdings in data bases. This trend has enabled the hydrodynamic processes modeling community, to gain access to a much denser array of data that can be used for calibrating and validating hydrodynamic models in addition to providing data sets for model improvement. While this access to data is a very welcome development it has also prompted the need to develop a common vocabulary and grammar to exchange model data among users of various models in order to ease the burden of re-formatting numerical model data when using different codes. A key to the successful development of such a framework is the adequate description of the model data through a metadata system. Metadata is commonly known as “data about data” [5] and is used to provide conceptual information about a data object. Unfortunately, metadata has always been treated as a lesser important aspect in the world of numerical modeling [6]. Typically the information about numerical model data is placed in a few lines in the header and throughout the data files, an approach that is a common for many of the currently used hydrodynamic models.

A very few hydrodynamic and water quality modeling environments such as BASINS [1] are the exception of this trend. BASINS provides an integrating modeling environment, which uses metadata to describe the content, quality, condition, and other characteristics of model data. However, while the BASINS approach is a right and necessary step towards improving modeling efforts, it is limited to the use on WINDOWS systems and also requires the use of an integrated geographic information system (GIS) that is only commercially available. Another initiative is the *Harmon*IT project (http://www.harmonit.org/) funded by the European Commission that aims at developing and implementing a European Open Modeling Interface and Environment (OpenMI) that will simplify the linking of hydrology related models across Europe [7]. The basic underlying idea is to create a software standard interface that permits software components to request data from each other through “get_functions”. This approach however requires the adherence to an adopted software interface (OpenMI) by all programs and does not follow the idea of using a flexible description of model data whose organization and formatting does not follow any particular rules but can be parsed as needed. Other modeling environments such as SMS [8], GenScn [9] , or DELFT3D [10] permit only the use of specially designed components that are part of a code family or work only with specific set of codes that have been equipped with an custom interface. All of these systems though require a customization of the codes to conform to a certain standard, rather than defining a generic framework that leaves structure and convention of the existing codes unaltered. There are also a number of self-describing data formats available like netCDF [11] and HDF5 [12]. Both formats are intended for storage of large (several Giga or even Peta bytes) scientific data sets using an internal data model for

structuring the data and also employ a special encoding format. As a result, both formats need a suite of tools to en- and decode the files stored in the respective formats. netCDF does not permit external storage of metadata, which means that the files need to be opened to find out what has been stored. HDF5 on the other side does permit this external storage of metadata, which makes it more attractive for use. However, the meta information stored is not regulated (following a standard) and can be set arbitrarily leaving it to a user community to define what it should be. Hence, there exists a need for developing a data exchange and description system that is i) platform independent and generic so the information can be parsed on the Semantic WEB, ii) uses a metadata standard that is internationally recognized, and iii) that has a higher degree of generalization than what is currently available, [13].

3.3. Metadata-Standards and their representation

A number of metadata standards have emerged over the past years. The Dublin Core Metadata Initiative [14] is one of the most widely used metadata standards for describing a wide range of network resources. The most compelling virtue of the Dublin core is the simplicity (only 15 required elements) and its commonly understood semantics. However, the DCM is not adequate to describe the geospatial data for numerical models because it has its origin in the library sciences. Another metadata standard has been published by the Federal Geographic Data Committee (FGDC) whose “Content Standard for Digital Geospatial Metadata” (CSDGM), [15], is extensive and specifies the structure and content of some 220 metadata elements to describe geospatial data sets. At the time of writing this manuscript version 3 of the CSDGM recommendation, however, is slated to follow the International Organization for

Standardization (ISO) recommendation for geographic information metadata, ISO 19115:2003 standard [16] in the near future. Although the ISO standard is primarily developed for digital datasets, its applicability can be extended to use in many other forms of geographic data such as maps, chart and textual documents and general purpose data. Because hydrodynamic model data mostly deals with geographically referenced datasets, this standard is ideally suited for developing a hydrodynamic modeling metadata set.

Given the need to make metadata descriptions available across the WEB, and also realizing that metadata instances need to be passed between researchers (human readability) and also computer systems (machine readability), a suitable encoding scheme must be selected. A first choice would be to use the eXtensible Markup Language (XML) developed by World Wide Web Consortium (W3C) because it is a simple, very flexible text format that has been designed for electronic publications of any kind [58]. However, XML provides only a syntactic framework with very little semantic or meaning-between-data-elements capabilities, and also is quite limited in its ability to be easily extended in a Schema. Alternatively, ongoing W3C efforts have created several other languages based on the XML format to make web resources more understandable for both human and computers. The W3C is currently working on providing the basic building blocks for the future vision of the WEB known as “Semantic Web”, with the goal to build a machine understandable meaningful form of web resources [26]. One of these building blocks is the Web Ontology Language (OWL) that is intended to provide a language that can formalize the domain knowledge with explicit specification in a machine readable format [27].

Ontologies have been successfully applied in the geosciences field for information retrieval, efficient searching, or analyzing domain knowledge. Ontologies can be applied in a variety of ways in geographic information metadata systems to better address consistency validation, interoperability needs, verification, and required system integrity [28]. Bermudez and Piasecki [29] have outlined an approach that can help in overcoming semantic heterogeneities among different community specific metadata standards. A Dublin core Metadata Ontology was used to provide meta information for other ontologies in the HealthCyberMap domain [30]. In another application, Wariyapola *et al.* [31] showed an example to create an ontology and a metadata set for a coastal zone management system. They adopted the FGDC standard for geospatial and biological metadata and the Dublin core metadata standard for cataloguing information. Handschun *et al.* [32] developed a framework, CREAM, to allow creation of metadata for existing web pages based on a domain ontology. Islam *et al.* [28] developed an ontology for the geographic information metadata ISO 19115:2003 norm, by defining mapping rules that transform the ISO 19115:2003 conceptualization in UML (Uniform Modeling Language) into OWL. The availability of the ISO 19115:2003 norm in OWL, even though not (yet) recommended by ISO, is a basic building block for the proposed model data metadata set as it allows the use of the metadata set together with the capabilities inherent in OWL to express the meaning and relations between elements of numerical modeling data sets.

3.4. Hydrodynamic codes ontology

The very first step for users of hydrodynamic models is to select the appropriate numerical code depending on the problem at hand. This depends first of course on the range of physical processes involved and the necessity to include those in the modeling

effort (or not). There are other factors, however, determining the selection of a code, one of which may be the need to accurately map a very complex domain for which rectangular or curvilinear grids are not as suitable as unstructured grids. This need may determine the selection of the integration method, for example the use of Finite Difference versus a Finite Element code. In addition, other circumstances, like the availability of computational resources, may also play an important role when selecting numerical codes. For example, the use of an implicit versus an explicit code may be determined by the need to do long term simulations for which explicit codes may be unsuitable because the available CPU power is too low.

Also, in view of the goal to develop a modeling ontology for a specific hydrodynamic code, it is necessary to bring some structure to the hydrodynamic modeling code world so any specific code ontology ties in into an upper level structure. This upper structure is particular important when considering the desire to exchange input/output data between different codes, a problem that typically requires multiple conversion routines to move data from one format specification to another. A better approach to this time consuming process is to introduce a framework in which data is stored format independent so it can be parsed (on the accompanying metadata) and re-inserted into other codes without reformatting.

Based on the above two motivations, A hydrodynamic model ontology is developed that describes in broad terms a number of numerical codes using several characteristics. A number of fundamental characteristics are selected such as (1) dimensionality of the problem (should it be a 1D, 2D, or 3D formulation?), (2) numerical integration method (Finite Difference, Finite Element, Finite Volume?), (3) numerical

integration scheme (implicit, explicit, semi-implicit?), (4) developer (private, business, government?), and (5) availability (proprietary or public domain?) as shown in Figure 3.1. This list, of course, is not complete and certainly not all-encompassing when thinking about possible other characterizations. For example, one could further sub-divide the 3D code class into what type of grid is being used (rectangular, curvilinear, and un-structured) and also try to consider whether a *Z*-grid or a σ-grid (in the vertical) has been employed. Other criteria could include the bottom or surface friction formulations or (in 3D) what type of turbulence closure model has been implemented (constant eddy diffusion coefficient, one-equation, or two-equation model). This detail however is not necessary when trying to just broadly categorize hydrodynamic codes by using a few but significant criteria. Please notice that the sub-classes *Developer* and *Availability* have been collapsed in order not to overload the image with information content. These two classes though contain other sub-classes and properties for more specificity of these two sub-classes.

Figure 3.2 shows the second block of the code descriptions, i.e. the detailed expansion of the class *IO_Model* that focuses on the actual data sets of the system and the metadata description block. The latter contains the code specific data set (*MD_FEM2D* and “others”) and the link to the classes describing the spatial and temporal extent of the input and output data (marked by the red dotted line rectangles). It should be noted that the blue arrows identify properties of a class, while the black arrows depict the relationship of a parent- and child-class (or sub-class). The difference is that the properties can be range restricted (only a certain set of values can be used when creating instances) and the sub-classes can be given a different set of properties other than those

they inherit from the parent class. These differences are quite significant and will not be further elaborated on here, yet the careful identification of sub-classes and properties is a crucial step of the development work as it determines the ability to properly identify and also extent this code framework.

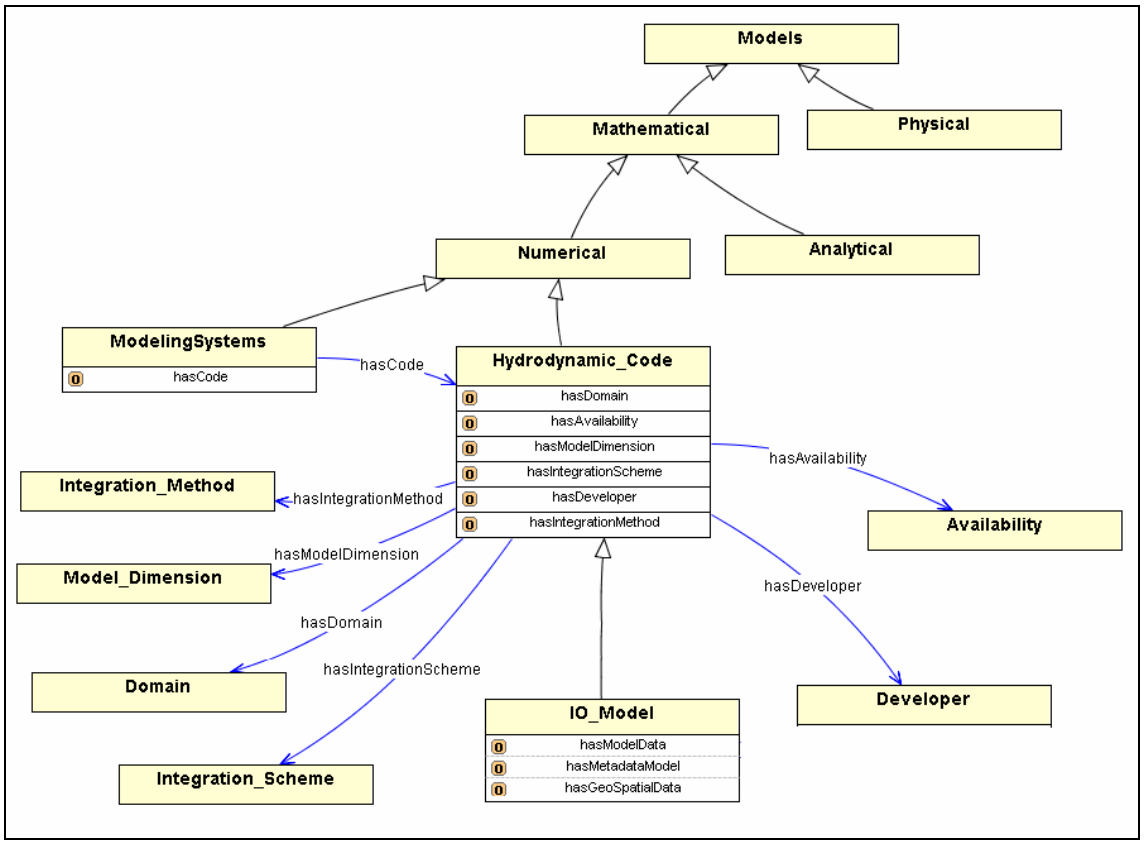


Figure 3.1. General Ontology for describing Hydrodynamic Codes

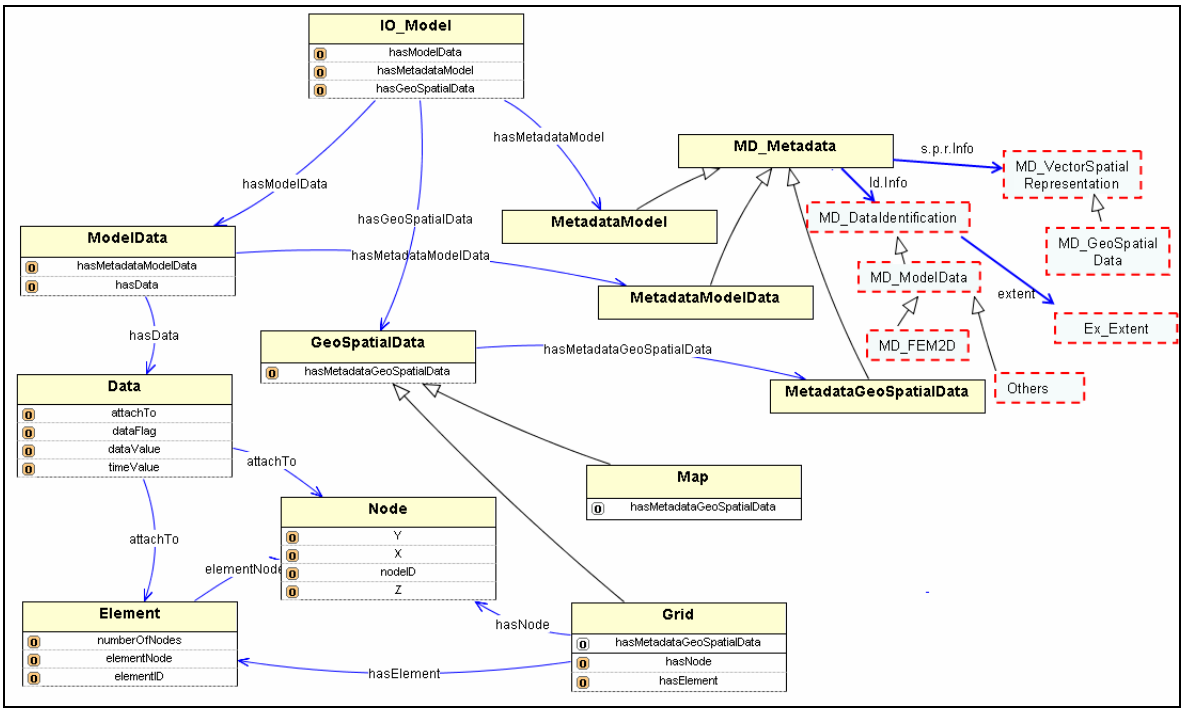


Figure 3.2. Data and Metadata Ontology for Hydrodynamic Codes

Some common surface water and water quality models (in case the hydrodynamic code module could also be used) are shown in Figure 3.3, which contains instances of the *Hydrodynamic_Code* class in the ontology. The list shown (by no means complete) contains several of the commonly known models out of a total of currently 57 different numerical codes that are described without the *IO_Model* components (for the entire ontology see http://loki.cae.drexel.edu/~wbs/ontology/2004/08/model#). These instances are created based on a number of published code collections, like the list provided by the Surface Water and Water Quality Models Information Clearinghouse (SMIC), and also through an extensive search for numerical codes. An in-house two dimensional finite element code, referred to as "*FEM2D*" is also added. The *FEM2D* instance of the *Hydrodynamic_Code* class in the ontology will be used to develop and demonstrate how to domain ontology for the data and the accompanying metadata of this instance. Before

describing that extended ontology, however, it is useful to review a few important characteristics of the geographic information metadata ISO 19115:2003 specification, to better understand where the required metadata elements are emanating from and how the extension rules must be applied to this specific development.

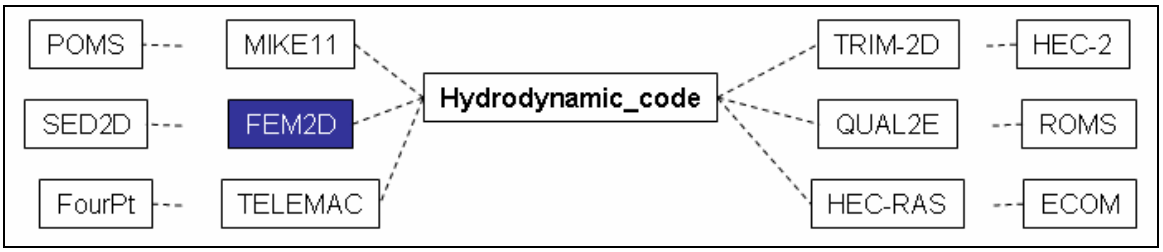


Figure 3.3. Some common codes as an instance of the Hydrodynamic_Code class.

3.5. Metadata profile for hydrodynamic modeling community

ISO categorizes metadata into a number of Unified Modeling Language, UML, [18] packages that are clusters of logically related components. ISO defines an extensive set of metadata elements (about 300) but typically only a subset (albeit carefully selected) of them is used. Metadata elements are also classified as: (1) mandatory (M), (2) conditional (C), and (3) optional, which determine their prominence. Mandatory elements must be documented whereas optional elements may be added or not. Conditional elements are tested through the associated conditions and if the answers are positive they must be documented. ISO recommends about twenty six metadata elements to be used as a core element set of metadata for any dataset description, which are listed in Table 3-1. Of those only seven elements as defined as mandatory (dark grey background), four of them are conditional (light grey background) while the rest are optional.

Table 3-1: Metadata core elements as defined in ISO 19115:2003

Parent Class	**Core Elements**	**Parent Class**	**Core Elements**
MD_Metadata	*fileIdentifier*	MD_Resolution	*equivalentScale*
MD_Metadata	*metadataStandard Name*	MD_Resolution	*Distance*
MD_Metadata	*metadataStandard Version*	CI_Responsible Party	*Role*
MD_Metadata	*language*	MD_Format	*Name*
MD_Metadata	*characterSet*	MD_Format	*Version*
MD_Metadata	*dateStamp*	CI_Citation.	*Title*
MD_DataIdentification	*geographicBox*	CI_Date	*Date*
MD_DataIdentification	*geographicIdentifier*	CI_Date	*dateType*
MD_DataIdentification.	*language*	CI_Online Resource	*Linkage*
MD_DataIdentification	*characterSet*	EX_Extent	*Ex_Temporal Extent*
MD_DataIdentification	*topicCategory*	EX_Extent	*EX_GeographicExtent*
MD_DataIdentification	*spatialRepresentation Type*	MD_Reference System	*Reference SystemIdentifier*
MD_DataIdentification	*abstract*	LI_Lineage	*Statement*

Although a large extent of standard metadata for digital geographic data has been documented in the ISO 19115:2003, the standard does not contain all metadata

descriptions that are needed to describe hydrodynamic codes and their associated data sets. As a result, the ISO standard needs to be extended which can be done by creating a community profile for that specific user group. Based on the extension guideline of ISO, a community profile has been developed for hydrodynamic models as schematically shown in Figure 3.4. This profile is composed of three parts: core metadata components (as required by ISO and shown in Table 3-2), some elements of the comprehensive metadata set (these are pick-and-choose from the existing definitions), and the extended metadata components (to be defined by the community). The selection of optional metadata in addition to the core metadata is left to the community metadata creators, however a balance must be found between excessive description of data ("more-is-better-approach") and a sufficient but small enough description set. In other words, although more metadata elements are better to fully capture all description aspects of a specific data set, the inclusion of too many optional metadata elements could become an overwhelming burden to the users reducing the acceptance threshold.

