12-16-2013 12:00 AM

Disaster Data Management in Cloud Environments

Katarina Grolinger
The University of Western Ontario

Supervisor
Miriam A.M. Capretz
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

Disaster Data Management in Cloud Environments

(Thesis format: Monograph)

by

Katarina Grolinger

Graduate Program in Engineering Science
Department of Electrical and Computer Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

The School of Graduate and Postdoctoral Studies
Western University
London, Ontario, Canada

Abstract

Facilitating decision-making in a vital discipline such as disaster management requires information gathering, sharing, and integration on a global scale and across governments, industries, communities, and academia. A large quantity of immensely heterogeneous disaster-related data is available; however, current data management solutions offer few or no integration capabilities and limited potential for collaboration. Moreover, recent advances in cloud computing, Big Data, and NoSQL have opened the door for new solutions in disaster data management.

In this thesis, a Knowledge as a Service (KaaS) framework is proposed for disaster cloud data management (Disaster-CDM) with the objectives of 1) facilitating information gathering and sharing, 2) storing large amounts of disaster-related data from diverse sources, and 3) facilitating search and supporting interoperability and integration. Data are stored in a cloud environment taking advantage of NoSQL data stores. The proposed framework is generic, but this thesis focuses on the disaster management domain and data formats commonly present in that domain, i.e., file-style formats such as PDF, text, MS Office files, and images. The framework component responsible for addressing simulation models is SIMONTO. SIMONTO, as proposed in this work, transforms domain simulation models into an ontology-based representation with the goal of facilitating integration with other data sources, supporting simulation model querying, and enabling rule and constraint validation.

Two case studies presented in this thesis illustrate the use of Disaster-CDM on the data collected during the Disaster Response Network Enabled Platform (DR-NEP) project. The first case study demonstrates Disaster-CDM integration capabilities by full-text search and querying services. In contrast to direct full-text search, Disaster-CDM full-text search also includes simulation model files as well as text contained in image files. Moreover, Disaster-CDM provides querying capabilities and this case study demonstrates how file-style data can be queried by taking advantage of a NoSQL document data store.

The second case study focuses on simulation models and uses SIMONTO to transform proprietary simulation models into ontology-based models which are then stored in a graph database. This case study demonstrates Disaster-CDM benefits by showing how simulation

models can be queried and how model compliance with rules and constraints can be validated.

Keywords

Disaster Data Management, Big Data, NoSQL, Cloud Computing, Knowledge as a Service, Document Data Stores, Graph Databases, Data Model Design, Ontologies, Ontology-based Simulation Models

Acknowledgments

First and foremost, I would like to thank my supervisor, Dr. Miriam Capretz, for her invaluable guidance, encouragement, and support. Dr. Capretz always valued my opinion, trusted in my abilities, and made me feel that we were partners, which encouraged me to reach for higher goals. I truly appreciate the freedom and flexibility I was given in my research pursuit.

I would like to thank my husband Vladimir for supporting and encouraging me to pursue this degree and my kids, Monica and Dominic, for their patience and understanding. I hope that from my journey the kids learned about believing in yourself, being dedicated, and pursuing dreams. In addition, I am grateful to my parents for their quiet but steady encouragement.

Furthermore, I am grateful for the opportunity to work with a number of professors, including Dr. Said Tazi, Dr. Ernesto Exposito, Dr. Jose Marti, Dr. Krishan Srivastava, and Dr. Americo Cunha. I have learned greatly from each one of them.

Finally, I want to extend my thanks to my colleagues, especially Emna Mezghani, Wilson Higashino, Kevin Brown, and Vinson Wang, for sharing their ideas and providing feedback.

Table of Contents

List of Tables

List of Figures

Listings

List of Acronyms

ACID - Atomicity, Consistency, Isolation, Durability

ANNIE - A Nearly-New IE system

BASE - Basically Available, Soft state, Eventually consistent

BSON - Binary JSON

CI - Critical Infrastructures

COSMO - COmponent Simulation and Modelling Ontology

DeMO - Discrete-event Modelling Ontology

DES - Discrete Event Simulation

DESO - DES Ontology

Disaster-CDM – Disaster Cloud Data Management

DOM - Document Object Model

DR-NEP - Disaster Response Network Enabled Platform

GATE - General Architecture for Text Engineering

IaaS - Infrastructure as a Service

I2Sim - Infrastructures Interdependencies Simulation

IE – Information Extraction

JSIM - Java-based SIMulation

JSON - JavaScript Object Notation

KaaS – Knowledge as a Service

LOD - Linked Open Data

MVCC - With Multi-Version Concurrency Control

NIST - National Institute of Standards and Technology

NLP - Natural Language Processing

NoSQL – Not only SQL

OCR - Optical Character Recognition

OWL - Web Ontology Language

PaaS - Platform as a Service

PIModel - Process Interaction Model

PSCAD - Power System Computer Aided Design

PSL - Property Specification Language

RDB - Relational database

RDBMS - Relational Database Management System

RDF - Resource Description Framework

RDFS - RDF Schema

REST - Representational State Transfer

SaaS - Software as a Service

SQWRL - Semantic Query-Enhanced Web Rule Language

SWRL - Semantic Web Rule Language

UFO - Unified Foundational Ontology

W3C - World Wide Web Consortium

XPIM - Extensible Process Interaction Markup

Chapter 1

1 Introduction

Each year, a number of natural disasters strike across the globe, killing hundreds and causing billions of dollars in property and infrastructure damage. Extreme weather events have been predicted by climate scientists and have been attributed to global warming. As the number of such events increases, minimizing the impact of disasters becomes imperative in today's society.

The role of information and communication technology in disaster management has been evolving. Large quantities of disaster-related data are being generated. Behaviour of critical infrastructures is being explored through simulation, response plans are being created by government agencies and individual organizations, sensory systems are providing potentially relevant information, and social media (Twitter, Facebook) have been flooded with disaster information [1]. Traditional storage and data processing system are facing challenges in meeting the performance, scalability, and availability needs of Big Data. In the context of disaster data management, Big Data refers to the massive collection of data sets generated by various participants and composed of diverse data structures, including structured, semi-structured, and unstructured data [1]. Current disaster data storage systems are disparate, providing few or no integration capabilities and limited potential for collaboration. To meet the needs of Big Data and make the most of available information, a reliable and scalable storage system provided by cloud infrastructure and supported by information sharing, reuse, integration, and analysis is needed.

1.1 Motivation

A vital element of successful disaster management is collaboration among a number of teams, including firefighters, first aid, police, critical infrastructure personnel, and many others. Each team or recovery unit is responsible for performing a well-defined task, but their collaboration is essential for decision-making and execution of well-organized and successful recovery operations [2]. The proliferation of social networking has introduced

citizens as collaborators in disaster decision-making since they can provide relevant information [3]. Such diverse disaster participants generate large quantities of heterogeneous disaster-related data, making information gathering, storage, and integration especially challenging.

The activities of various disaster participants can be observed through four disaster management phases, as illustrated in Figure 1.1: mitigation, preparedness, response, and recovery [4]. Mitigation includes all activities undertaken to reduce disaster effects by avoiding or decreasing the impact of a disaster. The preparedness phase is concerned with preparing for disaster occurrence and includes activities such as planning, establishing procedures and protocols, training, and exercises. In this phase, collaboration is an essential element to correlate activities and generate effective plans and procedures. Examples of data generated during the mitigation and preparedness phases include response plans, emergency procedures, records of training exercises, and data about available response resources. The transition from the preparedness to the response phase is triggered by a disaster occurrence. The response is focused on addressing the direct, short-term effects of a disaster and includes immediate actions to save lives, protect property, and fulfill basic human needs. Collaboration among participants is crucial for a successful disaster response. The transition to the recovery phase starts when the direct disaster threat subsides and includes activities focused on bringing society into a normal state. Examples of the data generated during the response and recovery stages include incident reports, lessons learned, and improvements to disaster plans. The approach proposed in this study carries out both data collection and delivery through all four phases; however, the focus is on data collection during the mitigation and preparedness stages, while during the response and recovery phases, the focus is on data delivery, as illustrated in Figure 1.1. In other words, the main intent is not real-time collection of information during disaster response, but better use of the information collected in different phases. The ultimate goal is to create a knowledge system which will provide effective support for disaster management as well as support for other disaster-related activities.

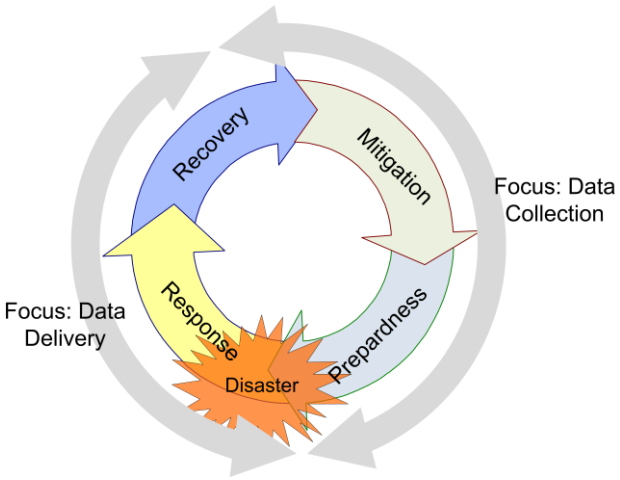

Figure 1.1: Disaster management phases

Recent advances in cloud computing, Big Data, and NoSQL have been changing how data are captured, stored, and analyzed. NoSQL solutions have been especially popular in Web applications [5], including Facebook, Twitter, and Google. However, the use of cloud technologies and NoSQL solutions in disaster management has been sparse.

A solution which stores disaster-related data in a cloud environment can provide the following benefits to disaster management [6]:

- *High availability*. Within the cloud environment, data are automatically replicated, often across large geographic distances. If a region is affected by a disaster and a local data centre fails, the system remains available because it can switch to another data centre.
- *Scalability and elasticity*. The amount of disaster-related data is massive, and a cloud solution can adapt storage resources based on real-time needs and priorities. Data can be automatically redistributed to take advantage of heterogeneous servers.
- *There is no need for a large initial investment*. The system can start small and be expanded by adding heterogeneous nodes as needed.

Moreover, NoSQL data stores have a number of characteristics that can benefit disaster data management, including [7]:

- *Flexible data structure*. Disaster data are extremely diverse, and therefore it would be almost impossible to store them in a predetermined data structure.
- *Horizontal scalability*. NoSQL data stores were designed for a cloud environment, and therefore they scale easily over a large number of commodity servers.
- *Performance*. For simple read/write operations, NoSQL data stores can provide better performance than relational databases.

Another crucial element of disaster management is simulation because it provides a means of studying the behaviour of critical infrastructures, as well as a way of exploring disaster response "what-if" scenarios. Therefore, simulation-related information must be an integral part of any disaster knowledge system.

Although the act of simulation is not domain-specific, simulation packages are usually application-oriented (designed for simulation experiments in a specific domain) and use different modelling approaches, diverse technologies and a wide variety of domain-specific vocabularies. This heterogeneity in the simulation domain, representation, and semantics presents an obstacle to simulation model querying and rule and constraint validation and hinders the integration of simulation data with other information sources. To be able to provide comprehensive knowledge services, a disaster knowledge system needs to take advantage of simulation-related information and integrate it with other sources. Moreover, to enable better exploration of simulation models, the solution needs to provide querying within simulation models and rule and constraint validation capabilities.

1.2 Goals and Scope

The ultimate goal of this research is to design a data management framework which will provide effective support for disaster management as well as support for other disaster-related activities. The main focus is on better use of existing information and not on real-time data collection and delivery during a disaster; however, the proposed approach

allows data collection and delivery through all four disaster phases. This research will facilitate disaster preparedness, response, and recovery efforts by providing a flexible and expandable storage solution for diverse disaster data. Supporting global information sharing, reuse, and integration, the proposed solution will provide improved and informed decision-making and will therefore reduce the impact of disasters on human lives and property.

Consequently, this research proposes a Knowledge as a Service (KaaS) framework for disaster cloud data management (Disaster-CDM). KaaS [8] aims to generate, from data stored in a cloud environment, knowledge such as advice or responses to meet organizational needs. Therefore, Disaster-CDM has the objectives of:

1. Facilitating information gathering and sharing through collaboration. Knowledge acquisition is responsible for acquiring knowledge from diverse sources and from various collaboration partners. Knowledge delivery is responsible for integrating information and delivering it to consumers as a service.
2. Storing large amounts of disaster-related data from diverse sources. The storage of massive quantities of immensely diverse disaster-related data is achieved by using a combination of various data stores in a cloud environment.
3. Facilitating search and supporting interoperability and integration. Knowledge delivery services are the primarily components responsible for this task. Data stored in diverse data stores are provided to consumers as a service.

The proposed framework is not disaster-specific and could potentially be applied for data management in other domains. However, Disaster-CDM was motivated by disaster scenarios and it was designed for the management of disaster-related data; consequently, this work applies it on disaster-related data.

A part of the proposed framework responsible for addressing simulation models is SIMONTO, an ontology-based representation of simulation models. SIMONTO, as proposed in this work, represents domain simulation models as interconnected instances of simulator-specific ontologies. Specifically, SIMONTO uses existing models in the simulation engines' proprietary file formats as the foundation for the creation of its

ontology-based representation. Such ontology-based simulation models are stored in the NoSQL data store with the goal of:

- Facilitating integration with other information sources,
- Providing querying capabilities,
- Enabling rule and constraint validation.

The proposed Disaster-CDM provides a flexible and customizable disaster data management solution which can be expanded and altered according to the needs of the organizations using it: Disaster-CDM accommodates new data sources by adding new data processing services and by taking advantage of various NoSQL data stores. The solution is based on cloud computing, NoSQL data stores, and the KaaS approach; however, it takes advantage of a large number of other technologies, such as Web services, full-text search, optical character recognition (OCR), ontologies, and various querying approaches.

The contributions of this thesis can be summarized as follows:

Disaster-CDM framework, a Knowledge as a Service (KaaS) framework for disaster cloud data management, is proposed. It supports disaster management and other disaster-related activities by providing disaster-related knowledge as a service. Disaster-CDM achieves the following objectives:

- Information gathering and sharing is facilitated by means of knowledge acquisition and knowledge delivery services.
- Storing large amounts of disaster-related data from diverse sources is achieved by taking advantage of cloud computing and NoSQL data stores.
- Search, interoperability and integration are supported primarily by means of knowledge delivery services.

Moreover, the research presented in this thesis defines a process for introducing a new data source into the proposed Disaster-CDM framework. The process consists of:

- adding new data processing services for dealing with the new data source;

- defining data processing rules for new data sources;
- determining suitable data storage, including choosing the type of data store and designing a storage data model.

SIMONTO is the part of the proposed Disaster-CDM framework responsible for processing simulation models. Existing simulation models expressed in simulator-specific model files are transformed to their corresponding ontology-based representations which are better suited for integration with other data source and for providing simulation model querying capabilities, and rule and constraint validation. The ontology-based simulation models are stored according to their intended use:

- For integration with other sources, simulation models are stored in a document database alongside other data.
- For querying within simulation models, and for enabling rule and constraint validation, ontology-based simulation models are stored in a graph database.

1.3 Thesis Organization

This thesis is organized into chapters as follows:

- Chapter 2 presents the main concepts and technologies relevant to this study: Big Data, cloud computing, and NoSQL data stores. The term "Big Data" in the context of disaster data management is defined. Because the disaster data management solution proposed in this work is cloud-based, the main characteristics, goals, and challenges of cloud computing are discussed. Next, since the Disaster-CDM storage model incorporates NoSQL solutions, NoSQL data stores are introduced and their characteristics described. Furthermore, the four NoSQL data models are discussed with an emphasis on characteristics relevant in the Disaster-CDM context.
- Chapter 3 surveys related work. First, work in disaster data management is examined, and the difference in focus between the reviewed work and the research reported in this thesis is highlighted. Because this research proposes a KaaS-based solution for disaster data management, studies that apply the KaaS

approach are examined. Next, work related to simulation model querying and rule and constraint validation is reviewed, and finally, the use of ontologies in simulation modelling is presented.

- Chapter 4 proposes Disaster-CDM, a Knowledge as a Service (KaaS) framework for disaster cloud data management. The two main parts of the Disaster-CDM framework are discussed: knowledge acquisition and knowledge delivery. Knowledge acquisition is responsible for acquiring knowledge from diverse sources, processing it to add structure to unstructured or semi-structured data, and storing it in data stores. Knowledge delivery is responsible for integrating information from different data stores and delivering knowledge to consumers as a service.
- Chapter 5 focuses on Disaster-CDM for file-style data, which are common in the disaster management domain. The generic process of adding a new data source to the proposed framework is introduced and then applied for file-style data sources. Details of applying each of the three steps to file-style data sources are discussed: establishing required data processing services, defining data processing rules, and data storage in the cloud environment.
- Chapter 6 proposes SIMONTO, an ontology-based representation of simulation models, which represents proprietary simulation models as interconnected instances of simulator-specific ontologies. In the context of Disaster-CDM, SIMONTO is responsible for simulation model processing. Integration with other file-style data is achieved by storing simulation models in a document data store along with other data sources. On the other hand, simulation model querying and rule and constraint validation are achieved by storing the ontology-based simulation models in a graph database.
- Chapter 7 presents an evaluation of the proposed Disaster-CDM framework on data collected during the CANARIE sponsored Disaster Response Network Enabled Platform (DR-NEP) project. The presented case study applies the Disaster-CDM framework on file-style data sources including simulation models. First, the Disaster-CDM implementation is described, including its two main knowledge acquisition components: data processing services and data storage.

Disaster-related knowledge is acquired from the DR-NEP data set and stored in a document data store. Finally, the benefits of Disaster-CDM are demonstrated on two knowledge delivery services: full-text search and querying.

- Chapter 8, like Chapter 7, presents an evaluation of the proposed Disaster-CDM framework; however, in contrast to Chapter 7 which addresses file-style data sources, this chapter is concerned with simulation models. The SIMONTO implementation and the ontology-based models created by SIMONTO are discussed first. In the presented case study knowledge acquisition service, specifically SIMONTO, transforms simulation models to their corresponding ontology-based representations and stores them in a graph database. Finally, the benefits of Disaster-CDM are demonstrated on two simulation-specific knowledge delivery services: simulation model querying and rule and constraint validation.
- Chapter 9 concludes this study by discussing the main contributions of this research as well as directions for future work. The two main contributions include the Disaster-CDM framework and SIMONTO, the part of the framework responsible for processing simulation models. Although this study has focused on disaster data management, the proposed Disaster-CDM framework is generic and could be applied in other domains. Consequently, future work will explore the potential of the proposed framework in other domains such as geological data management.

Chapter 2

2 Background

This chapter introduces the main concepts and technologies relevant to this work: Section 2.1 introduces Big Data, Section 2.2 portrays cloud computing, and Section 2.3 presents the background on NoSQL data stores.

2.1 Big Data

In recent years, advances in Web technology and the proliferation of sensors and mobile devices connected to the Internet have resulted in the generation of massive data sets that must be processed and stored. For example, Facebook today has more than one billion users, with over 618 million active users on a daily basis generating more than 500 terabytes of new data each day [9].

Traditional relational database management systems (RDBMS) as well as data processing approaches were designed in an era when available hardware, as well as storage and processing requirements, were very different than they are today [10]. Therefore, traditional approaches are facing many challenges in meeting the requirements of Big Data, including storage, processing, management, search, transfer among devices or storage locations, analysis, and visualisation.

The term “Big Data” refers to large and complex data sets made up of a variety of structured and unstructured data which are too big to be managed by traditional techniques. According to Beyer and Laney [11], Big Data is characterized by the 3Vs: volume, velocity and variety. Volume refers to the quantity of data, variety refers to the diversity of data types, and velocity refers both to how fast data are generated and how fast they must be processed. Occasionally, a fourth V is added [12]: veracity is the ability to trust the data to be accurate and to use them to make crucial decisions.