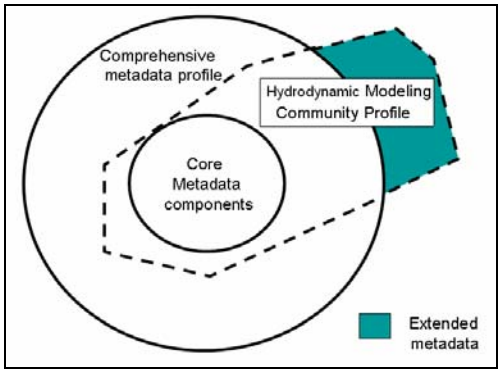


Figure 3.4. Hydrodynamic Modeling Metadata Community profile

Hydrodynamic model data is divided into two basic categories: (1) basic geospatial data; and (2) data related to geospatial data which is called *model data*. Into

the basic geospatial data category fall representations of the numerical grid, maps, digital terrain models, and boundary polygons. These data are typically time-invariant and could be vector-type such as the grid description or raster-type such as a topological map, and are necessary to define the geospatial extent of the model domain. The Model data category, while also geospatially referenced, contains the data that is required to describe physical processes (often also time variant) such as wind speed, water velocity, discharge, water level, viscosity coefficient, roughness coefficient, dispersion coefficient, flow direction, boundary types, tidal elevation or tributary discharge data. Besides these two data categories a hydrodynamic model typically also contains parameters that are not geospatially referenced at all, like run time controls, logical flags, fixed parameters (like the acceleration constant) and so on. These are of course quite code specific because no numerical code is alike.

Two specialization groups of metadata entities are proposed to represent hydrodynamic model data: (1) *MD_ModelData* and (2) *MD_GeospatialData* as shown in Figure 3.5. The *MD_ModelData* metadata entity was defined as a specialization of *MD_DataIdentification* metadata entity (UML class) to uniquely identify resources of the hydrodynamic model. Inside the *MD_ModelData* metadata entity a set of metadata elements was created which describe the data types of the model (e.g. discharge, velocity, dispersion coefficients, and so on) and a code-specific class. In this example, specific class *MD_FEM2D* is created that is an extension from the *MD_ModelDat*a class and that is unique to the *FEM2D* code. This class contains FEM2D-specific information such as the parameter that controls the use of the Petrov Galerkin scheme and the number of iterations required, to name just a few. The *MD_GeospatialData* metadata entity was

defined as a specialized class of the *MD_VectorSpatial Representation* class to represent vector type geospatial data used in a hydrodynamic model such as the model grid. Because a model grid consists of a number of nodes and elements the *MD_GeospatialData* description includes the total number of nodes and elements, a code list of permissible element types and the grid type.

The code-list for *MD_ModelDataTypeCode* (Figure 3.5) is not complete of course, as many more data-types are possible within a numerical code, like pressure, turbulent kinetic energy, energy dissipation, eddy viscosity and eddy diffusivity to name just a few that are inherent in three-dimensional code for example. Our intention here is to provide an example only (for FEM2D), yet it is realized that this list must be extended in the future to encompass all possible data types, possibly including a finer granularity (common data types, 1D types only 2D types only, 3D types only) for better organization and categorization of the metadata.

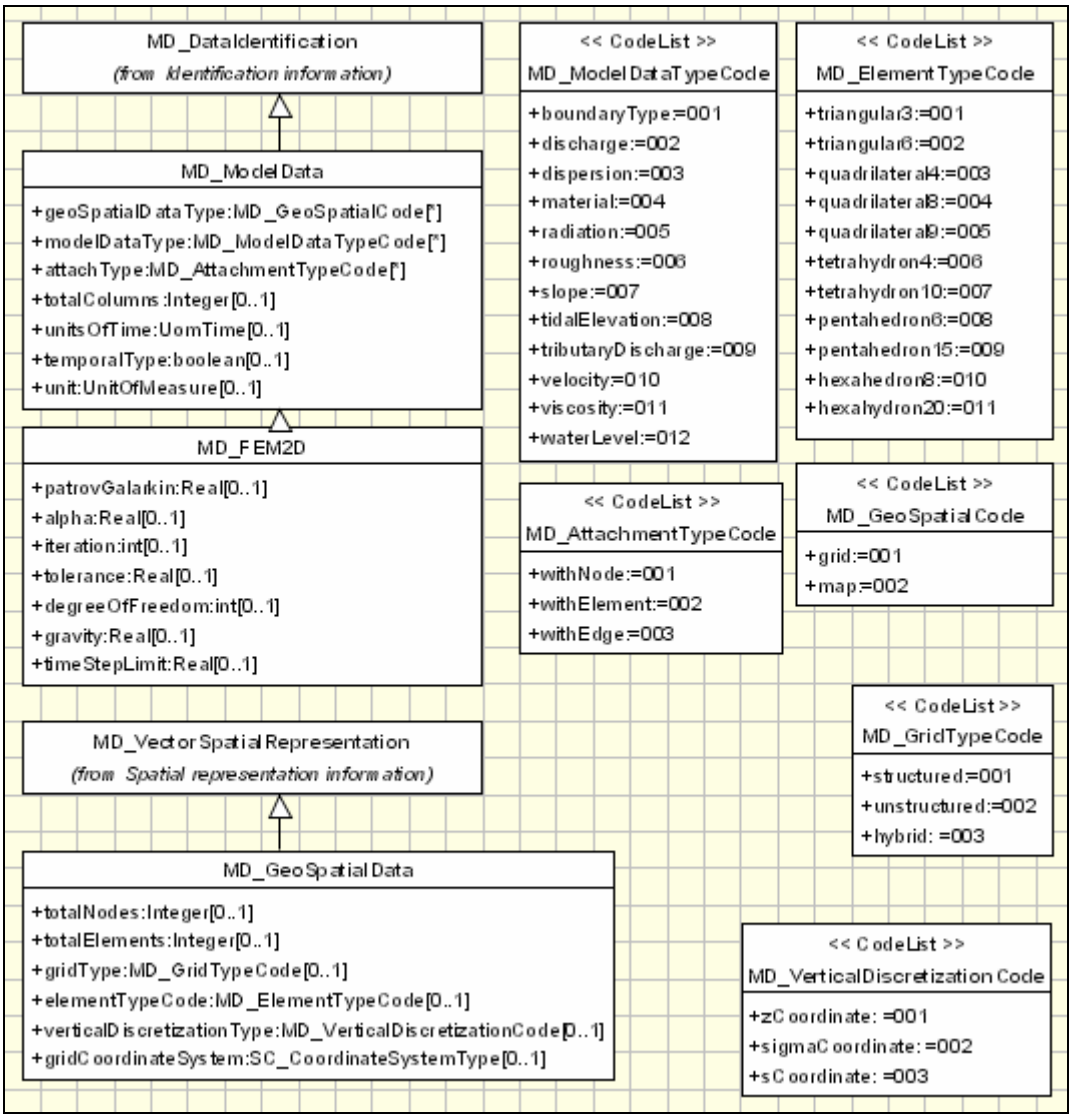


Figure 3.5. List of ISO 19115 Metadata Extensions and Code-Lists for Hydrodynamic Codes

In accordance with the ISO 19115:2003 rules for extensions, each additional metadata element must be defined by its name, short name, domain code, definition, obligation, condition, data type, domain value, maximum occurrence, parent entity, rule, and its rationale and source must be documented and published. A Data directory of the extended elements of numerical model is presented in Table 3-3 and Table 3-4. Table 3-3 describes the name, short name, definition, data type and domain value for all extended

elements, while maximum occurrences, parent entity, rule, and rationale of the extended elements are listed in Table 3-4. The obligation of all extended elements is set to “optional” in order to provide necessary flexibility when describing model data, i.e. there is no “mandatory” or “conditional” settings for the extended elements. Recall however, that each model data description also contains the ISO 19115:2003 core elements, some of which are mandatory or optional. Tables 3-5 to 3-10 provide the contents of the various code lists that have been developed for the metadata descriptions. These contain the different attachment condition of model data with numerical grid (Table 3-5), different permissible element types for finite element (difference) grids (Table 3-6), non-associated geospatial data (at present only grid and map data has been shown as geospatial data, Table 3-7), an enumeration of different grid types, namely structured and unstructured (Table 3-8), an open list for different input and output constituents used in the hydrodynamic model which are attached to the grid (Table 3-9) and an open list of vertical discretization type of geospatial data (Table 3-10).

Table 3-2: Data directory for the extended elements [Obligation = Optional, Condition = none, Source = Drexel University] (Part-1).

Name	Short Name	Definition	Data Type	Domain Value
MD_Model Data	MdlData	information about model data	Specified Class (MD_ DataIdentific.)	
geoSpatial DataType		defines type of geospatial data	Class	MD_GeoSpatial Code<<codeList>
Model DataType	mdlData Typ	defines type of boundary condition	Class	MD_ModelData TypeCode<<Cod eList>
attachType	atchTyp	defines type of attachment with grid	Class	MD_Attachment TypeCode<<Cod eList>
totalColum ns	columns	total number of columns of data sets	Integer	Integer
unitsOf Time	timeUnit	definition of the unit time used for hydrodynamic model	Class	UomTime
unit	unit	definition of the unit used for the dataset	Class	UnitOfMeasure
Temporal Type	tempTy p	indication of whether or not data set is temporal	Boolean	Boolean
MD_FEM2 D	MdlFE M	information about model data for FEM2D model	Specified Class(MD_ ModelData)	
iteration	iteration	number of iteration used to solve set of equations	Integer	Integer
degreeOf Freedom	degFree dom	number of degree of freedoms	Integer	Integer
timeStep Limit	timeStp Lim	time step limit for the simulation	Real	Real
tolerance	Toleranc e	iteration limit for the solver	Real	Real
alpha	Alpha	time integration	Real	Real
Patrov Galarkin	Patrov	indication of Patrov-Galarkin method	Real	Real

Table 3-2: (Continued)

Md_Geo SpatialData	Mdlgrd	description of the grid for the model	Aggregated Class (MD_ Spatial Data)	
totalNodes	nodes	total number of nodes in a numerical grid	Integer	Integer
Total Elements	elements	total number of elements in a numerical grid	Integer	Integer
Element Type	elemTyp	code describing the element type in a numerical grid	Class	MD_ElementTypeCode<<CodeList>>
GridType	grdTyp	grid type: uniformity of the spacing in the element	Class	MD_GridTypeCode<<CodeList>>
verticalDiscretization	verDisc	vertically discretization type	Class	MD_ VerticalDiscretizationCode <<CodeList>>
gridCoordinateSystem	grdCorSys	coordinate system used in the numerical grid	Class	MD_ SC_CoordinateSystemType <<CodeList>>

Table 3-3: Data directory for the extended elements [Obligation = Optional, Condition = none, Source = Drexel University] (Part-2).

Name	**Max Occurrence**	**Parent Entity**	**Rule**	**Rationale**
MD_Model Data	N	MD_Data Identification	New Metadata class to MD_Data Identification	Provides information for documentation of hydrodynamic model data
geospatial DataType		MD_Model Data	New Metadata class	Identifies what type of geospatial data
modelData Type	N	MD_Model Data	New Metadata class	Describes type of boundary: open, close, source, sink
attachType	N	MD_Model Data	New Metadata class	Data value is attached: to node or edge of the numerical grid.
totalColumns	1	MD_Model Data	New Metadata attribute	It is useful for Vector or Tensor type data
unitsOfTime	1	MD_Model Data	New Metadata attribute	Information about the time unit used for the model
unit	1	MD_Model Data	New Metadata attribute	Information about the unit used for measurement
temporal Type	1	MD_Model Data	New Metadata attribute	Identifies whether data is temporal or not
MD_FEM2D	N	MD_Model Data	New Metadata class to MD_Data Identification	Provides information for documentation of hydrodynamic model data
Iteration	1	MD_FEM2D	New Metadata attribute	Identifies the number of iterations to solve the problem
degreeOf Freedom	1	MD_FEM2D	New Metadata attribute	Indicator or the number of degree of freedom
timeStep Limit	1	MD_FEM2D	New Metadata attribute	Sets the limit of the time step
tolerance	1	MD_FEM2D	New Metadata attribute	Sets the iteration limit for solver
alpha	1	MD_FEM2D	New Metadata attribute	Gives the alpha coefficient
patrovGalarkin	1	MD_FEM2D	New Metadata attribute	Identifies whether Patrov-Galarkin scheme is used

Table 3-3: (Continued)

Md_Geo SpatialData	N	MD_Vector Spatial Represen-tation	New Metadata class to MD_VectorSpatial Representation	Provides information for documentation of different geospatial data used in model
totalNodes	1	MD_Geo SpatialData	New Metadata attribute	Gives the total points in a grid
total Elements	1	MD_Geo SpatialData	New Metadata attribute	Gives the total elements of the grid
element Type	1	MD_Geo SpatialData	New Metadata class	Type of element used is important for simulation
gridType	1	MD_Geo SpatialData	New Metadata class	Indicates if grid spacing is uniform or not
verticalDiscretization	1	MD_Geo SpatialData	New Metadata class	Indicates vertical discretization of grid
gridCoordinateSystem	1	MD_Geo SpatialData	New Metadata class	Type of coordinate system used in grid

Table 3-4: Md_AttachmentTypeCode <<CodeList>>

	Name	**Domain code**	**Definition**
1.	MD_AttachmentTypeCode	AtchTypCd	type of the attachment with model grid
2.	withNode	001	data attached with the node
3.	withElement	002	data attached with the element
4.	withEdge	003	data attached with the edge

Table 3-5: MD_ElementTypeCode <<CodeList>>

	Name	**Domain code**	**Definition**
1.	MD_ElementTypeCode	ElmTypCd	definition of the element type for the grid
2.	tringular3	001	each element contains 3 nodes
3.	triangular6	002	each element contains 6 nodes
4.	quadrilateral4	003	each element contains 4 nodes
5.	quadrilateral8	004	each element contains 8 nodes
6.	quadrilateral9	005	each element contains 9 nodes
7.	tetrahydron4	006	each element contains 4 nodes
8.	tetrahydron10	007	each element contains 10 nodes
9.	pentahedron6	008	each element contains 6 nodes
10.	pentrahydron15	009	each element contains 15 nodes
11.	hexahydron8	010	each element contains 8 nodes

Table 3-6: MD_GeoSpatialCode <<CodeList>>

	Name	**Domain code**	**Definition**
1.	MD_ GeoSpatialCode	GeoSpCd	definition of the geospatial data Type
2.	grid	001	numerical grid
3.	map	002	map or chart

Table 3-7: MD_GridTypeCode <<CodeList >>

	Name	**Domain code**	**Definition**
1.	MD_GridTypeCode	GrdTypCd	definition of the grid types
2.	structured	001	the volume elements are well ordered and a simple scheme (e.g., i-j-k indices) can be used to label elements and identify neighbors
3.	unstructured	002	volume elements can be joined in any manner, and special lists must be kept to identify neighboring elements
4.	hybrid	003	volume elements are mixed with both structured and unstructured types

Table 3-8: MD_ModelDataTypeCode <<CodeList>>

	Name	**Domain code**	**Definition**
1.	MD_ModelDataTypeCode	MdlDataTypCd	definition of different types of data used in the model
2.	boundaryType	001	boundary conditions such as open boundary, discharge boundary, no flow boundary or tidal boundary
3.	discharge	002	river discharge or flow
4.	dispersion	003	dispersion co-efficient
5.	material	004	material property to represent roughness
6.	radiation	005	radiation stress coefficient
7.	roughness	006	Manning's roughness coefficients
8.	slope	007	slope or angle of the direction of flow
9.	tidalElevation	008	boundary data for tidal elevation
10.	tributaryDischarge	009	boundary data for tributary discharge
11.	velocity	010	velocity of water
12.	viscosity	011	viscosity coefficient data
13.	waterLevel	012	water elevation or stage

Table 3-9: MD_VerticalDiscretizationCode <<CodeList >>

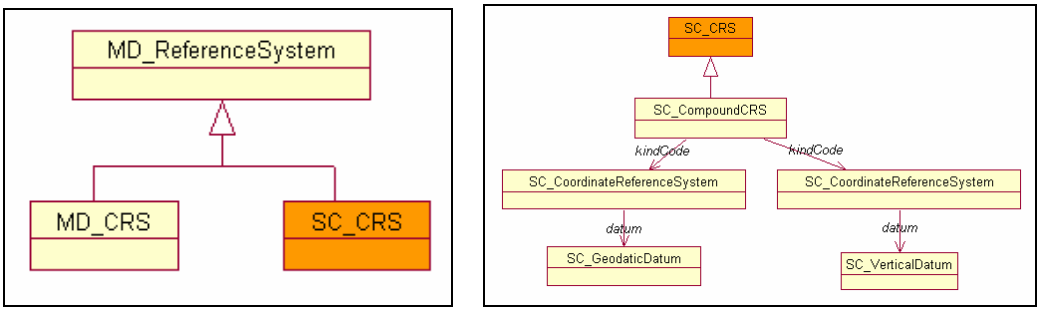
	Name	**Domain code**	**Definition**
1.	MD_ VerticalDiscretizationCode	VerDisCd	definition of the vertically discretization types
2.	zCoordinate	001	coordinates are measured from a fixed levels in the vertical z-direction
3.	sigmaCoordinate	002	coordinates are expressed as sigma defined as the ratio of water surface to fluid depth
4	sCoordinate	003	Coordinates are expressed as function of water fluid depth and sea surface

Finally, while the ISO 19115:2003 contains many of the desired description elements, links to other norms within the 19000 group are necessary to deal with specific

aspects, like the ISO 19108 for temporal schema [59], the ISO 19103 for units of measurements [57], and the ISO 19111 for spatial referencing by coordinates [60]. The ISO 19115:2003 shows only a dependency on the ISO 19103 norm without providing any direct relation to express unit of measurement of the dataset. To this end a new metadata element "*unit*" is defined to describe the measurement unit of the dataset. However, the ISO 19103 norm provides support for only 7 data types namely, area, angle, time, scale, length, volume, and velocity without addressing physical quantities such as discharge, viscosity, and dispersion. Therefore, our extension of the *UnitOfMeasure* class now contains the sub-classes *UomDischarg*e, *UomDispersion*, *UomViscosity*. Accordingly the elements "*discharge*" (with value "*cubic meter/second*"), "*viscosity*" (with value "*square meter/second"* for kinematic viscosity), and "*dispersion*" (same units as viscosity) are also added to the *ISOStandardUnits* code list.

The description of the coordinate reference system (CRS) is important for all geospatial data such as grids or coastlines. The ISO 19115:2003 norm provides a separate entity "*MD_ReferenceSystem*" and a subclass of that entity "*MD_CRS*" to describe this information. The root class "*MD_Metadata*" has been connected by an association, "*referenceSystemInfo*" with "*MD_CRS*" to provide the reference system information. Three types of metadata descriptors namely *ellipsoid*, *projection* and *datum* are used to identify these features in the CRS. These classes, however, are not sufficient to adequately a numerical grid, because often the vertical coordinates of a numerical grid are related to the mean sea level, MSL, or the mean lower low water, MLLW. In other words, the horizontal and vertical components of a grid may originate from or are based on two different geospatial reference systems, which prompts the need to use a compound

reference system. While the ISO 19115:2003 does not presently support this idea the ISO 19111 norm does contain the option for compound CRS which can be used as an extension to the ISO 19115:2003. Hence, the "*SC_CRS*" class as a subclass of "*MD_ReferenceSystem*" is defined as shown in Figure 3.6.



(a) (b)

Figure 3.6. (a) Extension of ISO 19115:2003 class "MD_ReferenceSystem" (b) Compound coordinate reference system using two different datum

3.6. An example data/metadata set for the FEM2D code

As a consequence of using the FEM2D code as an example instance, the creation of twelve ModelData file pairs (one for the metadata description, one containing the raw data), and one GeospatialData file pair containing the grid information is required. These numbers may differ from code instance to code instance as more or fewer constituents are being identified as necessary model data. The exact number is determined in the model specific sub-class as mentioned before. Because each of the metadata files contain several hundreds of code lines and the raw data files potentially many thousands, the GeospatialData file will be restricted to use as a demonstration. The file contains the grid information (x,y coordinates and bathymetry, as well as the connectivity matrix) for a model domain (about 23 km in length) encompassing the upper Potomac estuary adjacent

to Washington, DC, as shown in Figure 3.7, and is comprised of 1171 quadrilateral elements and 1408 nodes.

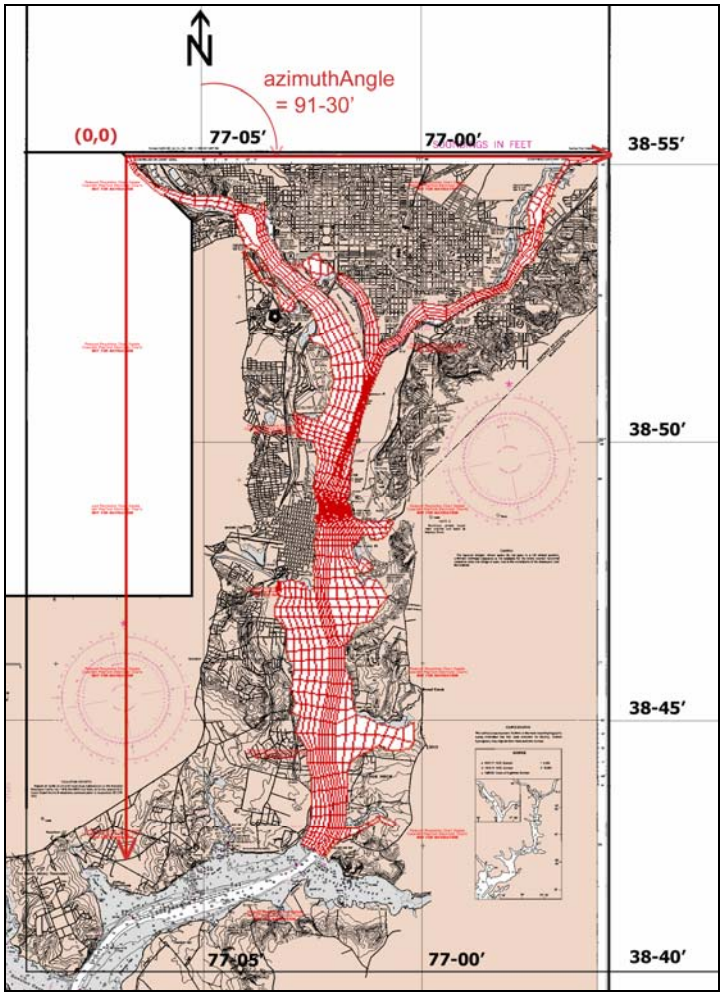


Figure 3.7. Numerical grid overlay with part of nautical chart showing grid coordinate system and its rotation with respect to compass North.

The complete set of descriptive entries in the grid_metadata.xml file encompasses 23 elements for which a total of 80 property values must be supplied. Part of the metadata content is shown in the left panel of Figure 3.8. The top lines contain the Universal Resource Locator (URL) identifier for the raw-data file (that can reside anywhere on the WWW) and the lower block refer to information regarding the grid. The right panel

shows a segment of the node coordinate descriptions (top) and a subset of the connectivity information (bottom). The formatting of the raw-data file in this instance is format-less, i.e. the numbers are only space separated, which allows for easy parsing. The chosen number types are float (coordinates) and integer (connectivity), which are ISO standard types.

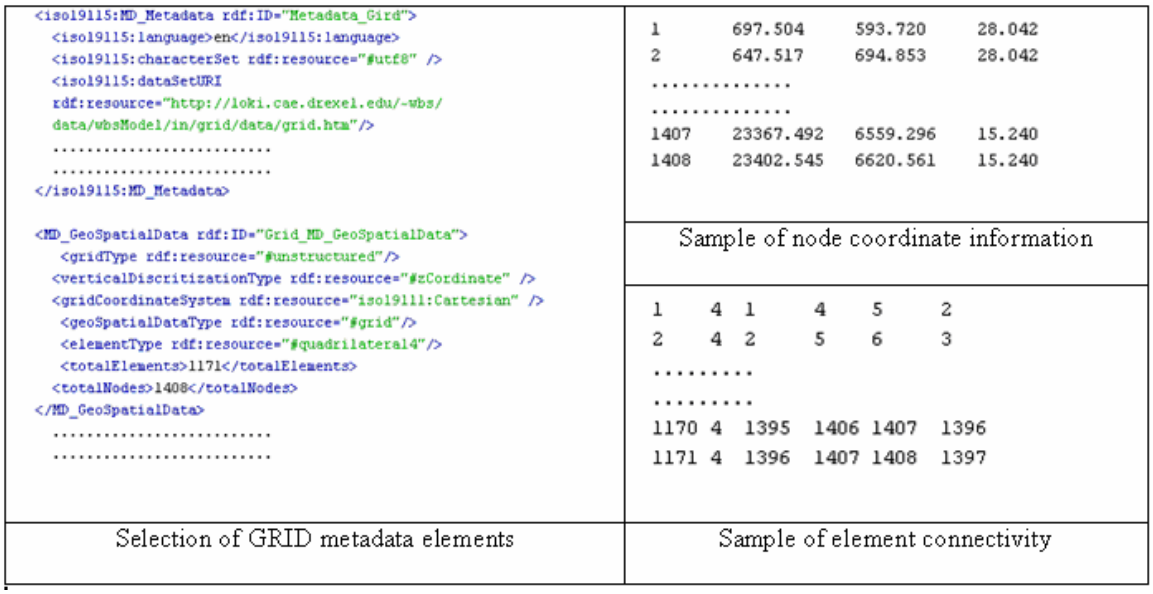


Figure 3.8. Samples from the Example Metadata and Grid-Data Files

In creating metadata instances for each of the previously mentioned data files, it is important to recall that that each metadata file will contain the core element set (required by the ISO 19115) as was discussed in section 2. This can be done automatically in many instances because the values required repeat themselves in each metadata file. The second step is to select a subset of all other optional and conditional elements that can be related to the grid metadata description. For example, one can add the email address and the telephone number as additional information elements to the *MD_ResponsibleParty* class of this metadata set. The final step is to fill the extended elements of metadata as

described in section 3. The grid files are being made available and can be viewed and downloaded at http://loki.cae.drexel.edu/~wbs/data/wbsModel/ that also hosts a pair of example files containing time variant information associated with the velocity field. The site also provides a number of EXCEL files that contain the detailed lists of all elements, their properties, and the chosen values to create the instances shown. The metadata instance document of the numerical grid has been documented in Appendix 1.

3.7. Summary

In this study a metadata structure for hydrodynamic model data is proposed that can potentially pave the way of how to inter- and exchange data between different types of codes and model domains. More specifically

- The ISO 19115:2003 metadata norm is selected as a basis to generate the metadata framework for hydrodynamic codes. The ability of the ISO to provide a very generic base set of elements (and properties) for this purpose that is based on the set of recommended core elements is demonstrated. While some of these elements are considered mandatory and as a result must be included regardless, others optional. Conditional elements whenever possible are tried to avoid either through avoidance or through removing the condition by making them mandatory or optional.
- The need to extend the overall coverage of the ISO 19115:2003 in various aspects is shown as it was not able to provide all necessary descriptions for hydrodynamic model data. While have assumed the regular elements to be present, this work emphasized the extensions (elements , properties and code-lists) necessary, where they need to be inserted and also how

certain aspects of the metadata structure, like time and units, needed a linkage to other ISO metadata standards as the ISO 19103 (units) and ISO 19108 (time). In particular, descriptions for elementary components like a numerical grid (nodes forming elements, connectivity matrix, reference systems), and linkage between geospatial reference points (nodes) and associations of these points with hydrodynamic variables has been demonstrated.

- The use of the Web-Ontology language OWL as an encoding medium for the hydrodynamic metadata community profile is suggested, because it best captures all aspects of the Unified Modeling Language, UML, in which the ISO norms are presented. In other words, all elements, their properties, and their relation to and among each other are manifested in an ontology that can be changed and updated as the profile might change. This flexibility is a core aspect of the chosen approach because it permits the change and growth of this framework to cover more numerical codes and their specifications.
- An ontology is created to better describe and therefore distinguish numerical codes based on their integration schemes, dimensionality, type, grid selection, and availability. While not exhaustive in its descriptions, it provides for some categorization of the available codes (not a complete list) that can be used at later stages when filling in the code specific description classes (or ontologies). This ontology too is flexible and easily being expanded as additional codes are being added.

- An example is provided using a two-dimensional FEM code for which a specific ontology is developed to describe the data structure of this particular code. The example demonstrates the creation of one geospatial (grid) data and 12 model data file pairs, one containing the metadata information the other the “raw” data. Each of the metadata files contain the core selection as a basis and then the specific extensions necessary to describe the associated data variables, while the raw data is stored in a format less (space limited only) fashion.

It is clear that the scope of this framework is extensive and would need to be filled over time. Yet this approach can aid in overcoming the currently existing lack of data interoperability among different hydrodynamic codes.

CHAPTER 4: KNOWLEDGE BASED WEB SIMULATION OF HYDRODYNAMIC PROCESSES

4.1. Abstract.

Hydrodynamic models generally deal with large sets of data and utilize substantial computational resources. Powerful, robust servers with extensive storage capabilities are desirable for rapid execution. Unfortunately, it is not always possible to effort those kinds of facilities whereas a centralized computer system together with a user access interface can be a viable alternative for many clients. The simplest way a client can communicate with the central simulation server is by a web browser because it is available as a pre-installed application on most every computing platform purchased today. This type of environment is called web based simulation or WBS. In this chapter, The concepts necessary to design and develop a WBS for the simulation of hydrodynamic processes using legacy (FORTRAN) code are introduced here. A formal specification of the simulation domain or an ontology has been developed that is the underlying concept to share, retrieve, and move the simulation data between the different components of the WBS. This ontology can also be used for future analysis and reuse of the simulation domain concepts and the associated data sets.

4.2. Introduction

Numerical models have been used in many scientific disciplines to better understand the physical behavior of nature. A number of hydrodynamic codes are now available to investigate complex flow phenomena of rivers, estuaries, lakes or coastal regions. Despite recent advances to include 3-D representations of the simulation domain

and the ever improving level of graphical display options, the modeling user community still faces some shortcomings when faced with the need to move data around between pre- and post processors and to exchange model data in the user community. Typical problems include the lack of a standardized framework to describe model data, inadequate model data exchange facilities, insufficient interoperability of models among different platforms or operating systems, inefficient search and retrieval of modeling information, and the absence of the possibility to share and reuse the knowledge already gathered about a certain simulation domain. In addition, several scientific communities have recognized the need to develop numerical models that will serve long term objectives, are not proprietary but based on open source components, and should serve as a resource to the community for scientific exchange and further improvement. This has spawned the idea to develop community models. There are many relatively sophisticated community models available such as the groundwater community model MODFLOW [33], atmospheric community models MM5 [34], and community climate system models CCSM [35], to name just a few. These models are freely available as a modeling tool within the respective community and allow scientists to focus on their needs rather than building a model from scratch [36]. However, most of these community models are not yet designed to operate in an integrated environment that would ease the work burden that comes with the need to share and exchange modeling data within the community. Consequently, the next step for the development of numerical models should focus on a standardized data description, an improved functionality that permits better sharing of both codes and model data, and provide a platform for preprocessing, execution, and retrieval of simulation results in an environment that is operating system independent. In

addition, a modeling system should permit a multi-user simulation environment where models can run in the server machine and users can interact with the server applications using a standard access tool like a web browser. A system of this kind is typically classified as a web based simulation, or WBS.

The concept of WBS was first introduced in mid 90's when web browsers became available [37]. The WBS concept is based on the idea that any user can perform a simulation either in the client machine or on the server machines using a web browser. Moreover, it can potentially utilize a wide range of databases and information systems through the web. Because one of the priorities is to build a platform independent system, Java has been recognized as the essential language for WBS [38]. Several WBS environments have been developed based on Java such as JSIM [39], simjava [40], DEVSJAVA [41] to this date. Unfortunately, most of these environments are either currently unavailable or have not been further developed. Wiedeman [42] conducted a review of these systems and found that most of them were used for test scenarios only and that actual user requirements were not taken into account. It has been found that only very few WBS systems that permit data I/O operations of several 100Mbytes of data one of which is the Websim3D system [43] that permits fast access to view and download earth quake simulation results. The development of a WBS environment for large scale simulations such as the simulation of hydrodynamic processes is quite challenging the process of which have been outlined in the following sections.

4.3. Web Based Simulation of Hydrodynamic Model

Typically hydrodynamic model runs require the I/O of a substantial amount of data related to water elevation, discharge, dispersion data, wind effect data, roughness,

viscosity data, boundary and initial condition data, all of which may be spatially and temporally invariant. As discussed in chapter 3, these data are classified into two categories: (1) geospatial data, and (2) model data. Geospatial data includes maps, the numerical grid, the bathymetry, and the digital elevation model, while model data includes the state variables and all coefficients and constants, which are geo-referenced to the geospatial data sets. In addition, parameters such as gravity, iteration counters, tolerance limits, to name just a few, have also been included in the model metadata. A multi layered data model has been developed to handle the model data as depicted in Figure 4.1, for which the geospatial data set serves as the basis. Each layer represents a snap shot in time of the model data for a specific time, i.e. time evolution of a specific variable is stacked in layers above the base layer.

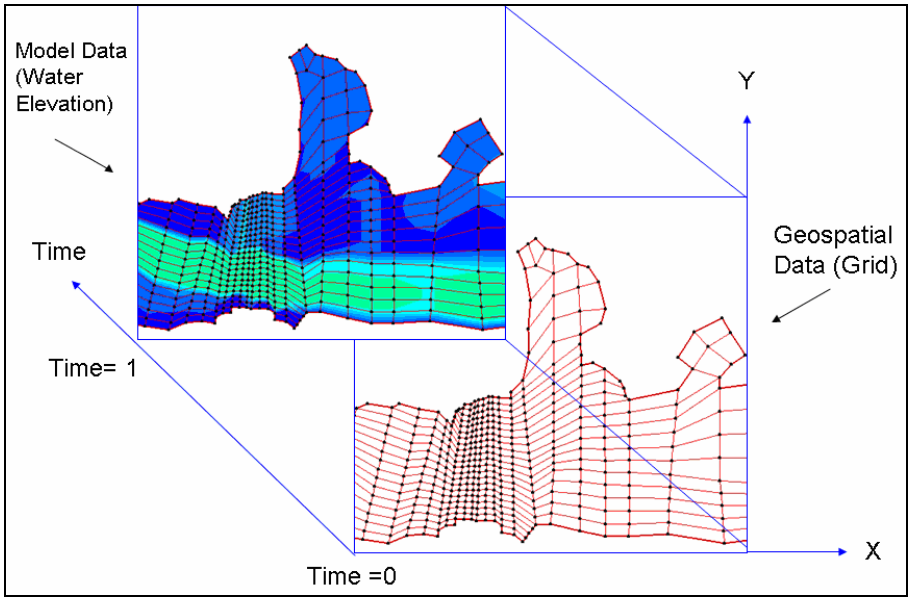


Figure 4.1. Layer data model of the hydrodynamic model data

The next required step is to create an unambiguous description for the layer model data. This is done through defining a metadata set that encompasses and incorporates controlled vocabularies [13]. To this end the metadata community profile from chapter 3 will be utilized, where the need to publish metadata using Web Ontology Language, or OWL is elaborated. OWL was specifically designed to formalize the knowledge of any specific domain using explicit specifications that are expressed in a machine understandable format. The formal specification of the domain knowledge through an ontology permits much better retrieval, share, reuse or analyzing of this knowledge, attributes that are essential when designing a WBS. Ontologies have been successfully applied in many fields that deal with information retrieval, efficient searching, or analyzing domain knowledge see for example Miller *et al*. [61]. Therefore, in this study a WBS environment for hydrodynamic processes based on a simulation domain ontology is developed. This WBS environment has four major components: (1) simulation domain ontology, (2) hydrodynamic code and coding language, (3) Graphical User Interface (GUI) and its architecture, and (4) data storage system. In the following sections, these components of WBS environment will be discussed in details.

4.3.2. Simulation domain ontology

Using OWL, an ontology was created to describe a numerical model as shown in Figure 4.2. Geospatial data encompasses the numerical grid, maps, costal boundaries, charts, or a digital elevation model all of which are represented as "*GeoSpatialData*" class in the ontology. The most important geospatial data set is the numerical gird, which is represented as "*Grid*" class in the ontology. Numerical grids have nodes and elements, which are represented as "*Node*" and "*Element*" class in the ontology, respectively.

Model data includes data, which is related to geospatial data and is represented as "*ModelData*" class in the ontology. In this instance, model data includes but is not limited to wind, discharge, velocity, water elevation, viscosity coefficient, boundary types, Manning's roughness coefficient, dispersion coefficients, tidal elevation, and tributary discharge. These model data have been represented as a subclass of the "*ModelData*" class in the ontology and could consist of thousands of individual data components. These individual data components are represented as "*Data*" class in the ontology and are connected with either the "*Node*" or "*Element*" class of the ontology.