Big Data in the context of disaster data management, and even more specifically in the Disaster-CDM framework, refers to large collections of disaster-related data sets owned by various disaster participants. These data sets must be integrated to provide efficient

support for disaster management. In addition to volume, the variety of disaster-related data is a major challenge that Disaster-CDM must overcome to be able to provide integrated disaster knowledge as a service. Moreover, the veracity of disaster data is also significant as the decision-makers must be able to trust the data to use it in decision-making.

Enterprises are aware that Big Data has the potential to impact core business processes, provide competitive advantage, and increase revenues [12,13]. Therefore, organizations are exploring ways to make better use of Big Data by analyzing them to find meaningful insights which would lead to better business decisions and add value to their business. In the disaster management domain, better use of available information has the potential to improve decision-making, thus reducing the impact of disasters on human lives and property.

A trend in the Big Data world of special interest to this research is collaboration. This refers to data sharing as well as treating data as a commodity which considers data as a product and even offers it as a service [13]. In the disaster management domain, collaboration among large numbers of participants is essential for successful response and recovery operations. Specifically, in the proposed Disaster-CDM approach, data sharing is achieved through knowledge acquisition from a variety of data sources owned by different collaborators. The integrated data are provided to consumers as a knowledge service.

2.2 Cloud Computing

Various cloud computing definitions have been proposed [14,15]; however, the definition proposed by the National Institute of Standards and Technology (NIST) has been gaining acceptance [5,15]. According to NIST, cloud computing is [16]

“a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”.

It is important to point out the synergy between Big Data and cloud computing. Big Data, due to its size, volume and velocity, imposes continuously increasing computing demands on traditional computing techniques. Cloud computing promises to meet these demands by using a large number of networked resources. Therefore, cloud computing is one of the key enabling techniques for handling Big Data; hence, this work uses it for management of disaster-related Big Data.

In cloud computing, service providers offer computer-based services, and service consumers use these services over the network. A large number of IT companies, including Amazon, Google, Microsoft, Rackspace, and IBM are now providing cloud computing services. According to the NIST definition, the main characteristics of cloud computing include [5,14,15]:

- On-demand self-service. Services are consumed as needed, without the need for human interaction.
- Broad, ubiquitous network access. Services are provided over the network through standard mechanisms.
- Resource pooling. Computing resources are pooled to serve multiple consumers in a multi-tenant environment.
- Rapid elasticity. Dynamic resource provisioning is achieved by obtaining and releasing resources on the fly.
- Utility-based pricing (a pay-per-use pricing model). The consumer pays only for resources used.

Consequently, the goal of cloud computing systems is to provide the following benefits [5]:

- *Availability*. The system needs to remain operational and accessible in case of server, network, or even data centre failure.
- *Scalability.* This refers to the ability to handle growing demands.
- *Elasticity*. Changing requirements need to be accommodated by scaling up or down.

- *Performance*: In a pay-per-use pricing model, performance is directly correlated with cost.
- *Multi-tenancy*. Many tenants (services, applications) reside on the same hardware and software infrastructure.
- *Fault tolerance*. This refers to the ability of a system to continue operating in the presence of failures.
- *Load balancing*. Loads are automatically moved among servers to achieve effective resource utilization.
- *Ability to run on heterogeneous commodity servers*. In infrastructures involving a large number of nodes, heterogeneity is almost unavoidable.

In this research, all mentioned attributes contribute to the choice of a cloud environment for management of disaster data; however, it is important to highlight scalability and availability attributes. Scalability makes it possible to start the system small and expand as needs grow by adding heterogeneous nodes. High availability ensures system operation in the presence of failures, which in the disaster management domain is particularly important as it can be expected that disasters will cause a variety of failures.

From the delivery perspective, the three common cloud computing models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The IaaS model provides resources such as servers (physical or virtual), networks, storage, and operating systems. The PaaS model offers a higher-level environment and delivers a computing platform including data storage, programming languages, and Web application servers. Finally, the SaaS model provides on-demand software by offering access to software applications through the Internet.

Specialized variations of these three models have emerged, including Storage as a Service, Database as a Service, Security as a Service, Integration as a Service, and Testing as a Service [15]. The Disaster-CDM approach proposed in this work applies the Knowledge as a Service (KaaS) model, in which requests presented by consumers are

answered by knowledge providers through knowledge services [17]. In other words, the proposed Disaster-CDM provides disaster-related knowledge as a service.

Even though cloud computing is gaining popularity in industry and academia, further adoption is facing a number of challenges. Because the approach proposed in this thesis draws on cloud computing, it is exposed to the same challenges:

- Security and privacy. In a public cloud, data are stored and processed on third-party premises and in a shared multi-tenant environment; therefore, security and privacy vulnerabilities are increased. Providing an adequate solution is difficult as it needs to include both the service provider and the service consumer.
- Customer lock-in. Due to lack of standardization within the cloud computing industry, it is challenging to move from one cloud provider to another. Customer lock-in makes cloud consumers vulnerable to price increases.
- Data transfer challenges. The physical locations of provider and consumer may result in significant network traffic which must be considered when evaluating performance and cost.
- Legal issues. Public cloud resources may reside in a geographical region with different security and privacy regulations than those in the cloud consumer region. For example, European companies storing data in the United States expose their data to easier access by government agencies due to the U.S. Patriot Act [15].
- Application parallelization. In the cloud computing environment, additional resources are typically acquired by allocating additional servers; however, only applications with parallelizable workload can take advantages of such resources.

Even though the cloud computing challenges just described are generic and are outside the scope of this work, they have a major impact on possible adoption of this work in practice. Moreover, these challenges need to be taken into consideration when implementing the proposed approach in practice.

Since this research focuses on data storage in the cloud, new storage solutions, namely NoSQL data stores, are introduced in Section 2.3.

2.3 NoSQL Data Stores

Relational databases (RDBs) are traditional data storage systems designed for structured data. They have been used for decades due to their reliability, consistency, ACID (Atomicity, Consistency, Isolation, Durability) transactions and query capabilities through SQL. However, RDBs exhibit horizontal scalability challenges, Big Data inefficiencies, and limited availability [18]. In an attempt to address the challenges encountered by RDBs in handling Big Data and in satisfying cloud requirements, new storage solutions, namely NoSQL data stores [6], have emerged. Because this work aims to provide a storage solution for disaster-related Big Data, the proposed solution takes advantage of NoSQL data stores.

Today, the term "NoSQL" refers to "Not only SQL", which emphasizes that SQL-style querying is not the crucial objective of these data stores. Therefore, the term encompasses a large number of immensely diverse data stores that are not based on the relational model, including some solutions designed for highly specific applications such as graph storage. Even though there is no agreement on what exactly constitutes a NoSQL solution, the following set of characteristics is often attributed to them [7,15,19]:

- Simple and flexible non-relational data models. NoSQL data stores offer flexible schemas or are sometimes completely schema-free and are designed to handle a wide variety of data structures [7,20].
- Ability to scale horizontally over many commodity servers. Some data stores provide data storage scaling, while others are more concerned with read and/or write scaling.
- High availability. Many NoSQL data stores are meant to be used in highly distributed scenarios and consider partition tolerance as unavoidable. Therefore, to provide high availability, these solutions choose to compromise consistency in favour of availability, resulting in AP (Available / Partition-tolerant) data stores, whereas most RDBMSs are CA (Consistent / Available).
- Typically, NoSQL data stores do not support ACID transactions as provided by RDBMS. NoSQL data stores are sometimes referred to as BASE systems

(Basically Available, Soft state, Eventually consistent) [21]. In this acronym, Basically Available means that the data store is available whenever accessed, even if certain parts are unavailable; Soft state highlights the fact that it can tolerate inconsistency for a certain time period; and Eventually consistent emphasizes that after a certain time period, the data store will arrive at a consistent state.

- Lesser emphasis on normalization. Denormalized schema can provide simpler data access, reduce use of resource-intensive operations such as joins, and can more easily scale horizontally. However, this approach will result in larger storage size than for data stored in normalized schema [15].

Distributed and cloud computing are the key enabling technologies for NoSQL data stores. At the time when relational databases emerged, available storage space was restricted and thus normalization was highly desired and redundancy unwanted. Today, distributed and cloud computing provide massive storage space, but the immense quantity of operations imposes strict performance requirements. Therefore, focus has shifted from minimizing redundancy and storage space to improving performance [15]. Consequently, NoSQL schemas are often denormalized resulting in large storage size, but providing a number of advantages including:

- Better horizontal scalability as denormalized schema can be partitioned easier,
- Because data can be redundant, it can be repeated in order to simplify data access,
- Resource-intensive operation such as joins can be avoided,
- Schema can closer resemble application object model and therefore reduce impedance mismatch.

The main characteristics responsible for making NoSQL stores a suitable storage option for the disaster data management solution proposed in this work include their flexible data model, horizontal scalability, and high availability. A flexible data model enables storage of diverse disaster-related data, horizontal scalability enables a NoSQL data store to accommodate growing storage needs by adding commodity servers, and high availability ensures continuous operation in case of disasters.

NoSQL data stores are typically further classified according to their data model. As there is no agreement on what exactly constitutes a NoSQL data store, various categorizations have been proposed [19,20]. This study adopts the categorization into four categories: key-value data stores, column-family stores, document stores, and graph databases [7,19,22]. The following discussion introduces the four NoSQL data store categories and highlights the main characteristics relevant for their use in the Disaster-CDM framework.

Key-Value Data Stores have the simplest data model: they provide a simple mapping from each key to its corresponding value. They are primarily used for simple operations in which all access to the store is through a primary key. Client applications can set the value for a key, get the value corresponding to a specified key, or delete a key. The value can be just about anything, and the client application is responsible for interpreting what is stored. Therefore, when using a key-value data store, relations between data are handled at the application level. Although such a simple data model is somewhat restrictive, accessing data only through the primary key provides for good performance and easy scalability. Examples of key-value data stores include Redis, Riak, and Berkeley [19].

In spite of their flexibility, scalability, and performance characteristics, key-value stores have major drawbacks with respect to Disaster-CDM. Relations between data are handled by the application, and data are accessed only through the primary key. Since the relations among data are not expressed in the data store's data model, integration possibilities are limited. Moreover, accessing data only through the primary key greatly restricts querying capabilities. In the context of Disaster-CDM, limited querying capabilities and integration possibilities present a major drawback.