Besides the necessary I/O organization of data and the description of the flow of these data streams, metadata about the model execution itself needs to be incorporated. These include more general descriptions about the purpose of the simulation, time intervals, modeler in charge, execution times associated with the run, and so on. To this end the WBS ontology has been incorporated into the OWL encoded ISO 19115:2003 to create the foundation of metadata classes. Because of the two major categories of data namely, model data and geospatial data, at least two metadata classes are needed. Moreover, one additional metadata class is needed to describe the hydrodynamic model. Therefore, based on the root class of the ISO metadata ontology ("*MD_Metadata*") three subclasses were created in the WBS ontology: (1) "*MetadataModel*", (2) "*MetadataGeoSpatialData*", and (3) "*MetadataModelData*". In the WBS ontology, each geospatial data type has a metadata class "*MetadataGeoSpatialData*". Similarly, each model data type has a metadata class "*MetadataModelData*". A hydrodynamic model could have some model parameters such as gravity, degree of freedom, number of iteration, tolerance limit etc. These parameters are represented as "*MetadataModel*" class

in the ontology. This web based simulation ontology based on OWL has been documented in Appendix 2.

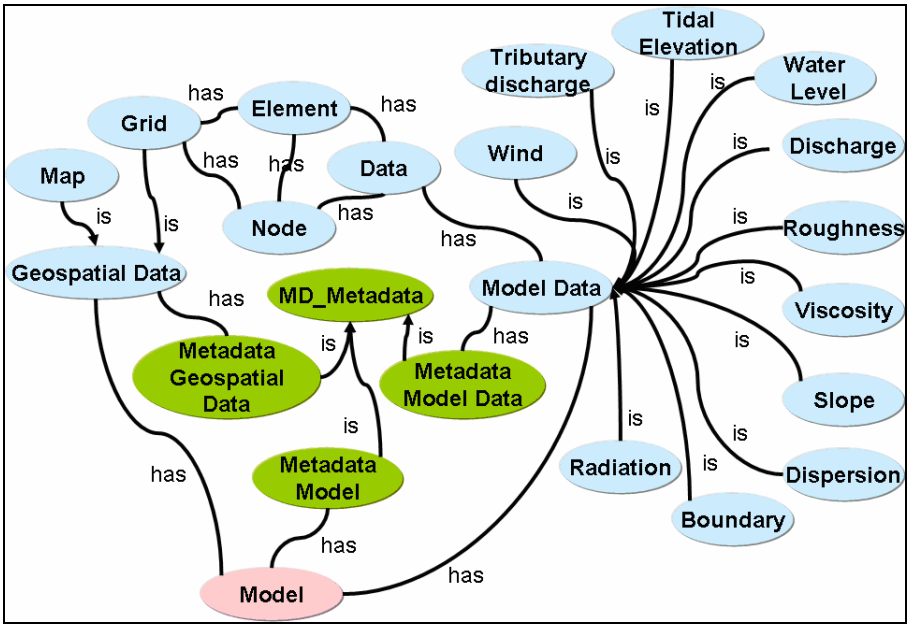


Figure 4.2. Ontology for data and metadata of a numerical model

4.3.3. Hydrodynamic code

This study uses a two dimensional, vertically averaged, finite element code for the numerical integration of the governing shallow water equations. The formulation is second order accurate in time and 5^{th} order accurate with respect to numerical dispersion and diffusion in space. The model was originally developed by Katopodes [62] and has since been applied and tested in a number of applications; see for example Piasecki and Katopodes [63]. The code has been applied to a test bed that encompasses the Upper Potomac estuary around Washington D.C. It is comprised of ~1400 nodes and extends over a domain approximately 23 kilometers in length. The flow in this part of the Potomac is primarily driven by the tide signal, which results in significant flow reversals making this domain a highly dynamic and unsteady flow regime.

4.3.3.1 Coding Languages

Most hydrodynamic models developed in the last decades were written in Formula Translation (FORTRAN) programming language. These codes are also called legacy-codes and provide a rich source of very sophisticated and far advanced numerical codes that are very valuable to the user communities. Because of this fact, it is very desirable to maintain these codes or even continue to use them because the translation into other languages is prohibitively expensive. Since JAVA language is used to develop the WBS environment, a communication or conversion system needs to be developed between the programs written in two languages. One way to solve this problem is to use Java Native Interface (JNI) to run the FORTRAN code. JNI permits Java code to run applications and libraries written in other languages, such as C, C++ [64] or, assembly language. However, JNI cannot call FORTRAN programs directly, rather, FORTRAN programs must be invoked from C/C++ subroutines. Aubourg [65] showed that a FORTRAN code can be accessed either being called from C++ or the FORTAN code itself calling a C++ routine. Zeng *et al*. [66] showed an example of using JNI to run a multi-reaction model (MRM) written in FORTRAN for contaminants transport in soil. They developed a JAVA based simulation Applet that runs in the client browser to solve contaminant problems.

Another approach to utilize FORTRAN code in a JAVA environment is to use the Common Object Request Broker Architecture (CORBA) [67]. CORBA provides interoperability among clients and servers distributed over a heterogeneous environment, is completely platform independent, and supports a number of languages such as C/C++,

JAVA, and COBOL. CORBA is accessible through the Enterprise JavaBeans (EJB) module, a server-side component architecture for the Java 2 Enterprise Edition (J2EE) platform [68]. EJB uses the Internet Inter-Orb Protocol (IIOP) as a communication bridge between the CORBA server and the Java Servlets [69]. While FORTRAN subroutines are accessible through C++ programs, the CORBA server provide access to these C++ programs. Then web browser clients can communicate with web servers invoking JAVA Servlets, a server side technology for extending the functionality of a Web server and for accessing existing business systems [70]. An overview of these communication steps are shown in Figure 4.3. Chen *et al*. [71] demonstrates an attempt to create a web based power system simulation environment using such an approach. Although this approach demonstrates the possibility of JAVA to communicate with programs written in a number of other languages, it needs both Web Server and EJB server running. In addition, CORBA's communications and programming interface is quite complex and also need extensive server side maintenance when implemented.

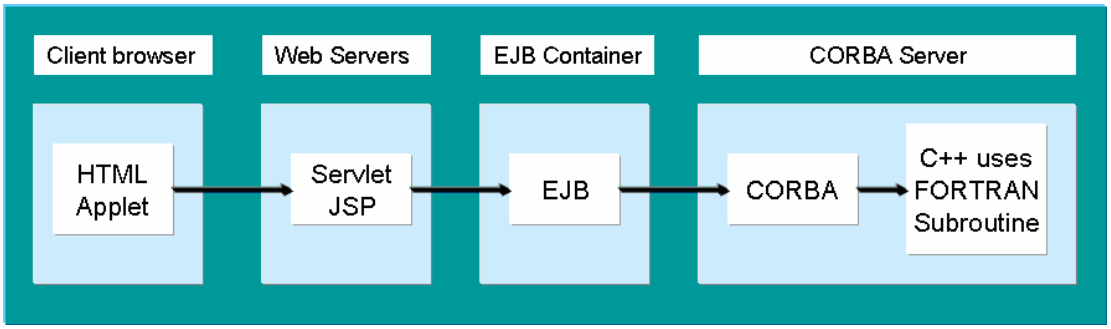


Figure 4.3. FORTRAN programs run through Java and CORBA Interface

It is clear from the previous elaborations that the use of FORTRAN legacy codes is not as straightforward in a WBS system as one would like and that it requires the construction of a carefully thought through approach to permit these legacy codes to run

in an WBS environment. Based on the above-shown two approaches, the first is chosen because it is less complex and also because of its less extensive maintenance requirement on the server side. The steps to run a FORTRAN code using Java Virtual Machine (JVM) are summarized in Figure 4.4. First, a Java class should be created to declare native methods, which includes a main method that calls these native methods. Secondly, a header file should be generated for the native method using javah with the native interface flag -jni. Once you have generated the header file, it can contain the formal signature for the native method. Third, FORTRAN code should be compiled to generate the object code. Finally, an implementation of the native method needs to be written in the C or C++ language. The header, the FORTRAN object code and the implementation files should be compiled and then linked into a shared library file (for UNIX) or a dynamic link library (for WINDOWS). A JAVA based program can then load the shared library files (or dynamic library files) and use the embedded FORTRAN subroutines that originated from the legacy. This is a little tedious of course, but appeared to be the only approach possible to achieve our objective. Also, there is a slight shortfall in the desire to have the WBS entirely operating system independent, because the compilation and linking process is typically different for each type of operating system (and different compilers), i.e. a number of these compilation and linking procedures would have to be executed to generate a set of WBS that are compatible with various OS.

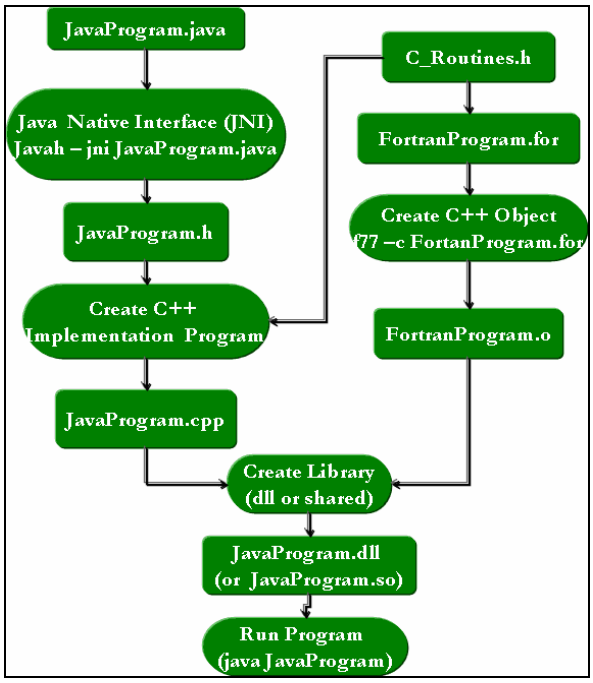


Figure 4.4. FORTRAN program runs using Java Native Interface (JNI)

4.3.4. Graphical User Interface (GUI)

Because of the potentially large amount of necessary data transfers between client and server, a client-side-request and server-side-simulation approach is adopted as execution mode. This will limit the communication between client and server and therefore omit a potential data transfer bottleneck. The best architectural design for of this kind of system was recommended by [72] who suggests to use a *Model-View-Controller (MVC)* architecture. The details of the *MVC* architecture and how it is adopted for creating a GUI to display different component of the WBS environment will be discussed in the next subsection.

4.3.5. Model View Controller (MVC) Architecture

The performance of large simulation systems can be improved if the model or business logic is separated from the model views or presentation logics. This design pattern has been known as *Model-View-Controller (MVC)* architecture, which suggests to divide the system into three components (1) *Model*, (2) *View* and (3) *Controller* [73]. A *Model* is the representation of the simulation domain; *Views* are the visual representations of the *Model*; and *Controllers* handle the user interactions with the model. *MVC* has been used in many software development projects and provides the fundamental basis for the JAVA SWING API to support GUI and graphics functionality [74]. The *MVC* is adopted as the basic architecture of our simulation environment whose components are shown in Figure 4.5. An object data *Model*, which is based on an OWL ontology, is the core of the system. The *Model* stores and retrieves data from an object relational database, PostgreSQL [75]. The *Model* also keeps track of each registered view and will notify its *View* components if any change in the *Model* occurs. These *View* components are built using Java Server pages (JSP) and contain visual components such as Java Applet or, HTML Form elements. During the session *View* components retain a state of the *Model* to receive data and also contain one or more *Controllers*. Any request for the changes in *View* component will be sent to the *Controller* (which is a Java Servlet program) that contains a reference of the *Model*. Any request for change from the *View* will prompt the *Controller* to update the *Model*. The *Controller* also decides which *View* will be displayed to the user. The *MVC* architecture gives maximum flexibility to the WBS system so that any *View* can be added or deleted from the system without having to change the *Model*.

Currently seven different *Views* are registered to display the different components of the WBS system: (1) *Search View*, (2) *Metadata View*, (3) *Edit View*, (4) *Boundary View*, (5) *Simulation View*, (6) *Display View* and (7) *Help View*. The *Search View* is used to search for any pre-existing model. This search can be performed in simple-mode through model description keywords or with a more advanced mode via metadata elements. Once the desired model has been found, the user can copy, rename, or delete it. The *Metadata View* is used to create, display, edit, or delete metadata for the hydrodynamic model. In *Display View* mode the user can display model results as a contour plot. The *Edit View* mode permits the display, creation, edit or deletion of model data. The *Boundary View* helps to set the water level and discharge boundary constraint of the model. The *Simulation View* tracks the execution of the model for the desired length of the simulation. The *Display View* also supports the display of temporal model data as a time series plot for a specific point. Finally, the *Help View* provides suggestions and tips to the user about the WBS environment. Figure 4.6 shows a schematic diagram of the communication of these client side *Views* with server side Java based Application Programming Interfaces (API) and data repositories. The details of the functionality of these different *Views* will be discussed in the following sub-sections.

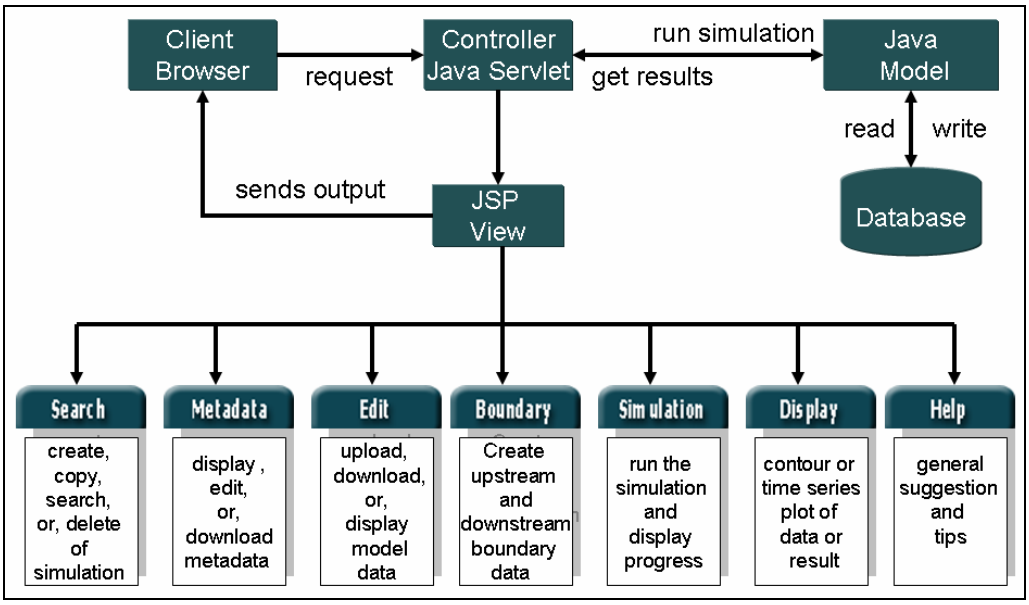


Figure 4.5. WBS system architecture based on *Model-View-Controller (MVC)* pattern

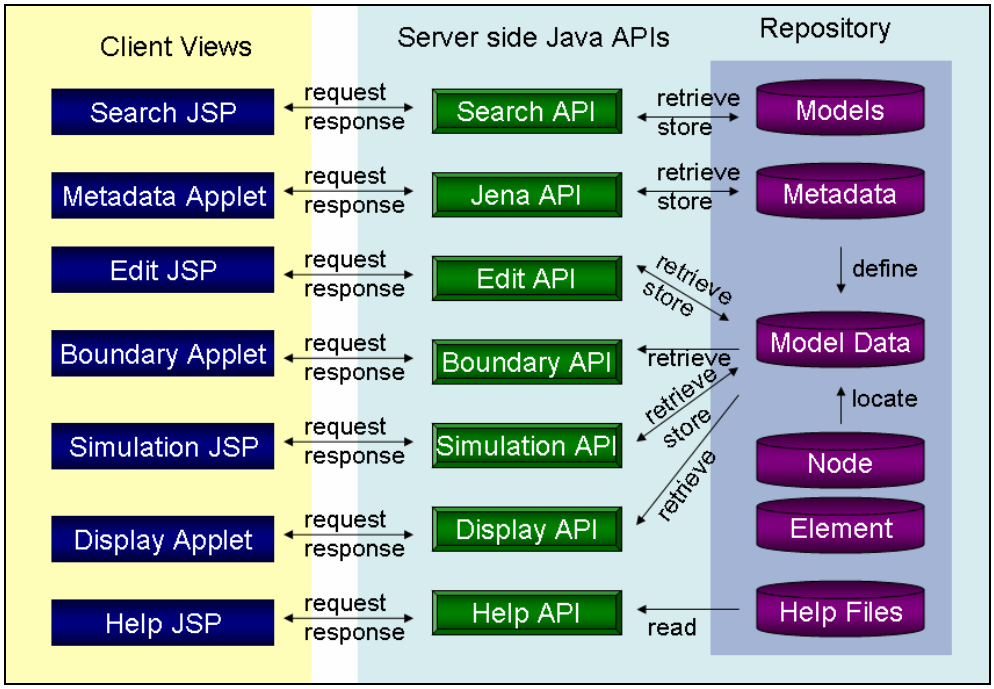


Figure 4.6. Java based server and client communication

4.3.5.2 Search View

Search View is introductory display for users after log in process is successfully completed (Figure 4.7). In this *View*, the user can search, display, create, rename, and delete models. Simple keyword searches and advanced searches are the two search options provided. Each model has a general description, which gives a short overview of the model. Users also can restrict their search using start and end time of the simulation or duration of simulation. Once the desired model has been found the user can perform several operations on it. A selected model can be copied, renamed, or deleted. There is a default model for each user (*wbsModel*) so that the user does not need to start from scratch. The user can simply make a copy of the default model and subsequently change the desired portions according to the desired modeling objective.

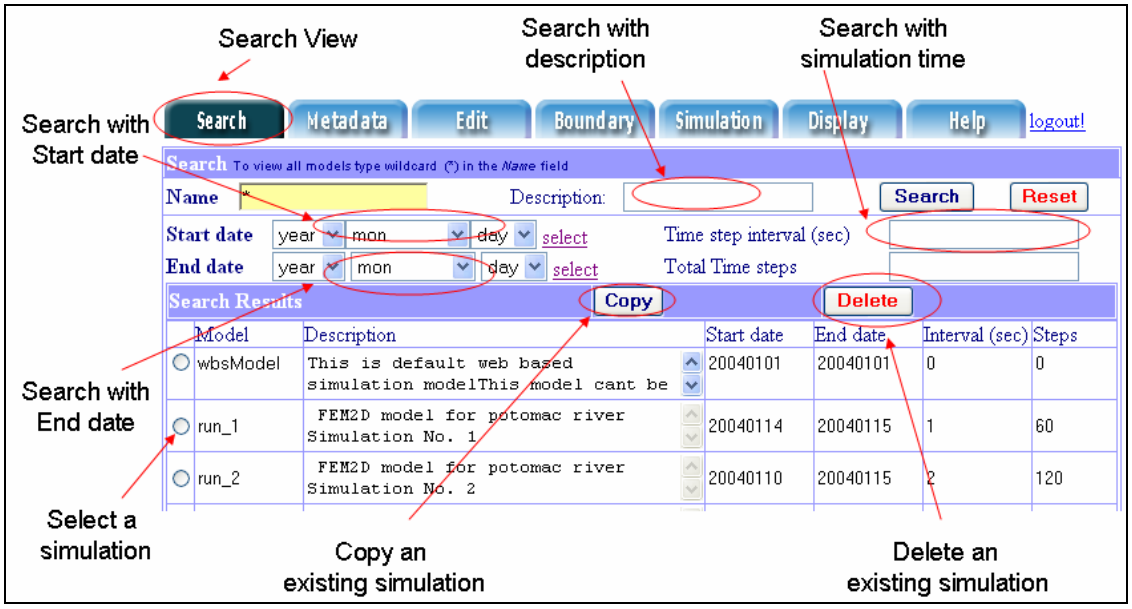


Figure 4.7. Interface for search, copy or delete of simulations

4.3.5.3 Metadata View

After selecting a model, the user should click on the *Metadata View* tap to enter or update model metadata (Figure 4.8). The first step is to select the data type for which metadata will be created. The display screen is divided into three parts: (1) the left side of the screen shows the ontology classes, (2) the right side of the screen shows properties for each instance, and (3) where the instances of the classes are shown. The user must create metadata for the root metadata class "*MD_Metadata*". The top part of the screen has two buttons "*Update*" and "*Download*". The "*Download*" button if clicked will download a metadata instance file in Resource Description Framework (RDF) format into the client machine. The "*Update*" button is for updating the metadata instances in the server database. The user can create any desired number of instances by clicking the "*C*" button and also can delete it using the "*X*" button. OWL data type properties can be created or deleted by using the "*C*" and "*X*" button respectively. OWL object type properties can be selected from the instance of other classes. If the "*C*" button is clicked a popup screen will appear and the instances of the respective class can be selected from that popup window. Ontology instances may contain Resource Description Framework Schema (RDFS) label, comments and annotation properties [76]. The *Metadata View* also supports creation of these annotation properties. Finally, the user can mark and select the core components by using the "M" button and the "core" check box which are located in the top of the "class" panel. This feature can help the user to select and view a small subset of the ontology based on their momentary needs. The *Metadata View* uses the frame-based-knowledge-presentation approach where knowledge is organized into chunks called frames [77]. Frames can be represented as objects (or classes) in object oriented programming languages such as JAVA. The well known ontology editor Protégé

with OWL plug-in [78], is one example application that too uses the frame-based-knowledge-representation approach. Although most of our ontologies were created using the Protégé ontology editor with OWL plug-in support, it is not possible to run Protégé as a JAVA Applet. Therefore, a GUI has been developed to facilitate the use of metadata inside the WBS. This GUI, or WBS component, is based on the JAVA Swing technology [79] and is implemented as a Java Applet to provide WBS operation via web browsers.

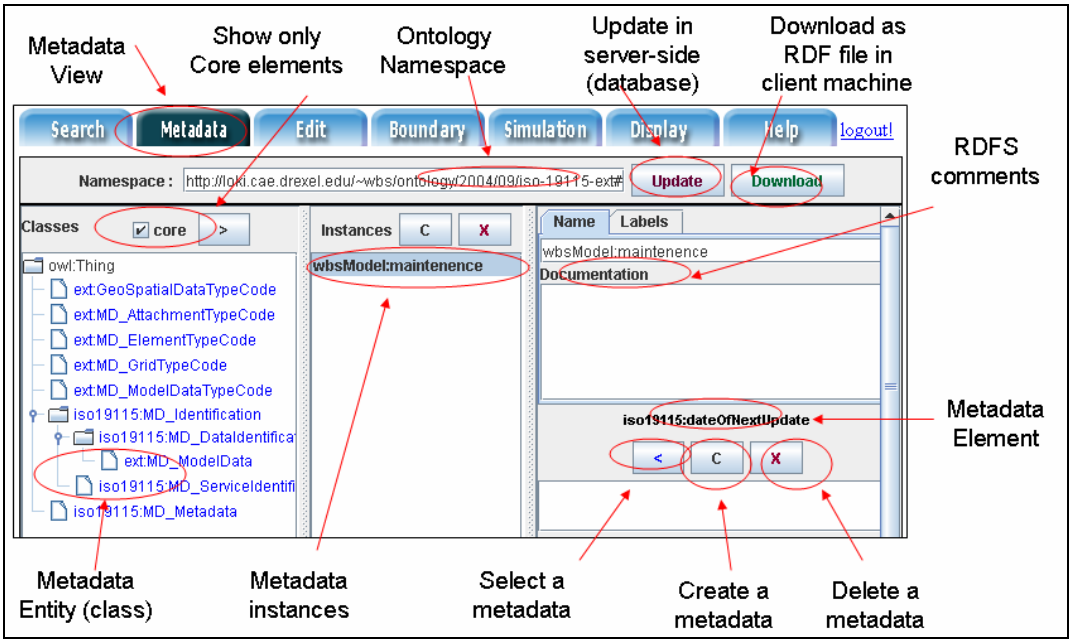


Figure 4.8. Interface for create, display, upload, or download metadata

4.3.5.4 Display View

The *Display View* was designed to display model data using two different types of plots: (1) contour plots and (2) time series charts. Both of these plot types are displayed by JAVA Applets based on the JAVA Swing classes. In contour plot mode, the user first needs to select a data type to be displayed (Figure 4.9). If the data is time dependent, a slide bar will appear to facilitate scrolling through time staring at

zero for the beginning of the simulation. Once the selection of time is completed, the user can click on the "*Plot Contour*" button to display a contour plot. The user can also zoom in and out of the contour plot by using the mouse. The right side of the screen shows the contour legend and the bottom of the screen displays the scale of the plot. For future versions of this software, a downloadable animation facility using the Graphic Interchange Format (GIF) is planned to add. The user also can show and hide the numerical grid in the contour display.

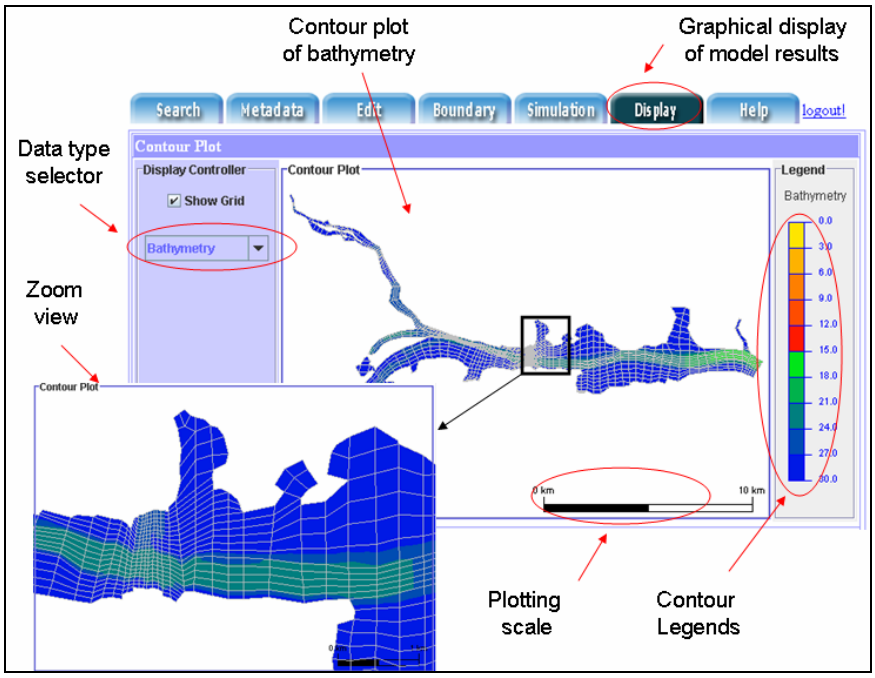


Figure 4.9. Interface for display of model data as contour plot

Display View also includes a time series plot of the model data at the bottom of the screen (Figure 4.10). The user must first select the data type, then insert desired node number, and then hit the "*Plot*" button after which a time series chart will be displayed on the right side of the screen. The time scale is automatically adjusted based on the time

series length. Also, the user can zoom in or out to get a clearer view of the chart plot using the mouse buttons.

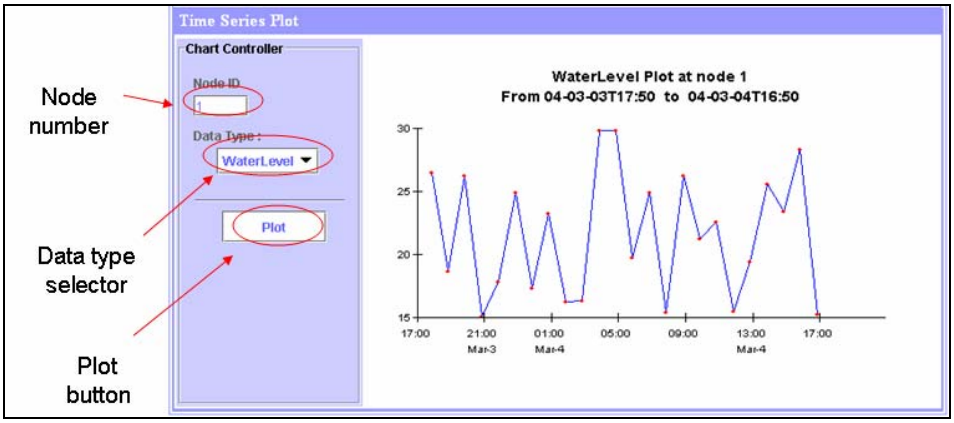


Figure 4.10. Interface for display of model data as time series plot

4.3.5.5 Edit View

Edit View was designed to help creating, displaying or deleting model data (Figure 4.11). The first thing the user needs to do is to select the data type and then hit the "*Display*" button to see data in text form. The default time indicates zero as the beginning of the simulation. However, the user can select different time steps based on the start of simulation for temporal type data. The user can edit any data value and click the "*Update*" button when done with the editing, which will update the model data in a database located in the simulation server. The "*Download*" button helps to download data in plain ASCII text to the client computer. As it is very tedious to update or edit thousands of data in the web forms, the user can upload data from a pre-existing file located in the client machine using the "*Upload*" button. Clicking on the "*Browse*" button will open a file selector in which the user can select the desired file for uploading.

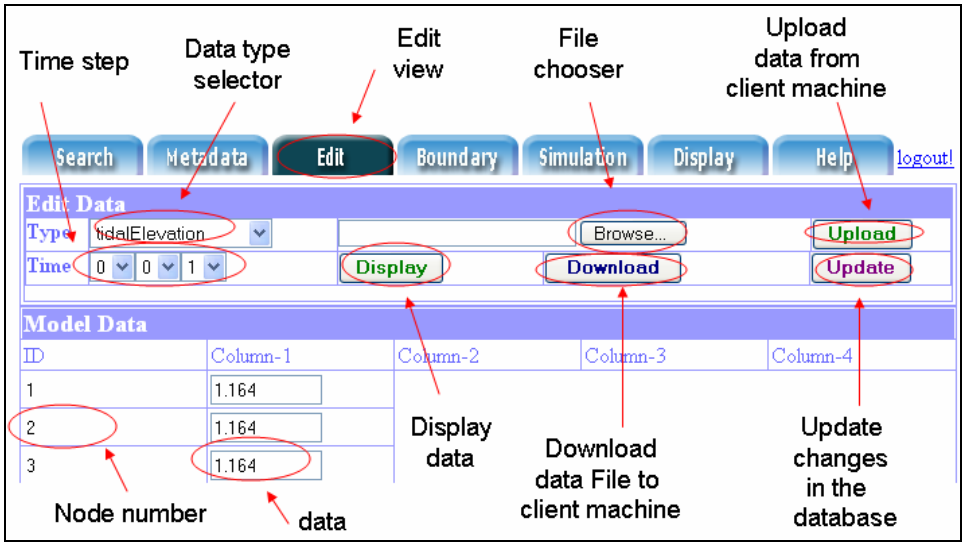


Figure 4.11. Interface for edit, uploads, download, and display of model data

4.3.5.6 Boundary View

Boundary View helps the user to setup the boundary constraints of the model. Figure 4.12 shows a snapshot of *Boundary View* to create and edit water level boundary (open boundary at downstream side) data of the model. Water level boundary data are generated from tidal curves which follow a pattern similar to sine curves. At present water level profile is assumed as a regular sine curve pattern. Amplitude, phase, and period of the curve can be changed by three horizontal scroll bars located in the left side of the graphic.

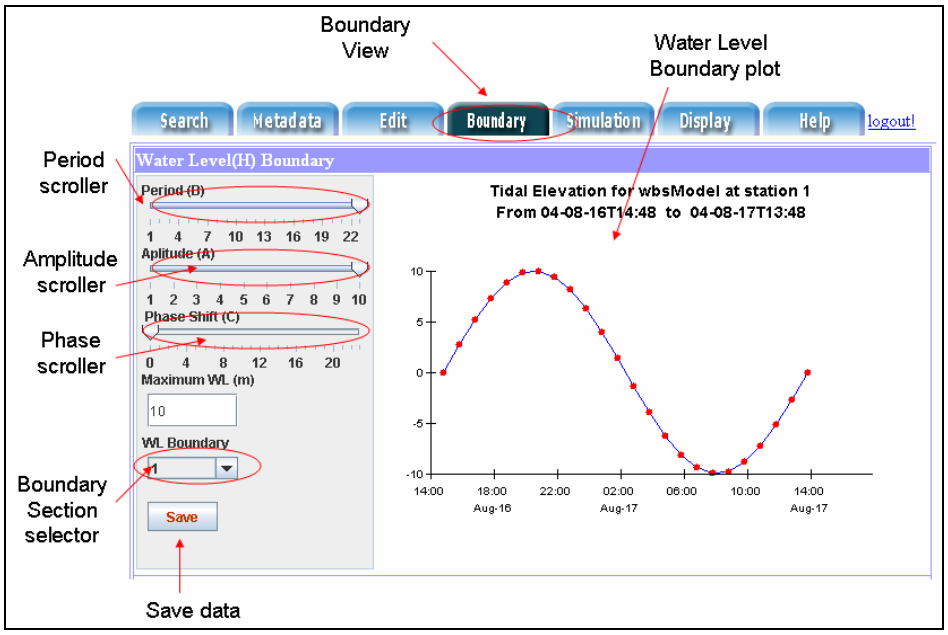


Figure 4.12. Interface for set up boundary constraints on upstream water level

Similar to water level boundary, discharge boundary data also can be generated with the aid of a graphical tool (Figure 4.13). First, the user will choose a certain inflow boundary section to create inflow data for the model. The top portion of the window shows overall view of the inflow discharge, and the bottom portion of the window shows a zoomed view of a selected region. The user can also control the selected region using the control buttons located in the left side of the *Boundary View*. Moreover, the user can even move any data point in the chart using the mouse upon which the inflow data will be generated according to the mouse position in the inflow chart. Finally, the user can save the inflow data into the database located in server machine by clicking the "*Save*" button.

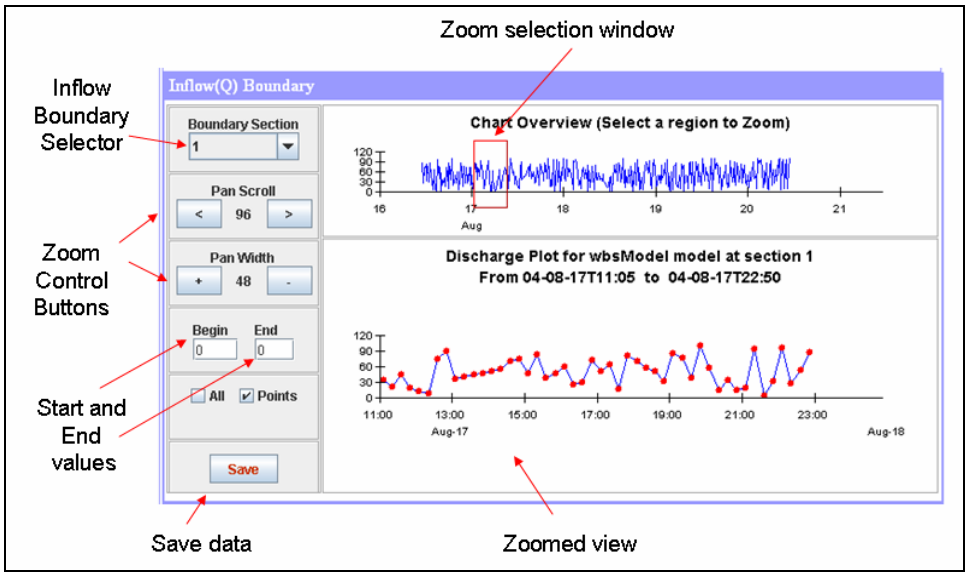


Figure 4.13. Interface for downstream discharge boundary

4.3.5.7 Simulation View

Simulation View helps user to run the model in the server (Figure 4.14). The user can select the desired time step for the simulation and then start the simulation by clicking the "*Start*" button. The user can interrupt the simulation anytime using the "*Stop*" button. The progress of the simulation run will be displayed in the bottom part of the screen based on an automatic 1-minute refresh rate (it is done to reduce Applet-Servlet communication), which can be accessed by clicking the "*Result*" button. Once the simulation is finished, the user can see the results as contour plot in *Display View* or as text data in *Edit View*.

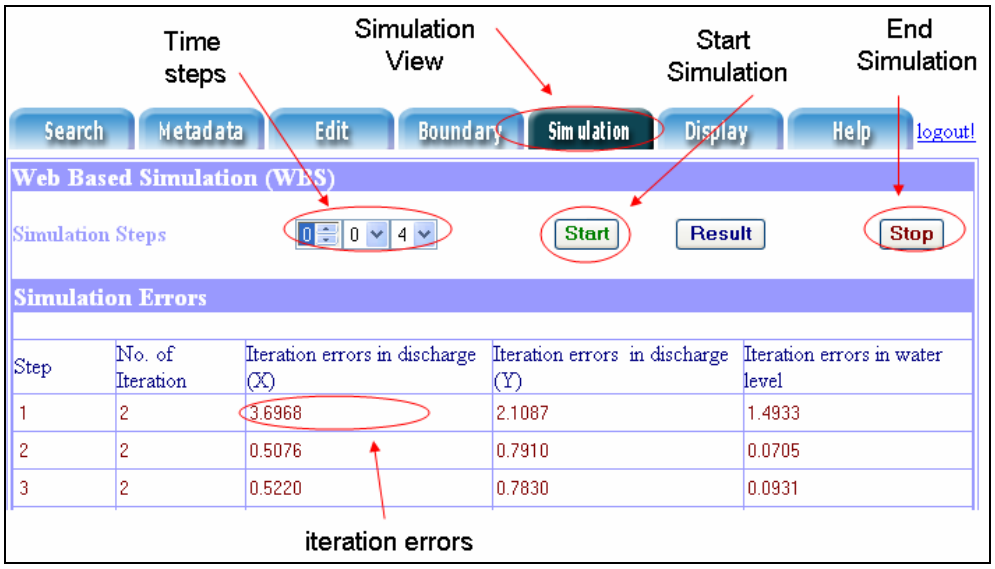


Figure 4.14. Interface for simulate of model

4.3.5.8 Help View

Help View provides different suggestions and tips of the web based simulation system (Figure 4.15). Users can select a topic related to their need and click on the "*Show help*" button. The help page will be displayed in the button part of the screen. Although most parts of the simulation environment are self explanatory, users may still find this section beneficial.