Document Data Stores are designed around the concept of a document and focus on optimizing storage and access for semi-structured documents as opposed to rows or records. They are derivatives of the key-value store data model with documents stored in the value part of the key-value pair. The documents, typically in JSON (JavaScript Object Notation) or BSON (Binary JSON) representation, are hierarchical trees which encapsulate and encode data. The documents within the data store can have different

structures, which provide storage flexibility. At the same time, the document structure enables querying capabilities as fields within documents can be used as query criteria. Example data stores from this category include CouchDB, MongoDB, and Couchbase Server [19].

In the context of Disaster-CDM, document data stores provide two advantages: querying capabilities and flexible storage. Querying capabilities are made possible by the structure of the documents within the data store, while storage flexibility is achieved by allowing documents within the store to have different structures. However, querying capabilities and storage flexibility are competing attributes: a certain structural consistency among documents is needed to support querying, while excessive structural consistency decreases storage flexibility.

Column-Family Data Stores, like key-value stores, map keys to their corresponding values; however, each value consists of a name-value pair. Key-value pairs can be perceived as rows in a relational database, while name-value pairs relate to column names and their corresponding values. Thus, column-family stores are on the surface similar to relational databases; however, in the relational database, columns are predefined, and each row contains the same fixed set of columns, whereas in the column-family data store, the columns that form a row are determined by the client application, and each row can have a different set of columns. Column-family data stores provide query capabilities. Cassandra, HBase, and Amazon SimpleDB belong to this category [19].

In the context of Disaster-CDM, column-family data stores provide the same advantages as document data stores: querying capabilities and flexible storage. Querying capabilities are supported by name-value pairs within rows, while storage flexibility is achieved by allowing each row to have a different set of columns. Similarly to document databases, a certain level of consistency among rows is needed to support querying capabilities.

Graph Databases originated from graph theory and use graph-like structures with nodes, edges, and properties to store data. This data model is very different from the key-value, document, and column-family data models and is designed for efficient management of heavily linked data. Applications based on data with many relationships are well suited

for graph databases because the cost of intensive operations like recursive joins can be replaced by efficient graph traversals [7]. Neo4J and Allegro Graph are example stores from this category [19].

In the context of Disaster-CDM, graph databases are suitable for storage of heavily linked data and for data with a graph-like data model. For example, ontology-based simulation models are based on simulation model graphs and therefore are suitable for storage in a graph database. In the Disaster-CDM framework, graph databases have the advantage of advanced querying capabilities: graph database implementations often provide powerful and diverse querying capabilities. For example, a Neo4j graph database can be queried using Cypher, a property graph query language developed by Neo4j; using Gremlin, a graph traversal language; or even using the RDF query language, SPARQL.

In addition to differences in their data models, data store implementations differ greatly in other aspects, such as scalability, fault tolerance, consistency, and concurrency control. These characteristics, in addition to the data model, are influential factors in determining the most suitable data store for the task at hand. Disaster-CDM offers a choice of storage solutions according to the characteristics of the data to be stored. Specifically, the data store category is chosen according to the data to be stored, and a specific data store implementation is then selected by matching the desired storage attributes with the characteristics of various data store implementations.

Because one of the main characteristics of NoSQL data stores is their ability to scale horizontally and effectively by adding more servers to the resource pool, scaling aspects are discussed further here. With regard to what is being scaled, three scaling dimensions can be distinguished: scaling read requests, scaling write requests, and scaling data storage. The partitioning, replication, consistency, and concurrency control strategies used by NoSQL data stores have significant impact on their scalability. For example, partitioning determines the distribution of data among multiple servers and is therefore a means of achieving all three scaling dimensions.

Another important factor in scaling read and write requests is replication: storing the same data on multiple servers so that read and write operations can be distributed over

them. Replication also has an important role in providing fault tolerance because data availability can withstand the failure of one or more servers. Furthermore, the choice of replication model is also strongly related to the consistency level provided by the data store. For example, the master-slave asynchronous replication model itself cannot provide consistent read requests from slaves. In the context of Disaster-CDM, the replication model is relevant when choosing the best data store implementation for the task at hand.

2.4 Summary

This chapter has presented the main concepts and technologies relevant to this study: Big Data, cloud computing, and NoSQL data stores. The term "Big Data" has been defined, and its meaning in the context of disaster data management, and specifically Disaster-CDM, has been emphasized. Because the disaster data management solution proposed in this work is cloud-based, the main characteristics, goals, and challenges of cloud computing have been discussed. The choice of the cloud environment for the storage of disaster-related data has been primarily motivated by its scalability and availability attributes. Next, because the Disaster-CDM storage model incorporates NoSQL solutions, NoSQL data stores were introduced and their characteristics described. The motivating factors for choosing NoSQL data stores in the proposed approach included data model flexibility, horizontal scalability, and high availability. Furthermore, the four NoSQL data models were described with an emphasis on the characteristics relevant in the Disaster-CDM context.

Chapter 3

3 Related Work

This chapter surveys three categories of related work: disaster data management, Knowledge as a Service (KaaS) and related simulation work.

3.1 Disaster Data Management

Research in disaster management involves many fields, including health science, environmental science, computer science, and a number of engineering disciplines. Crisis informatics [23,24], the area of research concerned with the role of information and technology in disaster management, has been attracting increased research attention recently.

Hristidis *et al.* [1] surveyed data management and analysis in the disaster domain. The main focus of their survey was on data analysis techniques without the storage aspect. In contrast, in Disaster-CDM, storage and analysis are considered as integral parts. Hristidis *et al.* identified the following data analysis technologies as relevant in disaster data management: information extraction, information retrieval, information filtering, data mining, and decision support. Similarly, Disaster-CDM uses a number of technologies from information extraction and retrieval. The survey reveals that the majority of research has focused on a very narrow area of disaster management, for example, a specific disaster event such as an earthquake or a flood, or specific disaster-related activities such as communication among actors, estimating disaster damage, and use of mobile devices. Hristidis *et al.* recognized the need for flexible and customizable disaster management solutions that could be applied in different disaster situations. Disaster-CDM aims to provide such a solution using cloud computing and NoSQL data stores.

Othman and Beydoun [25] pointed out the importance of providing sharable disaster knowledge in facilitating better disaster decision-making. They proposed a Disaster Management Metamodel with the objective of improving knowledge sharing and supporting the combination and matching of different disaster management activities.

This metamodel was instantiated twice: for an earthquake, and for a nuclear meltdown disaster situation. Although they highlighted the large amount of information generated in the disaster domain, their study does not consider disaster data storage. Disaster-CDM provides a scalable and flexible data storage solution in a cloud environment, accommodates both structured and unstructured data, and supports data sharing.

Silva *et al.* [26] aimed to integrate diverse, distributed information sources by bringing them into a standardized and exchangeable common data format. Their approach focused on data available on public Web sites. Data were first extracted from different source Web sites and stored in a relational database. Next, the data were transformed into Linked Open Data (LOD) and published. In contrast to their work which addressed data available on public Web sites, the proposed Disaster-CDM can accommodate various information sources. In addition, Disaster-CDM is designed for high availability and large amounts of data.

Palen *et al.* [23] presented a vision of technology-supported public participation during disaster events. They focused on the role of the public in disasters and how information and communication technology can transform that role. Similarly to Hristidis *et al.* [1], they recognized information integration as a core concern in crisis informatics.

Anderson and Schram [27], like Palen *et al.* [23], studied the role of public and social media in disaster events. They proposed a crisis informatics data analytic infrastructure for the collection, analysis, and storage of information from Twitter. The main objective of their work was the support of other crisis information research by extracting disaster-related tweets from Twitter and storing them in a database. In their initial study [27], data were stored in a relational database, specifically MySQL. Later, after encountering scalability challenges, they transitioned to a hybrid architecture that incorporates relational database and NoSQL data store [24]. Similarly, Disaster-CDM also uses a combination of relational database and NoSQL data stores. However, a combination of several NoSQL data stores has been used to address the storage requirements of diverse data. Specifically, Disaster-CDM allows the choice of storage solutions to suit a variety of data structures and access patterns.

Chou *et al.* [28] proposed an ontology for developing Web sites for natural disaster management. Web elements contained in the ontology were identified using a ground theory approach with an inventory of disaster management Web sites. To represent the Web page elements, they adopted a combination of XML, XML schemas, and document object model (DOM). The proposed ontology provides support for designing dynamic emergency response management Web sites. Like Chou *et al.* [28], Disaster-CDM also uses ontologies, but their purpose is data integration in the knowledge delivery stage. Moreover, while Chou *et al.* addressed disaster Web sites, Disaster-CDM is concerned with a variety of diverse data sources.

3.2 Knowledge as a Service (KaaS)

Disaster-CDM incorporates the KaaS approach to make disaster-related knowledge available as services. Within KaaS, a knowledge provider answers requests presented by knowledge consumers through knowledge services [17]. In Disaster-CDM, the main goal is to acquire knowledge from the diverse data sources and expose it as service to knowledge consumers. Generally, KaaS publishes knowledge models that represent a collection of learned lessons, best practices, and case studies as services that help consumers get knowledge from a distributed computing environment.

The KaaS approach has been used in various domains [29-31]. Lai *et al.* [29] presented a KaaS model for business network collaboration in the medical context. The main objective of this KaaS is to facilitate the interoperation and the collaboration among members in a knowledge network. In contrast to Disaster-CDM, their work did not tackle the data management layer from which the knowledge is provided nor did it address the storage aspect.

In the agricultural domain, Qirui [30] introduced the KaaS in order to provide farming recommendations according to user requirements and farming environment. The knowledge representation in their KaaS is based on ontologies and data are stored exclusively in a relational database (MySQL). Another interesting study is the work of Kannimuthu *et al.* [31] in the e-commerce domain. In their work, the KaaS purpose is the extraction of knowledge from data using data mining techniques. The extracted

knowledge assists in attracting users to buy other products of the same enterprise. In contrast to the approach proposed by Qirui [30] which stores data exclusively in relational database and that of Kannimuthu *et al.* [31] which stores data in XML database, the KaaS in Disaster-CDM accommodates both structured and unstructured data by taking advantage of relational databases and NoSQL data stores.

3.3 Related Simulation Work

Simulation is an established way of observing the behavior of a real-world system by developing models that represent the structure and behavior of the system of interest [32]. One of the main factors contributing to the increasing use of simulation involves its non-confinement to a specific discipline [33] as simulation is employed in a variety of domains, such as military operations, critical infrastructures, medical and life sciences, learning, and chemical and biochemical engineering.

Computer simulation, where computer models are developed to represent real-world scenarios, is supported by a variety of software simulation packages or simulation engines [34]. Although the act of simulation is not domain-specific, simulation packages are usually application-oriented, designed for simulation experiments in a specific domain. These packages use different modelling approaches, diverse technologies and a wide variety of domain-specific vocabularies. Moreover, simulation models are saved in simulators' engine-specific proprietary file formats. This heterogeneity presents an obstacle to querying, rule and constraint validation, as well as data integration.

Related work in simulation model querying and rule and constraint validation is presented first. It is followed by the review of ontology use in simulation modelling highlighting different roles of ontologies in the reviewed work and the research presented in this thesis.

3.3.1 Simulation Model Querying, Rule and Constraint Validation

Integration among simulators has attracted significant attention and resulted in a standard, the IEEE High Level Architecture [35], and numerous research studies [36,37].

Furthermore, semantic heterogeneity has been addressed by creating simulation ontologies [38,39].

Moreover, extensive research has been done on simulation model verification and validations [40,41]. Here, verification is the process of confirming that the model is implemented correctly while the validation checks that the model is accurate representation of the real system. In contrast, this work is concerned with how can simulation models be queried, and how can rules and constraints be written and model compliance with those rules and constraints validated. Nevertheless, research on the topic of simulation model querying, rule and constraints validation has been sparse.

Querying simulation mesh data has been addressed by Lee *et al.* [42]; however, their AQSim system is intended for querying mesh data and cannot be used with other non-mesh simulation models such as infrastructure networks or logistic systems. In contrast, the proposed SIMONTO focuses on infrastructure-like simulation models, transforms them into ontology-based representations and, as a result, enables simulation model querying and rule and constraint validation.

Querying from the perspective of model discovery and selection in component-based simulation model development has been addressed in the work of Szabo and Teo [43]. In their approach, the COSMO (COmponent Simulation and Modelling Ontology), ontology is applied as a terminology for describing the attributes and behavior of components. Consequently, ontology-based description is queried for the purpose of component discovery and selection. In contrast, SIMONTO represents simulation models as instances of an ontology and Disaster-CDM queries those ontology-based models after storing them in a NoSQL data store.

Rule and constraint validation in electronic systems domain can be performed using Property Specification Language (PSL) which has been standardized [44]. PSL is domain specific; it is intended for use with electronic system design languages. This work, on the other hand, is concerned with a range of application-oriented simulations packages related to disaster management with a special focus on infrastructure simulators.

3.3.2 Ontologies in Simulation Modelling

Although ontologies have been used in a variety of domains [45], their application to the field of simulation has been limited and primarily constrained within the research community. The potential of ontologies in simulation and modelling was explored by Lacy and Gerber [46]. From the perspective of these authors, ontologies are beneficial in simulation and modelling because they formalize semantics and allow querying, inference, sharing, and reuse of developed models.

The studies that are particularly relevant to our research are related to the use of ontologies to represent real-world scenarios for simulation purposes such as Tofani *et al.* [36], Miller *et al.* [38] and Silver *et al.* [47].

Tofani *et al.* [36] proposed an ontology framework to model the interdependencies among Critical Infrastructures (CI). Like our SIMONTO, the approach of Tofani *et al.* represents infrastructures as instances of an ontology and uses proprietary simulation packages for the simulation execution. However, this study creates ontology-based representations from existing simulation models, while Tofani *et al.* model CIs directly as instances of ontologies and then map them manually onto the proprietary simulation models. Therefore there are two main drawbacks to the work of Tofani *et al*: firstly, the CI network has to be modelled twice, as instances of an ontology and in the domain simulation language; and secondly, the mappings between ontology representations and simulation models must be established manually.

Miller *et al.* [38] investigated the development requirements and benefits of ontologies in Discrete Event Simulation (DES), and consequently, these authors presented the Discrete-event Modelling Ontology (DeMO). According to Miller and Baramidze [48], the main challenges in building DeMO, or a similar ontology for simulation and modelling, are twofold: firstly, it needs to be domain-independent, as DES can model any domain, and secondly, since simulation formalisms are founded in mathematics and statistics, the DES ontology should be based on the ontologies of those domains. DeMO captures generic discrete event simulation knowledge without addressing domain-specific simulation aspects. In contrast, SIMONTO approach uses simulator-specific ontologies;

therefore, it facilitates domain experts' understanding of ontologies and enables automated creation of the ontological representation from the proprietary simulation models.

Silver *et al.* [47] represented real-world scenarios as instances of the extended DeMO PIModel (Process Interaction Model). Subsequently, these instances are transformed to XPIM (Extensible Process Interaction Markup) instances, which are then translated to a JSIM (Java-based SIMulation) model. This approach models real-world scenarios in terms of an ontology, which may represent a challenge for domain experts that are accustomed to domain-specific simulation engines. Moreover, DeMO makes use of generic, domain-independent terminology that may differ significantly from specific domain terminology. Depending on the domain modelled, the majority of DeMO entities may be irrelevant and hence may obscure the modelling efforts. In contrast, the SIMONTO approach does not require modelling in an ontology form; it draws on existing proprietary simulation models to automatically generate its ontology-based representation. When new simulation models are needed, experts create them in the domain-specific simulation packages they are accustomed to using, and SIMONTO generates their corresponding ontology-based representation. Moreover, SIMONTO uses existing domain simulators for the simulation execution, while Silver *et al.* transform the ontology-model to JSIM for the simulation execution.

Like Miller *et al.* [38], Guizzardi and Wagner [39] also proposed a DES ontology. Their DES Ontology (DESO), a foundational ontology for discrete event system modelling, is derived from the Unified Foundational Ontology (UFO). In contrast to SIMONTO, whose objective is the representation of simulation models for querying and rule and constraint validation, or DeMO, whose aim is the representation of the real world for simulation purposes, the objective of DESO is to provide a basis for evaluating DES languages.

Benjamin and Akella [49] applied ontologies to facilitate semantic interoperability and information exchange between simulation applications. The ontology models for each simulation application domain are extracted from textual data sources, such as requirements and design documents. In the work of Benjamin and Akella [49], ontologies

describe different simulation domains, while in this research ontologies represent actual simulation models.

3.4 Summary

This chapter has surveyed related work. First, work in disaster data management has been examined, and the difference in focus between the reviewed work and the research reported in this thesis has been highlighted. Because this research proposes a KaaS-based solution for disaster data management, studies that apply the KaaS approach have been examined. Next, work related to simulation model querying and rule and constraint validation has been reviewed, and finally, the use of ontologies in simulation modelling has been presented.

Chapter 4

4 Disaster Cloud Data Management

A successful disaster management relies on the collaboration among participants; however, the diversity of the involved participants and their activities results in massive data heterogeneity. This heterogeneity of data, together with their volume, is one of the main challenges in providing a comprehensive solution that could be used by various stakeholders in diverse disaster situations. Disaster-CDM addresses those Big Data challenges by integrating storage in the cloud environment with the KaaS approach which provides disaster-related knowledge as a service.

This Chapter first introduces the overall Disaster-CDM framework in Section 4.1. Next, the two main parts of Disaster-CDM, knowledge acquisition and knowledge delivery are discussed in Sections 4.2 and 4.3.

4.1 Disaster-CDM Framework

The Disaster-CDM framework is illustrated in Figure 4.1 [50]. It consists of two parts: knowledge acquisition and knowledge delivery services. Knowledge acquisition is responsible for acquiring knowledge from diverse sources, processing it to add structure to unstructured or semi-structured data, and storing it. Heterogeneous data from sources like documents, simulation models, social media, and web pages, are handled by applying processes such as text extraction, file metadata separation, and SIMONTO simulation model transformation. This results in outputs including extracted text, annotated data, and ontology-based simulation models. Processed data are stored in a cloud environment, specifically in a variety of relational databases and NoSQL data stores. Knowledge delivery services are responsible for integrating information from different data stores and delivering knowledge to consumers as a service.