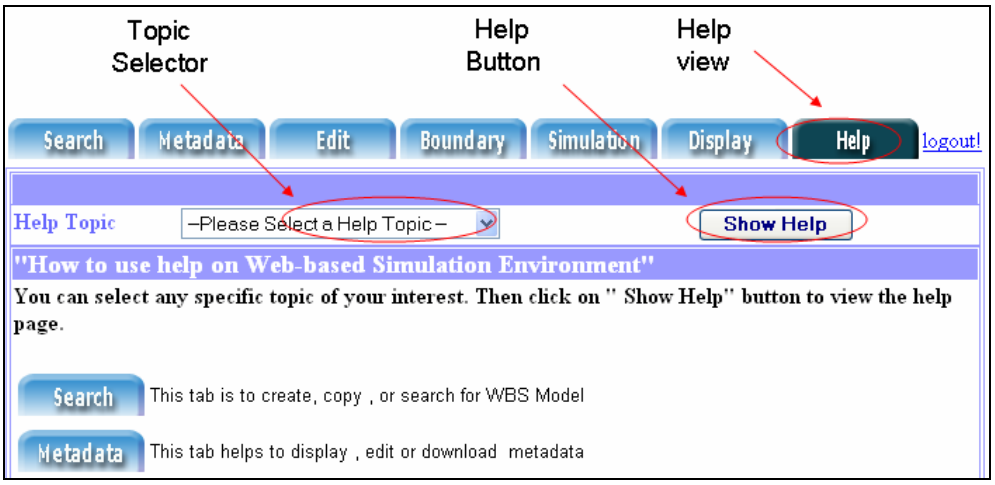


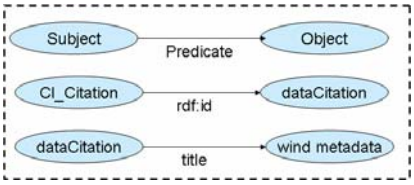
Figure 4.15. Interface for provide general help for the system

4.3.6. Repository System

As outlined previously, the WBS system is designed not only to host and provide access to numerous runs using the same numerical code operating on a single grid, but also to use different grids for the same domain or even provide access to different codes. Considering that one might want to execute a set of simulations that use one code and one grid but different boundary conditions, one quickly realizes that there are potentially many hundreds or even thousands of model runs that need to be stored. The data sets could either be stored in a flat file system or in a database. For a large number of data sets, however, storage in databases is preferable because querying and retrieving is more efficient when compared to using a flat file system. For this reason, the object relational database, PostgreSQL [75] is chosen from a group of possible products such as ORACLE [80], MS ACCESS [81], MySQL [82], to name just a few but prominent programs. The reason to decide on PostgreSQL is that it is a public domain code, i.e. free of charge, that it provides sufficient features that is needed (interface with Jena for example), but at the same time offering a much smaller administrative overhead when compared to ORACLE,

which is the leading product on the relational database market, but at a heavy licensing cost.

As mentioned above, Jena, a JAVA framework is used for developing semantic web applications developed by HP Labs semantic web research [83]. Jena is an open source package that has an OWL Application Programming Interface (API), and that also supports rule based interface engines. Jena has been used to create instances for model metadata that are based on the simulation ontology. The instances of the simulation ontology are expressed as Resource Description Framework (RDF) in either RDF/XML format or N-3 triple format. Although RDF/XML is the official serialization technique for RDF data, the N-triple format is preferable when using medium to large databases [83]. Figure 4.16 shows an example using a RDF graph model that depicts the serialization and storage of RDF triples in a database. For example, the ontology class "*CI_Citation*" has a property "*title*" to describe a dataset. If "*CI_Citation*" has an instance "*dataCitation*", it can be represented in the RDF graph using the prefix "*rdf:ID"* (Figure 4.16(a)). RDF/XML serialization of this RDF graph is shown in Figure 4.16(b), while Figure 4.16(c) shows how this RDF/XML can be stored in a relational database with tables that contain subject, object and property.



(a)

```
<CI_Citation rdf:id="dataCitation">
<title>wind Metadata<title>
</CI_Citation>
```

(b)

Subject	Predicate	Object
CI_Citation	rdf:id	dataCitation
dataCitation	title	wind metadata

(c)

Figure 4.16. RDF triple graph, (b) RDF/XML Document , and (c) Store RDF/XML into database

Different database tables have been created to store different input and output data components of the hydrodynamic model. Figure 4.17 shows a relationship among different tables in the database. The "*Models*" table stores some common information of each simulation: (1) unique ID for the simulation, (2) a title (3) a short description, (4) start and end date and time, (5) length of each time step. Numerical grid information has been stored in two different tables, namely "*Element*" and "*Node*". Nodal information such as unique identity and x, y, and z coordinates of each node has been stored in the "*Node*" table. The "*Element*" table stores data in 3 columns: (1) *elementID* to provide a unique identity, (2) *numberOfNodes* to estimate the total nodes, and (3) *nodeID* to identify the nodes of each element. The "*ModelData*" table stores information about different types of model data such as discharge, water level, viscosity, dispersion etc.

”*ModelData*” table stores different components of model data such as name, unique identity number, anchor point with the grid, and data values. Finally, the “*MetadataModelData*” table stores a small subset of the metadata of the model data namely, temporal characteristics, type of attachment with the grid, and number of data columns. This “*MetadataModelData”* table has been created for faster access and retrieval of the “*ModelData*”. For example, to retrieve velocity data from the “*ModelData*” table, it is needed to know: (1) attachment type (node or element), (2) temporal characteristics (true or false), (3) total number of data columns (two for 2D velocity components), and (4) variable name (“velocity”). The “*MetadataModelData”* table can be accessed directly through JAVA using a Java Database Connectivity (JDBC) adapter. This saves a substantial amount of time when the call to the Jena program is made to read metadata instances every time needed to retrieve model data.

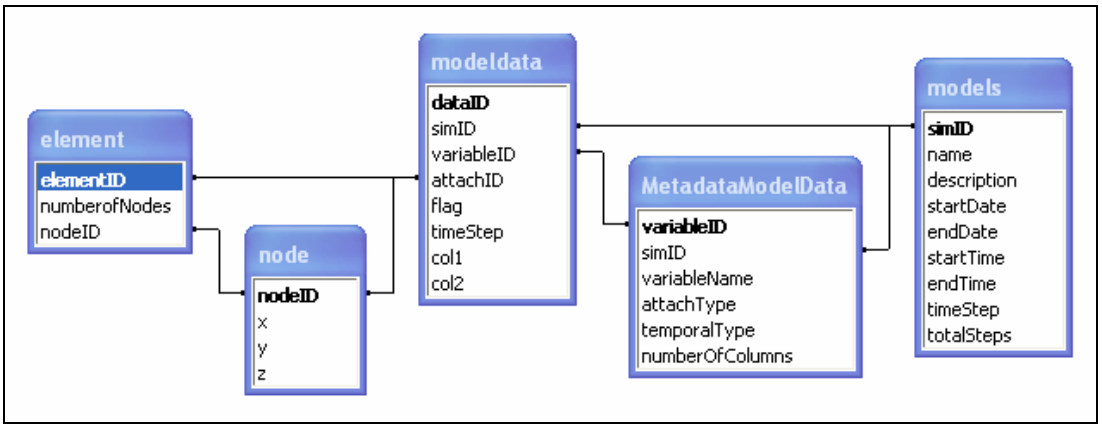


Figure 4.17. Relationship of tables in database for hydrodynamic model

4.4. Summary

In this chapter the concept for a web-based simulation system for simulating hydrodynamic processes is introduced. The main thrust of this study is to provide a possible new approach for remote clients to explore the computational facility of powerful servers to simulate large scale hydrodynamic processes. To this end the metadata profile from chapter 3 that describes hydrodynamic model data based on the ISO 19115:2003 metadata standard have been utilized. In addition, a simulation ontology to search and retrieve this metadata information and to reuse it as simulation domain knowledge within the WBS has been developed. The WBS system architecture is based on the *Model-View-Controller (MVC)* approach, which entails a set of 7 components that not only reflect the system logic, but also guides the user via a GUI through the different stages of the WBS application. The advantage of this approach is i) that it separates the business logic from its representation, and ii) that is entirely modular allowing additional components to be added without having to redesign the WBS system.

In order to store the model run instances (in conjunction with the associated metadata sets), a Jena API is used to move and retrieve these instances from a relational data base system. The decision to use PostgreSQL was based on the desire to have a free of charge program, the advantage of having a medium seized system with a reduced administrative overhead, and a list of features that makes the use of this application ideal for web-based systems that require access via the internet.

The sample numerical code using a two dimensional shallow water equation with finite element formulation was wrapped into a Java environment and was placed on

the web Server, which in turn run the Tomcat server which utilizes technologies such as Java Servlet and Java Server Pages (JSP). The client side graphical user interfaces (GUI) implements a Java Applet or HTML forms to display different presentations (*Views*) of the model.

Although this study shows that it is possible to execute large scale numerical model simulations using a web based simulation architecture, there are still many challenges to overcome in case more complex numerical models are targeted for WBS. A possible future direction of the study could be the development of a web based system for three dimensional water quality or subsurface transport models, or the development of efficient search facilities using software agents, and building robust systems using Enterprise Java Bean (EJB) technology for enterprise applications.

CHAPTER 5: CONCLUDING REMARKS

The study reported in this thesis addresses several important scientific and engineering issues.

Among these issues are:

1. The need of expressing metadata in both human and machine readable format.

2. The necessity of developing a metadata community profile which can be used for different types of hydrodynamic models to as data descriptors.

3. The necessity of developing a simulation environment to share both data and numerical models among the users located anywhere in the world.

4. The need of a simple but elegant tool which can act as a graphical interface for the user and is capable of storing, creating, searching and running a numerical model.

Chapter 2 focuses of the first issue to select a suitable publishing format for metadata of hydrodynamic model which can be understand by both human and machine. Web Ontology Language (OWL) has been selected as metadata publishing language which serves this purpose best. But, geographic information metadata, ISO norm has published its conceptual model in UML and it becomes a challenging task to convert this UML model into OWL ontology because both of these specifications have many conceptual differences.

A mapping approach for geographic information metadata, ISO norm 19115:2003, from its conceptualization in UML into an ontology using the Web Ontology Language, OWL was examined. In addition, a step by step conversion process from ISO 19115:2003 UML model to OWL ontology was demonstrated addressing each

of the items that, in our opinion, needed a specific mapping rule. While many of the dissimilarities have been resolved and a mapping rule has been derived, it should be noted that a complete mapping has not been achieved and that the present specification of UML does not permit a complete conversion of an object oriented concept into a knowledge based concept. The ISO 19115:2003 ontology was made available for public use and the expectation is that it will be used and also tested with a number of extensive real world examples for future modification and corrections.

Chapter 3 examines the second important issue which was to develop suitable metadata sets that can serve as a profile for hydrodynamic modeling community. Chapter 3 proposes a metadata structure for hydrodynamic model data that can potentially pave the way of how to inter- and exchange data between different types of codes and model domains. ISO 19115:2003 metadata norm has been selected as a basis to generate the metadata framework for hydrodynamic codes. Although ISO 19115:2003 defines more than 300 metadata elements, it was not able to provide all the necessary descriptions for hydrodynamic model data. It has been shown that it is a need to extend the overall coverage of the ISO 19115:2003 in various aspects and a metadata community profile for hydrodynamic models has been created based on the extension guideline of ISO. In particular, descriptions for elementary components like a numerical grid and linkage between geospatial reference points (nodes) and associations of these points with hydrodynamic variables has been demonstrated. It is clear that the scope of this framework is extensive and will need to be filled over time. Yet this approach can aid in overcoming the currently existing lack of data interoperability among different hydrodynamic codes.

It has also suggested the use of Web-Ontology language OWL as an encoding medium for the hydrodynamic community profile, because it best captures the semantic of metadata for both human and machine understandable format. Ontology based approach has chosen to create community profile, because it provides both flexibility and ease for change, extent and update of this framework.

Chapter 4 studied the third and fourth issues to the development strategy of web based simulation architecture for hydrodynamic processes. The main focus of this study is to provide a possible new approach for remote client to explore the computational facility of powerful servers to simulate large scale hydrodynamic processes. A simulation ontology has been created which includes both model data and metadata components and their relationships. This simulation ontology can help to search and retrieve model data and to represent the domain knowledge within the WBS.

The WBS system architecture has developed based on the concept of *Model-View –Controller (MVC)* which separates business logic (*Model*) from presentation logic (*Views*) to the user. Based on *MVC* pattern, a set of 7 *View* components has been developed that not only reflects the system logic, but also guide user via a GUI through the different stages of the WBS application. The simulation code was wrapped in a platform independent language; Java and execution of models was done by Java client-server based architecture. Model runs in a Java based web Server; Tomcat based on Java Server side technology such as Java Servlet and Java Server Pages (JSP). The client side Graphical User Interfaces (GUI) implements Java Applet or HTML forms to display different presentations (*Views*) of the model.

An object relational database management system PostgeSQL has been selected for persistent storage of the model data and metadata. The instances of the ontology were serialized as RDF triples and store in the database using Java Database Connectivity (JDBC) interface. The decision to use PostgreSQL was based on the desire to have a free of charge program, the advantages of having a medium seized system with a reduced administrative overhead, and a list of features such as persistent storage of objects that makes the use of this application ideal for Web-Based systems.

Although this study shows that it is possible to simulate a large scale numerical model using web based simulation architecture, there are still many challenges to overcome for more complex numerical models. Future direction of the study could be development WBS for three dimensional more complex water quality models, development of efficient agent based searching facility, and building of robust systems using Enterprise Java Bean (EJB) technology for enterprise applications etc.

The achievements of this study, in particularly:

- Implementation of geographical information metadata ISO 19115:2003 norm using Web Ontology Language (OWL)
- Development of a metadata community profile for hydrodynamic codes using ISO 19115:2003
- Development of a tool to simulate a large scale hydrodynamic model using web browser.
- As an application of these web based tools, a two dimensional finite element model was successfully tested for simulating the flow in upper Potomac estuary.

LIST OF REFERENCES

[1] http://www.epa.gov/docs/ostwater/BASINS/

[2] http://www.w3.org/XML/

[3] http://www.w3c.org/RDF/

[4] http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[5] Ahmed, K., et al., *Professional XML Metadata*. Profession. 2001, Birmingham: Wrox Press Ltd.

[6] Sen, A., *Metadata management: past, present and future*. Decision Support Systems, 2003: p. 1-23.

[7] http://www.harmonit.org/documents/pubview.php?view=all&doctype=all

[8] http://www.ems-i.com/SMS/SMS_Overview/sms_overview.html

[9] http://water.usgs.gov/software/genscn.html

[10] http://www.netcoast.nl/tools/rikz/delft3d.htm

[11] http://my.unidata.ucar.edu/content/software/netcdf/index.html

[12] http://hdf.ncsa.uiuc.edu/

[13] Bossomaier, T.R.J. and D. Green, *Online GIS and spatial metadata*. 2001, New York: Taylor & Francis.

[14] http://dublincore.org/

[15] FGDC, *Content Standard for Digital Geospatial Metadata (CSDGM)*. 1998.

[16] ISO, *Geographic Information - Metadata (ISO 19115:2003)*. 2003.

[17] ISO, *Geographic Information - Metadata - Implementation Specification (ISO 19139.3)*. 2003.

[18] http://www.omg.org/technology/documents/formal/uml.htm

[19] http://www.w3.org/XML/Schema

[20] Hendler, J.A., *XML and the Semantic Web.* XML Journal, 2002.

[21] Klein, M., *XML, RDF, and Relatives,* in *IEE Intelligent System*. 2001. p. 26-28.

[22] http://www.w3.org/TR/rdf-concepts/#ref-rdf-semantics

[23] http://www.w3.org/TR/rdf-schema/

[24] Ahmed, K., et al., *Professional XML Metadata*. 2001, Birmingham: Wrox Press. ix, 567 p.

[25] http://www.w3.org/TR/2004/REC-owl-features-20040210/

[26] Berners-Lee, T., J.A. Hendler, and O. Lassila, *The Semantic Web.* Scientific American, 2001. 284(5): p. 34-43.

[27] http://www.w3.org/TR/2004/REC-owl-guide-20040210/

[28] Islam, A.S., et al., *Implementation of the Geographic Information - Metadata (ISO 19115:2003) Norm using the Web Ontology Language (OWL).* Transactions in GIS, 2004. submitted for publication.

[29] Bermudez, L. and M. Piasecki, *Community Metadata Profiles for the Semantic Web.* Geoinformatica Journal, 2004. Submitted for publication.

[30] Kamel Boulos, M.N., A.V. Roudsari , and E.R. CarSon *Towards a Semantic Medical Web: HealthCyberMap's Dublin Code Ontology in Protege-2000*. in *Fifth International Protege Workshop, Sowerby Centre for Health Informatics at NewCastle (SCHIN)*. 2001. Newcastlem, England.

[31] Wariyapola, P.C.H., et al. *Ontology and Metadata Creation for the Poseidon Distributed Coastal Zone Management System*. in *IEEE Forum on Research and Technology Advances in Digital Libraries*. 1999. Baltimore, Maryland.

[32] Hanschuch, S. and S. Staab, *CREAM: Creating Metadata for the Semantic Web.* Computer Networks, 2003. 42: p. 579-598.

[33] http://water.usgs.gov/nrp/gwsoftware/modflow.html

[34] http://www.mmm.ucar.edu/mm5/mm5-home.html

[35] http://www.ccsm.ucar.edu/

[36] http://walrus.wr.usgs.gov/transport/

[37] Miller, J.A., et al., *Research and coomercial opportunities in Web-Based Simulation.* Simulation Practice and Theory, 2001. 9(1-2): p. 55-72.

[38] Kuljis, J. and R.J. Paul, *An appraisal of web-based simulation: whither we wander?* Simulation Practice and Theory, 2001. 9(1-2): p. 37-54.

[39] Miller, J.A., A.F. Seila, and X. Xiang, *The JSIM web-based simulation environment.* Future Generation Computer Systems, 2000. 17(2): p. 119-133.

[40] F. Howell, R.M. *simjava: a discrete event simulation library for Java*. in *1988 International Conference on Web-Based Modeling & Simulation, The Society for Computer Simulation International*. 1988. San Diego, CA.

[41] Sarjoughian, H.S. and B.P. Zeigler. *DEVSJAVA: Basis for a DEVS-based collaborative M&S environments*. in *1988 International Conference on Web-Based Modeling & Simulation, The Society for Computer Simulation International*. 1988. San Diego, CA.