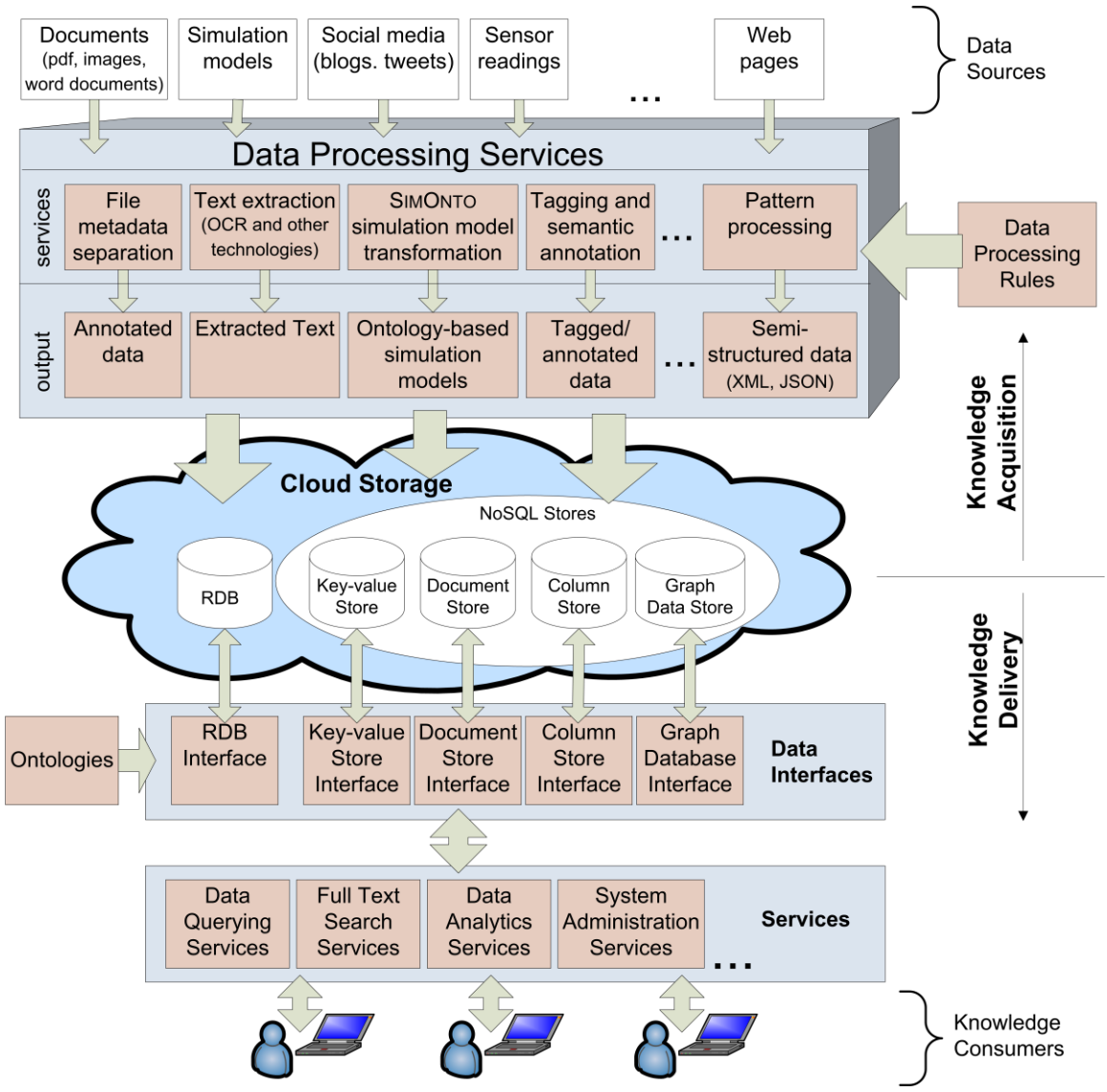

Figure 4.1: Disaster-CDM framework

The following two Sections 4.2 and 4.3 provide an overview of the two main parts of Disaster-CDM: knowledge acquisition and knowledge delivery.

4.2 Knowledge Acquisition

The knowledge acquisition services obtain data from heterogeneous data sources, process them, and store them in the cloud environment. It was decided to process the information and to store the processed, enriched data because this will allow shorter query response time than performing the processing "on the fly". For example tagging large text content, transforming simulation models and OCR on files with a large number of images may take time and storing already processed files will reduce query response time.

4.2.1 Heterogeneous Data Sources

A few examples of information related to disasters are disaster plans, incident reports, situation reports, social media, simulation models including infrastructure and health-care simulation. As for representation formats, examples include MS Word, PDF, XML, a variety of image formats (jpeg, png, tiff), and simulation model formats specific to simulation packages. Data representation is important because it determines the methods that can be used to add structure to unstructured or semi-structured data.

From our experience working with local disaster management agencies, the majority of information is stored in unformatted documents, primarily MS Word and PDF files. This agrees with the work of Hristidis *et al.* [1], who reported that most information is in MS Word and PDF files.

4.2.2 Data Processing Services

Because the input data are so diverse, they cannot be processed using a single approach. Therefore, the processing is driven by the input data and by *data processing rules*, as illustrated in Figure 4.1. *Data processing rules* specify what data processing services are to be applied to which input data and in which order. For example, a PDF incident report might go through *file metadata separation*, *text extraction*, and *pattern processing.* According to the KaaS approach, Disaster-CDM provides data processing services which can be composed by means of processing rules. The representative services with their associated outputs are included in Figure 4.1:

File Metadata Separation Service makes use of file and directory attributes, including file name, creation date, last modified date, and owner. File names themselves carry important information about content because they are typically chosen with the aim of describing the content. They are processed to separate the words contained in the file name. The creation date and last modified date can assist in distinguishing newer and potentially more relevant information from older and possibly outdated information. The file directory structure contains additional information about file content since directories are used to organize files. Directories can be seen as a categorization and therefore are included in metadata separation.

Text Extraction Service recognizes the text in a file and separates it [51]. An example of such process is optical character recognition (OCR) which Disaster-CDM uses to extract text from images. This step prepares images, MS Office and PDF files for other processing steps such as tagging. Text extraction is especially important in the case of diagrams such as flowcharts or event-driven process chains because these documents contain large amounts of text that can be used for tagging. In the case of MS office files, text is extracted from document body as well as from the images embedded in the document as they may also contain relevant information.

Pattern Processing Service makes use of existing patterns within documents to extract the desired structure. Hristidis *et al.* [1] observed that most of available disaster-related information is stored in unstructured documents, but that "typically the same organization follows a similar format for all its reports" [1]. Therefore, it is feasible to use patterns for information extraction. However, the number of organizations involved in disaster management is large, which may result in a large number of patterns. This represents a challenge because the patterns need to be identified before pattern processing can be applied. Another challenge is with new data sources where patterns need to be indentified manually.

SIMONTO Simulation Model Transformation Service is responsible for converting simulation models into a representation which enables model queries and integration with other disaster-related data. To extract as much information as possible from simulation

model files, an ontology-based representation of simulation models has been used [52,53]. Unlike text-processing approaches, an ontology-based representation makes it possible to:

- address simulator-specific terminology,
- remain schema-independent because ontologies do not have predefined schema,
- focus on entities and their relations.

SIMONTO transforms existing models in the simulator-specific file formats to their corresponding ontology-based representations. Those ontology-based simulation models are then stored in a NoSQL data store which facilitates integration with other data, querying, and rule and constraint validation.

Tagging and Semantic Annotation Services. Tagging is the process of attaching keywords or terms to a piece of information with the objective of assisting in classification, identification, or search [54]. Semantic annotations additionally specify how entities are related. In disaster management data tagging, both manual and automated tagging are needed. Automated tagging applies various natural language processing (NLP) and soft computing techniques to add tags automatically to pieces of information. Because disaster data are immensely diverse, it might not be feasible to tag all content automatically. Images are examples of data which may require computationally expensive tagging. Therefore, manual tagging is used to supplement the automated approach. Tagging will be explored in this study, while semantic annotations will be addressed in future work.

The presented data processing services are common processes for addressing file-style data; nevertheless, Disaster-CDM can be easily expanded to include new data processing services.

4.2.3 Data Storage in the Cloud Environment

Cloud computing offers a number of advantages over traditional approaches as discussed in Section 2.2. Moreover, Section 2.2 also pointed out the main attributes contributing to the choice of cloud environment for the storage of disaster data including: high scalability

and availability. In disaster data management, availability is greatly affected by replication strategy. Data should be replicated across data centers placed on geographically distant locations; therefore, if the region is affected by a disaster and a local data center fails, the system continues to be operational as a remote data center remains unaffected.

As illustrated in Figure 4.1, for data storage Disaster-CDM uses both relational database and NoSQL data stores. As discussed in Section 2.3, NoSQL data stores were designed to address Big Data challenges while taking advantage of cloud computing environments. Moreover, NoSQL data stores have a number of characteristics making them an adequate solution for disaster data management including horizontal scalability and flexible data model. Horizontal scalability enables NoSQL stores to take advantage of the cloud environment by scaling over a number of nodes. Flexible data model is crucial for storage of disaster data due to an immense variety of data that needs to be stored. On the other hand, NoSQL data stores are designed for different purposes and therefore not all problems can be gracefully solved using the same data store. Consequently, Disaster-CDM does not restrict storage to a specific NoSQL data model, but allows for the choice of storage according to the characteristics of the data to be stored.

Despite the advantages of NoSQL data stores, Disaster-CDM also accommodates relational databases. RDBs are still an appropriate solution for many applications because of their characteristics such as ACID (Atomicity, Consistency, Isolation, Durability) transactions, their status as an established technology, and their advanced query capabilities. Moreover, existing data in relational databases do not need to be migrated.

Additionally, if data are available in a form similar to a relational data model, a relational database can be used. Examples of online databases providing data in table-like form include The Canadian Disaster Database [55] and EM-DAT, The International Disaster Database [56]. By storing such data in a relational database, the structure of the data is preserved and data acquisition processing is reduced. This data stored in a relational database need to be integrated with data from NoSQL stores; however, integration among

relational databases and NoSQL data stores is a challenge. Part of this challenge is the fact that NoSQL data stores do not support a standard query language.

4.3 Knowledge Delivery

The Disaster-CDM knowledge delivery services answer information requests submitted by service consumers by integrating data stored in the cloud environment. In this stage, the collaboration is achieved by providing the integrated knowledge as a service to collaboration participants. As presented in Figure 4.1, the data access is mainly composed of three parts:

- **Ontologie**s: These provide an overall view of the local ontologies representing each data store independently of its category. Ontologies represent the mapping between heterogeneous sources which is needed to unify query capabilities.
- **Data interfaces:** After querying the ontology, it is necessary to access the data. Data interfaces enable translation of the generic query into a specific language that corresponds to the underlying data store system. Thus, the data stored in heterogeneous sources can be accessed, analyzed, and administered. An attempt to unify access to NoSQL systems is proposed in the work of Atzeni *et al.*[57] where NoSQL models and their programming tactics are reconciled within a single framework.
- **Services:** This is the access layer for users. It provides services independently of how the data are stored. Thus, users are unaware of the storage architecture and are provided with a unified view of the data. Examples of provided services are full-text search, data querying, data analytics, and system administration services.

The application of the proposed Disaster-CDM approach on data formats commonly present in the disaster management domain, i.e. file-style data formats, is further detailed in Chapter 5.

4.4 Summary

This chapter has proposed Disaster-CDM, a Knowledge as a Service (KaaS) framework for disaster cloud data management. Disaster-CDM addresses Big Data challenges,

including data heterogeneity and volume, by integrating storage in the cloud environment, specifically NoSQL data stores and relational databases, with the KaaS approach which provides disaster-related knowledge as a service. The two main parts of the Disaster-CDM framework have been discussed: knowledge acquisition and knowledge delivery services. Knowledge acquisition is responsible for acquiring knowledge from diverse sources, processing it to add structure to unstructured or semi-structured data, and storing it in data stores. Knowledge delivery services are responsible for integrating information from different data stores and delivering knowledge to consumers as a service.

Chapter 5

5 Disaster-CDM for File-style Data

The Disaster-CDM framework is designed to accommodate heterogeneous data sources, including PDF files, MS Word documents, simulation models, Web pages, and social media data. The introduction of a new data source to the framework requires:

1. Adding new processing services to existing data processing capabilities. For example, video processing would require a new service which would attach textual context to videos. Such a textual context is essential for effective search and querying of video sources.
2. Defining data processing rules for the new data source. For instance, a video processing rule might specify that video files first undergo metadata extraction followed by a new video-specific service.
3. Determining the data storage appropriate for the new data source. Disaster-CDM does not define storage data structure or even the type of data store; in this step, the data store type suitable for the new data source is determined and the storage data model is designed.

From our experience working with local disaster agencies, which agrees with the work of Hristidis *et al.* [1], the majority of information is stored in unformatted documents, primarily MS Word and PDF files. Another crucial element of disaster management is simulation because it provides a means of studying the behaviour of critical infrastructures, as well as a way of exploring disaster response "what-if" scenarios. Consequently, this chapter focuses on processing information stored in files, including:

- plain text,
- image files,
- MS Office documents including Word, PowerPoint, Excel, and Visio,
- PDF files, and
- simulation model files.

The common element among those information sources is that information is typically stored in self-contained and largely unrelated files.

The following sections describe the steps of introducing file-style data into the proposed Disaster-CDM framework: data processing services, data processing rules, and storage in the cloud environment.

5.1 Data Processing Services

The main data processing services required to handle file-style data are included in Figure 4.1 and were discussed in Section 4.2.2. With respect to processing file-style data common in disaster management domain, data processing services are applied as follows:

- **File metadata separation service** is used in processing anything that is stored as a file. Since metadata attributes vary among different file formats, resulting data annotations will also differ in annotation types.
- **Text extraction service** applies various technologies according to the type of file that is being processed. For example, to extract text from image files or from images embedded in MS Word or Visio files, optical character recognition (OCR) technologies are applied.
- **SIMONTO simulation model transformation service** is the process specific for simulation model files; nevertheless, it is applicable for various simulation model file formats.
- **Tagging and semantic annotation services** are applied on textual data; however, in the case of images or PDF files, text is first extracted from the image or PDF files and then passed on for tagging and semantic annotation. All files are tagged and semantically annotated unless other processes were unable to extract any text from the file.

Pattern processing service could potentially add more structure to processed data; however it is associated with a number of challenges including: patterns need to be known before processing, only a limited subset of files conforms to a specific pattern

with possible existence of a large number of patterns. Therefore, this work does not further address pattern processing service.

5.2 Data Processing Rules

Data processing rules define how a category of data sources needs to be processed before being stored in a data store. They are influenced by the format of the data source and the available processing services.

For example, Listing 5.1 illustrates a data processing rule for all MS Office files. First, metadata are separated (line 2), and text is extracted (line 3). Next, if there are images in the file, they are extracted (line 4). For each image, text is separated using OCR methods (lines 5 to 7). Finally, text extracted from the file and from the images is tagged (lines 8 and 9).

Listing 5.1: Data processing rule for MS Office files

```
1:  if file = MSOfficeFile then
2:      processMetadata(file)
3:      fileText = extractText(file)
4:      images = extractImages(file) //extract all images
5:      for each image in images
6:          imageText += OCRProcess(image)
7:      end for
8:      tagText(fileText)
9:      tagText(imageText)
10: End
```

The presented data processing rule represents a generic processing for all MS Office files regardless of file type. However, some MS Office files, such as Excel files, possess additional formatting that can be exploited to add additional structure to data. For example, since Excel organizes data in tabular form, data processing can take advantage of this formatting and create table-like structures in a data store. In this case, a service needs to be added which can take advantage of this specific formatting, and the data processing rule needs to be refined to include Excel-specific processing service.

Listing 5.2 illustrates a data processing rule for PDF files and Listing 5.3 shows a rule for image files.

Listing 5.2: Data processing rule for PDF files

```
1:  if file = PDFFile then
2:      processMetadata(file)
3:      fileText = extractText(file)
4:      tagText(fileText)
5:  End
```

Listing 5.3: Data processing rule for images

```
1:  if file = image then
2:      processMetadata(file)
3:      imageText = OCRProcess(file)
3:      tagText(imageText)
3:  End
```

Another category of files that is particularly significant in disaster data management is simulation files. An example of a processing rule for simulation models is presented in Listing 5.4. Like the MS Office rule, it starts with metadata separation (line 2). Next, SIMONTO transforms the simulation model to its corresponding ontology-based representation (line 5), which is described in an ontology representation language. Such an ontology-based representation then needs to go through additional processing service, *postProcessOntology*, to prepare it for tagging. This processing service deals with specifics of the ontology representation language; for example, it replaces special characters with spaces and separates compound words such as those in camel-case naming to assist subsequent tagging. Finally, the same as MS Office rule, the simulation model processing rule ends with text tagging (line 7).

Listing 5.4: Data processing rule for simulation models

```
1:  if file = SimulationModel then
2:      processMetadata(file)
3:      //SIMONTO - Transform simulation model to its
4:      // corresponding ontology-based representation
5:      ontModel = transformSimModelToOntology(file)
6:      fileText = postProcessOntology(ontModel)
7:      tagText(fileText)
8:  end
```

Similarly to these rules for MS Office files and simulation model files, rules are defined for other file categories that need to be processed, including plain text files, PDF files and a variety of image formats.

Overall, generic file processing consists of separating metadata, extracting text from source files using file type-specific processing followed by tagging of extracted text. When a source file contains additional formatting, such as in Excel documents, data processing rules can use this to add additional structure to processed data.

5.3 Data Storage in the Cloud Environment

Flexibility of data storage is the core of the proposed Disaster-CDM framework because it enables a choice of storage according to the characteristics of the data to be stored. For each data source category, two steps must be performed:

- determining the type of data store, and
- designing the storage data model.

Determining the type of data store consists of choosing among relational database, key-value, document, column-family, and graph stores. The file-style data considered in this chapter are stored in self-contained, apparently unrelated files. Although the file contents might be related, this relation is not explicitly specified. Therefore, storage models focusing on relations, including relational and graph databases, are not the best suited for such data. The document data store model has been chosen here for the storage of file

data because it is designed around the concept of a document, providing flexible storage while allowing structure specification within a document.

The storage data model design in the case of a document data store consists of defining a document structure. Document data store implementations differ in their internal representations of documents; however, they all encapsulate and encode data in some form of encoding. Therefore, the data model design is independent of the choice of data store implementation provided that the data store belongs to the document category.

Table 5.1 depicts the data model designed for storing file data in a document data store. It is a generic model for storing a variety of file-style data with flexibility that enables it to accommodate different file types and a variety of attributes. The proposed data model is relatively standardized to support querying abilities. In contrast, allowing uncontrolled naming of fields within documents would negatively impact querying abilities. Several fields, such as *fileName* or *origFileLocation*, are mandatory because they are common for all file types and must exist in each document in the data store. On the other hand, other fields such as *docImageText* and *tag* are optional and exist only in documents that need to record those attributes. Two fields, *metaData* and *tag*, have a number of child fields for storing different attributes of the parent field. The number and names of the child fields are different among files of different types: for example, an image file might have *metaData* child fields such as *imageWidth* or *resolutionUnits*, but these child fields will not exist for other file types. With respect to *tag* fields, the number and names of the child fields depends on the tagging approach used.

To accommodate other types of data the data model from Table 5.1 can be extended by adding new fields. For example, to handle geolocation new fields would be added to the model to record geographical location. If a document contains several entities with different geolocations, each entity would have child fields identifying its location. Consequently, this would allow for inclusion of geolocation is a search queries.

Table 5.1: File storage data model – document data store

Field Name	Child field name	Mandatory	Description
fileName		✓	Name of the original file
origFileLocation		✓	Full file path of the original file
origFileMachine		✓	Name of the computer from which the file originated
DBLoadDateTime		✓	Date and time that file was processed by Disaster-CDM
contentType		✓	Type of the content, such as PDF, MS Word, or MS PowerPoint
metaData	modified	✓	Metadata, including generic data such as creator and modified and created date and time. File-specific metadata such as number of slides or word count are also included here.
	created	✓	
	creator	✓	
	...	×	
docText		×	Text extracted from files, not including text from images.
docImageText		×	Text extracted from images.
tag	[]	×	Arrays of generic tags ([]), as well as arrays of dates, organizations, locations, and persons found in the file text.
	date []	×	
	organization []	×	
	location []	×	
	person []	×	
	...	×	
_attachment		✓	File in its original form

5.4 Summary

This chapter has focused on Disaster-CDM for file-style data, which are common in the disaster management domain. The generic process of adding a new data source to the proposed framework has been introduced and then applied for file-style data sources. In the first step, data processing step, various data processing services and their role in file-style data processing are defined. Rules for processing file-style data are introduced in the second step. Finally, in the third step, data storage step, the motivation for choosing

document data store has been explained and the data model for storing file-style data in a document data store has been presented.

Chapter 6

6 Ontology-Based Representation of Simulation Models

This chapter presents SIMONTO, an ontology-based simulation model representation. In the context of Disaster-CDM, SIMONTO is responsible for transforming simulation models into their corresponding ontology-based representations. Because SIMONTO is graph-based, the proposed simulation model graph is presented first. Next, the SIMONTO architecture is portrayed.

6.1 Simulation Model Graph

The ontology-based representation of the simulation model is founded on graph theory. In a simulation, the direction of the interaction or the dependence among entities is often significant; for example, in a transportation problem or in a provider-consumer arrangement, connections among entities have a specific direction. Consequently, a directed graph model [58] is used.