[42] Wiedemann, T. *Simulation application service providing (SIM-ASP)*. in *Winter 2001 Simulation Conference*. 2001. Arlington, VA, USA.

[43] Olsen, K.B. *Websim3d: A Web-based System for Generation, Storage and Dissemination of Earthquake Ground Motion Simulations*. in *2003 AGU Fall Meeting*. 2003. San Francisco, CA.

[44] Fensel, D., *Spinning the semantic Web: bringing the World Wide Web to its full potential*. 2003, Cambridge, Mass.: MIT Press. xxiii, 479.

[45] http://neptune.irit.fr/Biblio/03-09-04.pdf

[46] http://www.semanticweb.org/SWWS/program/full/paper1.pdf

[47] http://www.w3.org/TR/xslt

[48] http://www.exff.org./

[49] Kogut, P., et al., *UML for Ontology Development.* Knowledge Engineering, 2001. 17(1): p. 61-64.

[50] http://www.omg.org/technology/documents/formal/mof.htm

[51] Guizzardi, G., H. Herre, and G. Wagner. *Towards Ontological Foundations for UML Conceptual Models*. in *1st International Conference on Ontologies, Databases and Applications of Semantic (ODBASE 2002)*. 2002. Irvine,California, USA.

[52] http://protege.stanford.edu/plugins/xmi/download/2003-06-12%20SMI-Knublauch.pdf

[53] http://www.omg.org/technology/documents/formal/xmi.htm

[54] http://java.sun.com/products/jmi/index.jsp

[55] http://www.omg.org/cgi-bin/doc?ad/2003-03-40

[56] Baclawski, K., et al., *Extending the Unified Modeling Language for ontology development.* Software System Model, 2002. 1: p. 1-15.

[57] ISO, *Geographic Information - Conceptual Schema Language (ISO 19103)*. 2004.

[58] http://www.w3.org/XML/

[59] ISO, *Geographic Information - Temporal Schema (ISO 19108:2002)*. 2002.

[60] ISO, *Geographic Information - Spatial Referencing by Coordinate System (ISO 19111)*. 2000.

[61] Miller, J.A. and P.A. Fishwick. *Investigating Ontologies for Simulation Modeling*. in *37th Annual Simulation Symposium*. 2004. Arlington, VA.

[62] Katopodes, N.D., *A Dissipative Galerkin Scheme for Open-Channel Flow.* Journal of Hydraulics, ASCE, 1984. 110(4): p. 450-466.

[63] Piasecki, M. and N.D. Katopodes, *Control of Contaminant Releases in Rivers and Estuaries Part I: Adjoint Sensitivity Analysis.* Journal of Hydraulic Engineering, ASCE, 1997. 123(6): p. 486-492.

[64] Liang, S., *Java(TM) Native Interface: Programmer's Guide and Specification.* 2 ed. 1999: Addison-Wesley Pub Co.

[65] http://hfi-l2.in2p3.fr/fortranOO_EA.pdf

[66] Zeng, H., et al., *A web-based simulation system for transport and retention of dissolved contaminants in soil.* Computers and Electronics in Agriculture, 2002. 33(2).

[67] http://www.omg.org/cgi-bin/doc?formal/04-03-01

[68] http://java.sun.com/products/ejb/

[69] http://java.sun.com/developer/technicalArticles/ebeans/corba/

[70] http://java.sun.com/products/servlet/

[71] Chen, S. and F.Y. Lu, *Web-based Simulation of Power Systems.* IEEE Computer Applications in Power, 2002. 15(1): p. 35-40.

[72] Kurniawan, B., *Java for the Web with Servlets, JSP, and EJB*. 2002, Indianapolis, USA: New Riders Publishing.

[73] http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html

[74] Stelting, S. and O. Maassen, *Applied Java Patterns*. 2002, Sun MicroSystems Press: California, USA.

[75] http://www.postgresql.org/docs/

[76] http://www.w3.org/TR/2000/CR-rdf-schema-2000327

[77] Minsky, M., *The Psychology of Computer Vision*, ed. P. Winston. 1975, New York: McGraw-Hill. 211-277.

[78] http://protege.stanford.edu/plugins/owl/tutorial/index.html

[79] http://java.sun.com/products/jfc/

[80] http://www.oracle.com/database/index.html

[81] http://office.microsoft.com/home/office.aspx?assetid=FX01085791

[82] http://www.mysql.com/products/mysql/

[83] http://jena.sourceforge.net/documentation.html

APPENDIX 1: EXAMPLE METADATA INSTANCE DOCUMENT FOR A NUMERICAL GRID

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:iso19115="http://loki.cae.drexel.edu/~wbs/ontology/2004/08/iso-19115#"
   xmlns:iso19103="http://loki.cae.drexel.edu/~wbs/ontology/2004/08/iso-19103#"
        xmlns:iso19111="http://loki.cae.drexel.edu/~wbs/ontology/2004/08/iso-19111#"
   xmlns:iso19108="http://loki.cae.drexel.edu/~wbs/ontology/2004/08/iso-19108#"
   xmlns="http://loki.cae.drexel.edu/~wbs/data/wbsModel/grid_metadata.xml#"
        xml:base="http://loki.cae.drexel.edu/~wbs/data/wbsModel/grid_metadata.xml">
  <iso19115:MD_Metadata rdf:ID="Grid_MD_Metadata">
   <iso19115:language>en</iso19115:language>
   <iso19115:characterSet rdf:resource="#utf8" />
   <iso19115:date>20040804</iso19115:date>
   <iso19115:metadataStandardVersion>IS</iso19115:metadataStandardVersion>
   <iso19115:dataQualityInfo rdf:resource="#Grid_DQ_DataQuality"/>
   <iso19115:spatialRepresentationInfo rdf:resource="#Grid_MD_GeoSpatialData" />
   <iso19115:distributionInfo rdf:resource="#Grid_MD_Distribution"/>
   <iso19115:referenceSystemInfo rdf:resource="#Grid_Horizontal_MD_CRS"/>
   <iso19115:referenceSystemInfo rdf:resource="#Grid_Vertical_SC_CRS"/>
```

```
    <iso19115:contact rdf:resource="#Grid_CI_ResponsibleParty"/>
    <iso19115:fieldIdentifier>http://loki.cae.drexel.edu/~wbs/
data/wbsModel/grid_metadata.xml</iso19115:fieldIdentifier>
    <iso19115:metadataStandardName>ISO 19115</iso19115:metadataStandardName>
    <iso19115:identificationInfo rdf:resource="#Grid_DataIdentification" />
    <iso19115:hierarchyLevel rdf:resource="#_dataset"/>
    <iso19115:dataSetURI
rdf:resource="http://loki.cae.drexel.edu/~wbs/data/wbsModel/grid.htm"/>
          <iso19115:distributionInfo rdf:resource="#Grid_MD_Distribution" />
  </iso19115:MD_Metadata>

<iso19115:MD_Distribution rdf:ID="Grid_MD_Distribution">
<iso19115:transferOptions rdf:resource="#Grid_MD_DigitalTransferOption" />
<iso19115:distributionFormat rdf:resource="#Grid_MD_Format" />
</iso19115:MD_Distribution>

<iso19115:MD_Format rdf:ID="Grid_MD_Resource_Format">
<iso19115:name>Tab-delimated</iso19115:name>
</iso19115:MD_Format>

<iso19115:MD_DigitalTransferOption rdf:ID="#Grid_MD_DigitalTransferOption">
<iso19115:unitsOfDistribution>single-ASCII-file</iso19115:unitsOfDistribution>
<iso19115:transferSize>0.12</iso19115:transferSize>
```

<iso19115:onLine rdf:resource="Grid_CI_OnlineResource" />

</iso19115:MD_DigitalTransferOption>

<CI_OnlineResource rdf:ID="Grid_CI_OnlineResource">

<iso19115:linkage>http://loki.cae.drexel.edu/~wbs/data/wbsModel/grid.htm</iso19115:li

nkage>

<iso19115:protocol>http</iso19115:protocol>

<iso19115:name>Numerical Grid File</iso19115:name>

<iso19115:description>Contains node and element data for a numerical

grid</iso19115:description>

<iso19115:function rdf:resource="download" />

</CI_OnlineResource>

<iso19115:MD_DataIdentification rdf:ID="Grid_DataIdentification">

<iso19115:language>en</iso19115:language>

<iso19115:characterSet rdf:resource="#utf8" />

<iso19115:status rdf:resource="#completed"/>

<iso19115:spatialRepresentationType rdf:resource="#vector"/>

<iso19115:resourceConstraints rdf:resource="#Grid_MD_Constraint"/>

<iso19115:topicCategory rdf:resource="#inlandWaters"/>

<iso19115:resourceMaintenance rdf:resource="#Grid_MD_MaintenanceInformation"

/>

<iso19115:purpose>This grid was developed to simulate a two dimensional finite element model anyalysing flow in Potomac river</iso19115:purpose>

<iso19115:geographicBox rdf:resource="Grid_EX_GeographicBoundingBox" />

<iso19115:spatialResolution rdf:resource="#Grid_MD_Resolution" />

<iso19115:citation rdf:resource="#Grid_CI_Citation" />

<iso19115:resourceSpecificUsage rdf:resource="#Grid_MD_Usage" />

<iso19115:pointOfConact rdf:resource="#Grid_CI_ResponsibleParty"/>

<iso19115:resourceFormat rdf:resource="#Grid_MD_Resource_Format"/>

<iso19115:abstract>This document describes a two dimensional finite element grid. This grid has total of 1171 triangular type of elements and 1408 nodes</iso19115:abstract>

</iso19115:MD_DataIdentification>

<iso19115:MD_MaintenanceInformation rdf:ID="Grid_MD_MaintenanceInformation">

<iso19115:maintenanceAndUpdateFrequency rdf:resource="#asNeeded"/>

</iso19115:MD_MaintenanceInformation>

<iso19115:MD_Resolution rdf:ID="Grid_MD_Resolution">

<iso19115:equivalentScale>

<iso19115:MD_RepresentativeFraction rdf:ID="Grid_MD_RepresentativeFraction">

<iso19115:denominator>1</iso19115:denominator>

</iso19115:MD_RepresentativeFraction>

</iso19115:equivalentScale>

```
</iso19115:MD_Resolution>
<iso19115:CI_Citation rdf:ID="Grid_CI_Citation">
      <iso19115:resRefDate>
        <iso19115:CI_Date rdf:ID="Grid_CI_Date">
          <iso19115:date>20040804</iso19115:date>
          <iso19115:dateType rdf:resource="#publication"/>
        </iso19115:CI_Date>
      </iso19115:resRefDate>
      <iso19115:title>Two dimensional finite element grid</iso19115:title>
</iso19115:CI_Citation>
<iso19115:MD_Usage rdf:ID="Grid_MD_Usage">
   <iso19115:userContactInfo rdf:resource="#Grid_CI_ResponsibleParty"/>
   <iso19115:specificUsage>Used for general public</iso19115:specificUsage>
</iso19115:MD_Usage>
<iso19115:CI_Address rdf:ID="Grid_CI_Address">
  <iso19115:country>USA</iso19115:country>

<iso19115:electronicMailAddress>asi22@drexel.edu</iso19115:electronicMailAddress>
  <iso19115:administrativeArea>Pennsylvania</iso19115:administrativeArea>
  <iso19115:postalCode>19104</iso19115:postalCode>
  <iso19115:deliveryPoint>3141 Chestnut St, Room 280F</iso19115:deliveryPoint>
  <iso19115:city>Philadelphia</iso19115:city>
</iso19115:CI_Address>
```

```
 <iso19115:DQ_DataQuality rdf:ID="Grid_DQ_DataQuality">
  <iso19115:lineage>
   <iso19115:LI_Lineage rdf:ID="Grid_Li_Lineage">
    <iso19115:statement>This grid has developed from following sources: US
Department of Commerce, NOAA, Washington D.C., National Ocean Service Coast
Survey</iso19115:statement>
   </iso19115:LI_Lineage>
  </iso19115:lineage>
  <iso19115:scope>
   <iso19115:DQ_Scope rdf:ID="Grid_DQ_Scope">
    <iso19115:level rdf:resource="#_dataset"/>
   </iso19115:DQ_Scope>
  </iso19115:scope>
 </iso19115:DQ_DataQuality>
 <iso19115:CI_Citation rdf:ID="Grid_CI_Citation">
  <iso19115:title>Two dimensional finite element  grid</iso19115:title>
 </iso19115:CI_Citation>
 <iso19115:MD_Distribution rdf:ID="Grid_MD_Distribution">
  <iso19115:transferOptions>
   <iso19115:MD_DigitalTransferOptions rdf:ID="Grid_MD_DigitalTransferOptions">
    <iso19115:onLine rdf:resource="#Grid_CI_OnlineResource"/>
   </iso19115:MD_DigitalTransferOptions>
  </iso19115:transferOptions>
```

```
    <iso19115:distributionFormat>
     <iso19115:MD_Format rdf:ID="Grid_MD_Format">
          <iso19115:name>RDF</iso19115:name>
       <iso19115:version>1.0</iso19115:version>
     </iso19115:MD_Format>
    </iso19115:distributionFormat>
  </iso19115:MD_Distribution>
  <iso19115:MD_Constraints rdf:ID="Grid_MD_Constraint">
    <iso19115:useLimitation>The CFL should be less than one for accurate
result</iso19115:useLimitation>
  </iso19115:MD_Constraints>
  <MD_GeoSpatialData rdf:ID="Grid_MD_GeoSpatialData">
     <gridType rdf:resource="#unstructured"/>
          <verticalDiscritizationType    rdf:resource="#zCordinate" />
          <gridCoordinateSystem rdf:resource="iso19111:Cartesian" />
     <geoSpatialDataType rdf:resource="#grid"/>
     <elementType rdf:resource="#quadrilateral4"/>
     <iso19115:topologyLevel rdf:resource="#fullSurfaceGraph"/>
      <totalElements>1171</totalElements>
       <iso19115:geometricObjects>
       <iso19115:MD_GeometricObjects rdf:ID="Grid_GeometricObjects">
         <iso19115:geometricObjectType rdf:resource="#surface"/>
       </iso19115:MD_GeometricObjects>
```

```
  </iso19115:geometricObjects>
 <totalNodes>1408</totalNodes>
</MD_GeoSpatialData>
<iso19115:CI_ResponsibleParty rdf:ID="Grid_CI_ResponsibleParty">
  <iso19115:organisationName>Drexel University</iso19115:organisationName>
   <iso19115:positionName>Ph.D. Candidate</iso19115:positionName>
    <iso19115:contactInfo>
     <iso19115:CI_Contact rdf:ID="Grid_CI_Contact">
      <iso19115:address rdf:resource="#Grid_CI_Address"/>
      <iso19115:onlineResource>
       <iso19115:CI_OnlineResource rdf:ID="Grid_CI_OnlineResource">
         <iso19115:linkage>
          <iso19115:URL rdf:ID="Grid_URL"/>
         </iso19115:linkage>
       </iso19115:CI_OnlineResource>
      </iso19115:onlineResource>
      <iso19115:phone>
       <iso19115:CI_Telephone rdf:ID="Grid_CI_Telephone">
         <iso19115:voice>215-387-1560</iso19115:voice>
        <iso19115:fascimile>215-895-1363</iso19115:fascimile>
      </iso19115:CI_Telephone>
     </iso19115:phone>
    </iso19115:CI_Contact>
```

```
      </iso19115:contactInfo>
     <iso19115:individualName>Akm Saiful Islam</iso19115:individualName>
    <iso19115:role rdf:resource="#publisher"/>
 </iso19115:CI_ResponsibleParty>
 <iso19115:EX_GeographicBoundingBox
rdf:ID="Grid_EX_GeographicBoundingBox">
    <iso19115:westBoundLongitude>
      <iso19103:Angle rdf:ID="Grid_Angle_4">
        <iso19103:value>-77.08</iso19103:value>
              <iso19103:uom rdf:resource="#Grid_UnitOfAngle" />
        </iso19103:Angle>
      </iso19115:westBoundLongitude>
      <iso19115:southBoundLatitude>
        <iso19103:Angle rdf:ID="Grid_Angle_3">
          <iso19103:value>38.67</iso19103:value>
                <iso19103:uom rdf:resource="#Grid_UnitOfAngle" />
          </iso19103:Angle>
      </iso19115:southBoundLatitude>
       <iso19115:northBoundLatitude>
          <iso19103:Angle rdf:ID="Grid_Angle_2">
             <iso19103:value>38.92</iso19103:value>
                  <iso19103:uom rdf:resource="#Grid_UnitOfAngle" />
            </iso19103:Angle>
```

```
        </iso19115:northBoundLatitude>
         <iso19115:eastBoundLongitude>
          <iso19103:Angle rdf:ID="Grid_Angle_1">
            <iso19103:value>-77.00</iso19103:value>
                    <iso19103:uom rdf:resource="#Grid_UnitOfAngle" />
          </iso19103:Angle>
        </iso19115:eastBoundLongitude>
 </iso19115:EX_GeographicBoundingBox>
<iso19103:UnitOfAngle rdf:ID="Grid_UnitOfAngle">
<iso19103:uomName>Degree Angle</iso19103:uomName>
<iso19103:conversionToISOStandardUnit>0.01745</iso19103:conversionToISOStandar
dUnit>
<iso19103:hasISOStandardUnit rdf:resource="radian" />
</iso19103:UnitOfAngle>

<iso19115:RS_Identifier rdf:ID="Grid_Horizontal_Datum_RS_Identifier">
   <iso19115:code>NAD 27</iso19115:code>
   <iso19115:name>North American Datum</iso19115:name>
</iso19115:RS_Identifier>
<iso19115:RS_Identifier rdf:ID="Grid_Vertical_Datum_RS_Identifier">
   <iso19115:code>NGVD 29</iso19115:code>
   <iso19115:name>National Geodetic Vertical Datum</iso19115:name>
</iso19115:RS_Identifier>
```

```
<iso19115:RS_Identifier rdf:ID="Grid_Ellipsoid_RS_Identifier">
   <iso19115:code>Clarke 1866</iso19115:code>
</iso19115:RS_Identifier>
<iso19115:RS_Identifier rdf:ID="Grid_Projection_RS_Identifier">
   <iso19115:code>Lambert Conformal Conic Projection</iso19115:code>
</iso19115:RS_Identifier>

<iso19115:MD_CRS rdf:ID="Grid_Horizontal_MD_CRS">
  <iso19115:projection          rdf:resource="#Grid_Projection_RS_Identifier" />
  <iso19115:ellipsoid  rdf:resource="#Grid_Ellipsoid_RS_Identifier" />
  <iso19115:datum      rdf:resource="#Grid_Horizontal_Datum_RS_Identifier" />
  <iso19115:projectionParameters      rdf:resource="#Grid_MD_ProjectionParameter" />
  <iso19115:ellipsoidParameters       rdf:resource="#Grid_MD_EllipsoidParameters" />
</iso19115:MD_CRS>
<iso19115:MD_ProjectionParameter rdf:ID="Grid_MD_ProjectionParameter">
   <iso19115:falseNorthing>0</iso19115:falseNorthing>
   <iso19115:falseEasting>400000</iso19115:falseEasting>
   <iso19115:latitudeOfProjectionOrigin>37.67</iso19115:latitudeOfProjectionOrigin>
   <iso19115:logitudeOfCentralMeridian>77</iso19115:logitudeOfCentralMeridian>
   <iso19115:standardParallel>39.45</iso19115:standardParallel>
   <iso19115:standardParallel>38.3</iso19115:standardParallel>
   <iso19115:obliqueLineAzimuthParameter
        rdf:resource="#Grid_MD_ObliqueLineAzimuth" />
```

```
</iso19115:MD_ProjectionParameter>
<iso19115:MD_ObliqueLineAzimuth rdf:ID="Grid_MD_ObliqueLineAzimuth">
   <iso19115:azimuthAngle>0</iso19115:azimuthAngle>

<iso19115:azimuthMeasurePointLongitude>400000</iso19115:azimuthMeasurePointLo
ngitude>
</iso19115:MD_ObliqueLineAzimuth>
<iso19115:MD_EllipsoidParameters rdf:ID="Grid_MD_EllipsoidParameters">
   <iso19115:semiMajorAxis>6360263.7936</iso19115:semiMajorAxis>
   <iso19115:axisUnits  rdf:resource="Grid_UnitOfLength"  />

<iso19115:denominatorOfFlatteningRatio>294.9786982</iso19115:denominatorOfFlatte
ningRatio>
</iso19115:MD_EllipsoidParameters>

<iso19103:UnitOfLength rdf:ID="Grid_UnitOfLength">
<iso19103:uomName>Meter</iso19103:uomName>
<iso19103:conversionToISOStandardUnit>1.0</iso19103:conversionToISOStandardUnit
>
<iso19103:hasISOStandardUnit rdf:resource="meter" />
</iso19103:UnitOfLength>

<iso19111:SC_CoordinateReferenceSystem rdf:ID="Grid_Vertical_SC_CRS">
```

```
    <iso19111:CRSID rdf:resource="#Grid_Vertical_Datum_RS_Identifier" />
    <iso19111:datum rdf:resource="#Grid_SC_VerticalDatum" />
</iso19111:SC_CoordinateReferenceSystem>
<iso19111:SC_VerticalDatum rdf:ID="Grid_SC_VerticalDatum">
    <iso19111:datumID          rdf:resource="#Grid_Vertical_Datum_RS_Identifier" />
    <iso19111:point>-30</iso19111:point>
</iso19111:SC_VerticalDatum>
</rdf:RDF>
```

APPENDIX 2: ONTOLOGY FOR A TWO DIMENSIONAL FINITE ELEMENT HYDRODYNAMIC CODE WRITTEN IN OWL

```
<?xml version="1.0" ?>
<rdf:RDF
   xmlns:ext="http://loki.cae.drexel.edu/~wbs/ontology/2004/09/iso-19115-ext#"
   xmlns="http://loki.cae.drexel.edu/~wbs/ontology/2004/04/wbs#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:iso19115="http://loki.cae.drexel.edu/~wbs/ontology/2004/09/iso-
19115#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:wbs="http://loki.cae.drexel.edu/~wbs/ontology/2004/09/wbs#"
   xml:base="http://loki.cae.drexel.edu/~wbs/ontology/2004/09/wbs"
>
  <owl:Ontology rdf:about="">
   <owl:versionInfo>version 1.0</owl:versionInfo>
   <rdfs:comment>
   title- An ontology to Describe Hydrodynamic Model Data for Web Based
Simulation
   creator- Akm Saiful Islam and Michael Piasecki
   publisher- Drexel University
```

```
            date - 08/16/04
            language - english
            </rdfs:comment>
            <owl:imports
rdf:resource="http://loki.cae.drexel.edu/~wbs/ontology/2004/09/iso-19115-ext"/>
          </owl:Ontology>
          <owl:Class rdf:ID="MetadataModel">
            <rdfs:label xml:lang="en">MetadataModel</rdfs:label>
            <rdfs:subClassOf
rdf:resource="http://loki.cae.drexel.edu/~wbs/ontology/2004/09/iso-
19115#MD_Metadata"/>
            <rdfs:comment>Metadata of the model</rdfs:comment>
          </owl:Class>
          <owl:Class rdf:ID="Wind">
            <rdfs:subClassOf>
              <owl:Class rdf:about="#ModelData"/>
            </rdfs:subClassOf>
            <rdfs:label xml:lang="en">Wind</rdfs:label>
            <rdfs:comment>Wind stress data</rdfs:comment>
          </owl:Class>
          <owl:Class rdf:ID="Data">
            <rdfs:subClassOf>
              <owl:Restriction>
```

```
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#timeValue"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="#attachTo"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:comment>Individual data object</rdfs:comment>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#dataFlag"/>
```

```
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label xml:lang="en">Data</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#dataValue"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Element">
  <rdfs:comment>Elements of the grid</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#numberOfNodes"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
```

```
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:label xml:lang="en">Element</rdfs:label>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#elementNode"/>
          </owl:onProperty>
          <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
          >1</owl:minCardinality>
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#elementID"/>
          </owl:onProperty>
          <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
          >1</owl:cardinality>
        </owl:Restriction>
      </rdfs:subClassOf>
    </owl:Class>
```

```
<owl:Class rdf:ID="MetadataModelData">
 <rdfs:subClassOf
rdf:resource="http://loki.cae.drexel.edu/~wbs/ontology/2004/09/iso-
19115#MD_Metadata"/>
</owl:Class>
<owl:Class rdf:ID="InitialWaterLevel">
 <rdfs:comment>Initial water surface elevation</rdfs:comment>
 <rdfs:subClassOf>
  <owl:Class rdf:about="#ModelData"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="GeoSpatialData">
 <rdfs:label xml:lang="en">GeoSpatialData</rdfs:label>
 <rdfs:comment>Aggregated class for geospatial data such as grid , maps, coast
lines etc.</rdfs:comment>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
   >1</owl:cardinality>
   <owl:onProperty>
    <owl:ObjectProperty rdf:about="#hasMetadataGeoSpatialData"/>
   </owl:onProperty>
  </owl:Restriction>
```

```
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Model">
    <rdfs:comment>Main Element for Model Ontology</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasGeoSpatialData"/>
        </owl:onProperty>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasMetadataModel"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
```

```
<rdfs:label xml:lang="en">Model</rdfs:label>
<rdfs:subClassOf>
 <owl:Restriction>
  <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >1</owl:minCardinality>
  <owl:onProperty>
   <owl:ObjectProperty rdf:about="#hasModelData"/>
  </owl:onProperty>
 </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Discharge">
 <rdfs:comment>Flow or discharge data</rdfs:comment>
 <rdfs:label xml:lang="en">Discharge</rdfs:label>
 <rdfs:subClassOf>
  <owl:Class rdf:about="#ModelData"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Map">
 <rdfs:comment>Maps and coast lines used as model
boundary.</rdfs:comment>
 <rdfs:subClassOf rdf:resource="#GeoSpatialData"/>
```

```
    <rdfs:label xml:lang="en">Map</rdfs:label>
   </owl:Class>
   <owl:Class rdf:ID="ModelData">
    <rdfs:label xml:lang="en">ModelData</rdfs:label>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >0</owl:minCardinality>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#hasData"/>
      </owl:onProperty>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#hasMetadataModelData"/>
      </owl:onProperty>
     </owl:Restriction>
    </rdfs:subClassOf>
```

<rdfs:comment>All types of model data which are connected with geospatial data</rdfs:comment>

</owl:Class>

<owl:Class rdf:ID="WaterLevel">

<rdfs:subClassOf rdf:resource="#ModelData"/>

<rdfs:comment>water surface elevation</rdfs:comment>

<rdfs:label xml:lang="en">WaterLevel</rdfs:label>

</owl:Class>

<owl:Class rdf:ID="Grid">

<rdfs:label xml:lang="en">Grid</rdfs:label>

<rdfs:subClassOf>

<owl:Restriction>

<owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >0</owl:minCardinality>

<owl:onProperty>

<owl:ObjectProperty rdf:about="#hasElement"/>

</owl:onProperty>

</owl:Restriction>

</rdfs:subClassOf>

<rdfs:subClassOf rdf:resource="#GeoSpatialData"/>

<rdfs:comment>Grid used in numerical model.</rdfs:comment>

<rdfs:subClassOf>

```
            <owl:Restriction>
              <owl:onProperty>
                <owl:ObjectProperty rdf:about="#hasNode"/>
              </owl:onProperty>
              <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
              >1</owl:minCardinality>
            </owl:Restriction>
          </rdfs:subClassOf>
        </owl:Class>
        <owl:Class rdf:ID="TidalElevation">
          <rdfs:label xml:lang="en">TidalElevation</rdfs:label>
          <rdfs:subClassOf rdf:resource="#ModelData"/>
          <rdfs:comment>Tidal water level data</rdfs:comment>
        </owl:Class>
        <owl:Class rdf:ID="Velocity">
          <rdfs:comment>Velocity data</rdfs:comment>
          <rdfs:label xml:lang="en">Velocity</rdfs:label>
          <rdfs:subClassOf rdf:resource="#ModelData"/>
        </owl:Class>
        <owl:Class rdf:ID="TributaryDischarge">
          <rdfs:comment>Tributary discharge data</rdfs:comment>
          <rdfs:subClassOf rdf:resource="#ModelData"/>
```

<rdfs:label xml:lang="en">TributaryDischarge</rdfs:label>

</owl:Class>

<owl:Class rdf:ID="Slope">

<rdfs:subClassOf rdf:resource="#ModelData"/>

<rdfs:label xml:lang="en">Slope</rdfs:label>

<rdfs:comment>The angle of the flow</rdfs:comment>

</owl:Class>

<owl:Class rdf:ID="Viscosity">

<rdfs:comment>Viscosity coefficient data</rdfs:comment>

<rdfs:subClassOf rdf:resource="#ModelData"/>

<rdfs:label xml:lang="en">Viscosity</rdfs:label>

</owl:Class>

<owl:Class rdf:ID="Material">

<rdfs:label xml:lang="en">Material</rdfs:label>

<rdfs:subClassOf rdf:resource="#ModelData"/>

<rdfs:comment>Material property depending on the type of roughness coefficient</rdfs:comment>

</owl:Class>

<owl:Class rdf:ID="MetadataGeoSpatialData">

<rdfs:subClassOf rdf:resource="http://loki.cae.drexel.edu/~wbs/ontology/2004/09/iso-19115#MD_Metadata"/>

</owl:Class>

<owl:Class rdf:ID="Dispersion">
<rdfs:label xml:lang="en">Dispersion</rdfs:label>
<rdfs:subClassOf rdf:resource="#ModelData"/>
<rdfs:comment>dispersion coefficent data</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Roughness">
<rdfs:label xml:lang="en">Roughness</rdfs:label>
<rdfs:subClassOf rdf:resource="#ModelData"/>
<rdfs:comment>Mannings roughness coefficient</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Radiation">
<rdfs:label xml:lang="en">Radiation</rdfs:label>
<rdfs:subClassOf rdf:resource="#ModelData"/>
<rdfs:comment>Rediational stress data</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Boundary">
<rdfs:comment>Type of conditional biundary. It could be either upstream inflow boundary , or no flow boundary or downstream tidal boundary</rdfs:comment>
<rdfs:subClassOf rdf:resource="#ModelData"/>
<rdfs:label xml:lang="en">Boundary</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Node">
<rdfs:subClassOf>

```
                <owl:Restriction>
                  <owl:onProperty>
                    <owl:DatatypeProperty rdf:about="#Z"/>
                  </owl:onProperty>
                  <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                  >1</owl:maxCardinality>
                </owl:Restriction>
              </rdfs:subClassOf>
              <rdfs:subClassOf>
                <owl:Restriction>
                  <owl:onProperty>
                    <owl:FunctionalProperty rdf:about="#nodeID"/>
                  </owl:onProperty>
                  <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                  >1</owl:cardinality>
                </owl:Restriction>
              </rdfs:subClassOf>
              <rdfs:subClassOf>
                <owl:Restriction>
                  <owl:onProperty>
                    <owl:DatatypeProperty rdf:about="#Z"/>
                  </owl:onProperty>
```

```
                <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >0</owl:minCardinality>
               </owl:Restriction>
              </rdfs:subClassOf>
              <rdfs:comment>Node is point in space.</rdfs:comment>
              <rdfs:subClassOf>
               <owl:Restriction>
                <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >1</owl:cardinality>
                <owl:onProperty>
                 <owl:FunctionalProperty rdf:about="#Y"/>
                </owl:onProperty>
               </owl:Restriction>
              </rdfs:subClassOf>
              <rdfs:label xml:lang="en">Node</rdfs:label>
              <rdfs:subClassOf>
               <owl:Restriction>
                <owl:onProperty>
                 <owl:FunctionalProperty rdf:about="#X"/>
                </owl:onProperty>
                <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >1</owl:cardinality>
```

```
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasElement">
  <rdfs:comment>information about element</rdfs:comment>
  <rdfs:label xml:lang="en">hasElement</rdfs:label>
  <rdfs:domain rdf:resource="#Grid"/>
  <rdfs:range rdf:resource="#Element"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasGeoSpatialData">
  <rdfs:label xml:lang="en">hasGeoSpatialData</rdfs:label>
  <rdfs:domain rdf:resource="#Model"/>
  <rdfs:comment>information about geospatial data</rdfs:comment>
  <rdfs:range rdf:resource="#GeoSpatialData"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasData">
  <rdfs:comment>Information about model data</rdfs:comment>
  <rdfs:label xml:lang="en">hasData</rdfs:label>
  <rdfs:domain rdf:resource="#ModelData"/>
  <rdfs:range rdf:resource="#Data"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="attachTo">
  <rdfs:range>
```

```
            <owl:Class>
              <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Node"/>
                <owl:Class rdf:about="#Element"/>
              </owl:unionOf>
            </owl:Class>
          </rdfs:range>
          <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
          <rdfs:label xml:lang="en">attachTo</rdfs:label>
          <rdfs:comment>provides information about the attachment type of model data.
Attachment could be with node or element or edge.</rdfs:comment>
          <rdfs:domain rdf:resource="#Data"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="elementNode">
          <rdfs:comment>nodes which make the element</rdfs:comment>
          <rdfs:domain rdf:resource="#Element"/>
          <rdfs:label xml:lang="en">elementNode</rdfs:label>
          <rdfs:range rdf:resource="#Node"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="hasMetadataModel">
          <rdfs:comment>information about metadata of the model</rdfs:comment>
```

<rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:range rdf:resource="#MetadataModel"/>
<rdfs:label xml:lang="en">hasMetadataModel</rdfs:label>
<rdfs:domain rdf:resource="#Model"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasMetadataModelData">
<rdfs:range rdf:resource="#MetadataModelData"/>
<rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:domain rdf:resource="#ModelData"/>
<rdfs:comment>Information about metadata of the model data</rdfs:comment>
<rdfs:label xml:lang="en">hasMetadataModelData</rdfs:label>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasModelData">
<rdfs:comment>information about model variables like wind , water level ,
discharge</rdfs:comment>
<rdfs:domain rdf:resource="#Model"/>
<rdfs:range rdf:resource="#ModelData"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasNode">
<rdfs:domain rdf:resource="#Grid"/>
<rdfs:label xml:lang="en">hasNode</rdfs:label>

```
        <rdfs:range rdf:resource="#Node"/>
        <rdfs:comment>information about the node</rdfs:comment>
      </owl:ObjectProperty>
      <owl:ObjectProperty rdf:ID="hasMetadataGeoSpatialData">
        <rdfs:range rdf:resource="#MetadataGeoSpatialData"/>
        <rdfs:domain rdf:resource="#GeoSpatialData"/>
      </owl:ObjectProperty>
      <owl:DatatypeProperty rdf:ID="Z">
        <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
        <rdfs:label xml:lang="en">Z</rdfs:label>
        <rdfs:domain rdf:resource="#Node"/>
        <rdfs:comment>altitude or z coordinate</rdfs:comment>
      </owl:DatatypeProperty>
      <owl:DatatypeProperty rdf:ID="dataFlag">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
        <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Data"/>
        <rdfs:comment>Indicator of data or null value</rdfs:comment>
      </owl:DatatypeProperty>
      <owl:DatatypeProperty rdf:ID="dataValue">
```

```
    <rdfs:label xml:lang="en">dataValue</rdfs:label>
    <rdfs:comment>numerical value for the data</rdfs:comment>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#Data"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:ID="timeValue">
    <rdfs:comment>time expressed as hour, minute and second</rdfs:comment>
    <rdfs:label xml:lang="en">timeValue</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Data"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="elementID">
    <rdfs:comment>unique identity for a element</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Element"/>
    <rdfs:label xml:lang="en">elementID</rdfs:label>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="X">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
```

<rdfs:comment>longitude or x coordinate</rdfs:comment>

<rdfs:domain rdf:resource="#Node"/>

<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>

<rdfs:label xml:lang="en">X</rdfs:label>

</owl:FunctionalProperty>

<owl:FunctionalProperty rdf:ID="nodeID">

<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>

<rdfs:domain rdf:resource="#Node"/>

<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>

<rdfs:comment>unique identity number of each node</rdfs:comment>

<rdfs:label xml:lang="en">modelID</rdfs:label>

</owl:FunctionalProperty>

<owl:FunctionalProperty rdf:ID="Y">

<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>

<rdfs:domain rdf:resource="#Node"/>

<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>

<rdfs:label xml:lang="en">Y</rdfs:label>

<rdfs:comment>latitude or y coordinates</rdfs:comment>

</owl:FunctionalProperty>

<owl:FunctionalProperty rdf:ID="numberOfNodes">

<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>

<rdfs:domain rdf:resource="#Element"/>

<rdfs:comment>number of nodes in each individual element</rdfs:comment>

```
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
</rdf:RDF>
```

VITA

Akm Saiful Islam was born, July 26, 1972, in Patuakhali, Bangladesh. He graduated from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh with a Bachelor's Degree in Civil Engineering in 1996. In December of 1999, he obtained his Master's Degree in Water Resources Engineering from the same university. His Master's thesis was on "Rainfall estimation over Bangladesh using satellite (GMS-5) data".

In January 2000, he enrolled in the Ph.D. program in Civil Engineering at Drexel University, Philadelphia, USA. Saiful's primary research interests include: hydrologic information, web based large scale simulation, ontology based metadata, and Java based hydrologic portal. While in Drexel, Saiful was working in a research project entitled, "Integrated Monitoring Modeling (Im2) System" sponsored by the National Ocean Partnership Program (NOPP) to developing a web based information portal for atmospheric and oceanographic data. Saiful also serves as president of the Bangladeshi Graduate Students' Association, Drexel University from 2002-2003.

Saiful was joined as a research faculty in the Institute of Water and Flood Management (IWFM) of BUET in 1997. He worked in numerous flood management and water resources research projects at IWFM. After graduating from Drexel University in September of 2004 with a Ph.D. in Civil Engineering, Saiful looks forward to continue his career at BUET as an Assistant Professor.