Graph representations have been used for semantic Web search. Tran *et al.* [59] applied a graph-structured data model to represent resources on the Web as well as for search query representation. Moreover, their proposed semantic search strategy takes advantage of graph techniques. Delbru *et al.* [60] also made use of graph theory; they proposed entity retrieval and a high-performance indexing model for searching semi-structured Web documents by taking advantage of a labelled directed graph. They defined a labelled directed graph model which encompasses different types of Web data sources, including Resource Description Framework (RDF), RDF Schema (RDFS), and Microformats, and represents corresponding datasets, entities, and their relationships. In contrast to the work of Tran *et al.* and Delbru *et al.*, who applied graph models in Web search, this work exploits graphs to represent simulation models.

In addition to making use of a graph-structured representation, the SIMONTO simulation model exploits ontology formalisms. Since the Web Ontology Language (OWL) has been recommended by the World Wide Web Consortium (W3C) and has emerged as the

primary ontology specification language [45,61], this work uses OWL. Designed as an ontology language for the Semantic Web, OWL [62] has been established on the basis of RDF [63] and RDFS [64]. In particular, the fundamental mechanisms for describing classes and properties as well as their respective hierarchies are inherited from RDFS. In OWL terminology:

- a *class* is a collection of similar entities,
- an *individual* is an actual object in a domain. An *instance* refers to a class membership; individuals are *instances* of classes, and classes can be *instances* of other classes.
- a *data value* refers to a value of an attribute,
- *properties* establish relations:
    - an *object property* establishes relations between individuals,
    - a *datatype property* specifies attribute values by relating individuals and data values.

OWL is characterized by a formal semantics and an abstract ontology structure that can be perceived as a graph. Consequently, elements of the simulation model graph proposed in this work correspond to OWL elements:

- vertices:
    - *entity vertices* represent simulation entities or groups of entities and correspond to OWL *individuals* and *classes*,
    - *data value vertices* represent data values and are analogous to OWL *data values.*
- arcs:
    - *attribute arcs* relate entities to data values and correspond to OWL *datatype properties,*
    - *relation arcs* establish relations between two simulation entities and are analogous to OWL *object properties.*

Consequently, this work defines a simulation model graph as follows:

Definition 1: A *simulation model graph S* is a directed graph $S(V^S, A^S)$, where:

- V^S is a finite set of *vertices*; it is conceived as the disjoint union of *entity vertices* (*E-vertices*) V_E^S representing simulation entities and *data value vertices* (*V-vertices*) V_V^S representing data values:

$$V^S = V_E^S \cup V_V^S \tag{1}$$

- A^S is a finite set of *arcs* of the ordered form $a(v_1, v_2)$ with $v_1, v_2 \in V^S$. Two types of arcs are distinguished:
  - A_A^S : *A*-arc or *attribute arc*
  - A_R^S : *R*-arc or *relation arc*

$$A^S = A_A^S \cup \mathrm{A}_R^S \tag{2}$$

$$A^S = \begin{cases} A_A^S = \{a(v_1, v_2) \mid v_1 \in V_E^S \wedge v_2 \in V_V^S\} \\ A_R^S = \{\, a(v_1, v_2) \mid v_1, v_2 \in V_E^S\} \end{cases} \tag{3}$$

$$V^S \cap A^S = \emptyset \tag{4}$$

$$A_A^S \cap \mathrm{A}_R^S = \emptyset \tag{5}$$

E-vertices are simulation model entities, which are relevant objects for the observed system, while *V*-vertices represent data values. The *A*-arcs denote entity datatype properties by connecting entities (*E*-vertices) to data values (*V*-vertices), indicating a measure of an attribute. The relations between the two entities of the simulation model, the two *E*-vertices, are established with *R*-arcs.

Example: Figure 6.1 shows an example of a simulation model graph. It displays a fragment of an EPANET [65] water distribution network represented as a simulation model graph. Specifically, the model includes five individuals (*E*-vertices): pipes 36, 37, and 218, reservoir 264, and junction 40. The *A*-arcs include diameter, length, initial status, and total head; they define attribute values by linking entities to data values. The *R*-arcs establish the relationship between entities, such as in the statement, “pipe 218 has

start node junction 40", where "pipe 218" and "junction 40" are *E*-vertices and "has start node" is an *R*-arc.

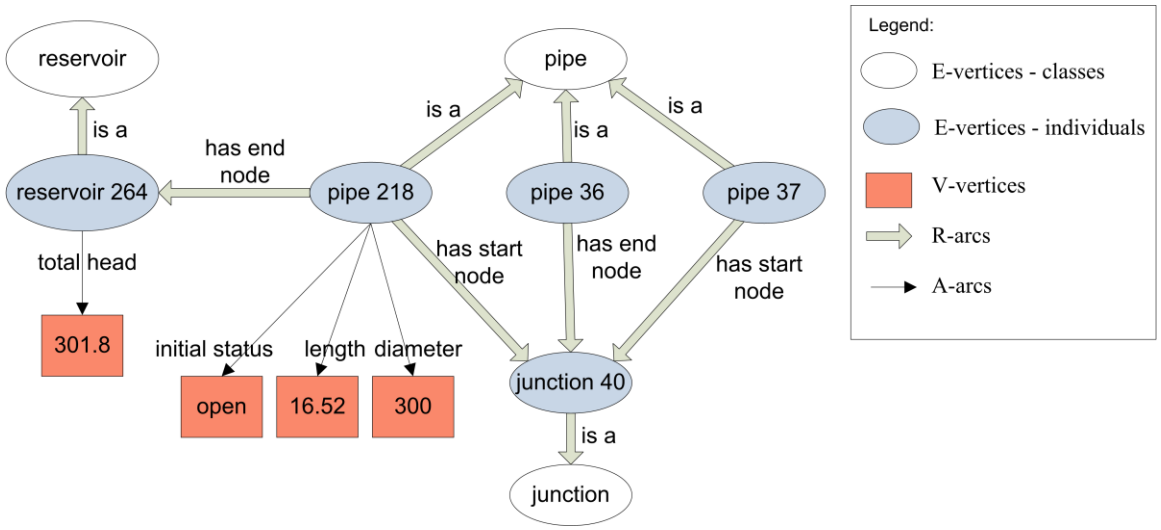

Figure 6.1 A SIMONTO graph-structured EPANET simulation model

The graph in Figure 6.1 can be perceived as an ontology-based graph, where the *E*-vertices are *individuals* and *classes*, the *V*-vertices are *data values*, the *R*-arcs are *object properties*, and the *A*-arcs are *datatype properties*. Individuals are contained in classes as indicated in Figure 6.1 by "is a" relations: pipes 36, 37, and 218 belong to the *pipe* class, junction 40 is in the *junction* class, and reservoir 264 belongs to the *reservoir* class. Therefore, an ontology related to Figure 6.1 contains the classes *pipe*, *reservoir*, and *junction*. The domain of the "has end node" object property includes the class *pipe*, while the range includes the *reservoir* and *junction* classes. The classes and properties contained in the ontology depend on the simulation domain as well as on the simulation package used for model creation.

The definition of a simulation model graph, as explained in Definition 1 and the observed relationship with ontology paradigms are the foundation of SIMONTO ontology-based simulation models.

6.2 SIMONTO Architecture

The overall SIMONTO architecture is presented in Figure 6.2. The SIMONTO inputs are the proprietary simulation models represented in their simulator-specific file formats. Specifically, the SIMONTO Engine uses proprietary simulation models to create their corresponding ontology-based simulation models. The resulting ontology-based simulation models are persisted in a data store, consequently enabling various services including integration, simulation model querying, and rule and constraint validation.

The following Sections 6.2.1, 6.2.2, 6.2.3, and 6.2.4 describe the SIMONTO components from Figure 6.2: SIMONTO ontologies, the SIMONTO Engine, storage for ontology-based simulation models, and simulation services.

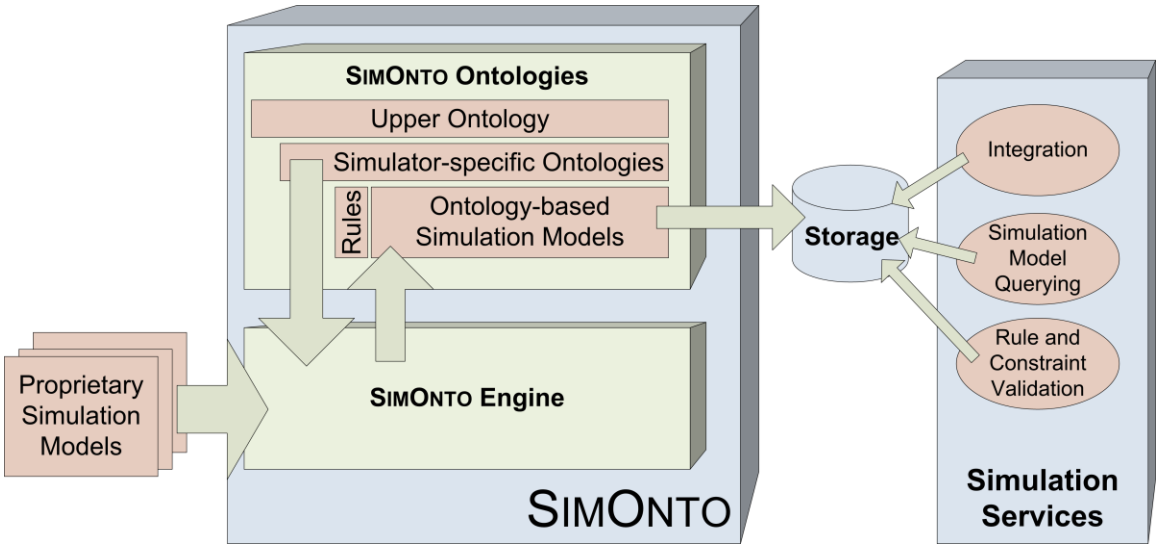

Figure 6.2: Overall SIMONTO architecture

6.2.1 SIMONTO Ontologies

To separate different concerns, the SIMONTO ontologies block has a layered design, as depicted in Figure 6.3 [53]. The top layer, the upper ontology, introduces general concepts which are common across different simulation domains. The second layer, or

the simulator-specific ontologies layer, defines the ontologies of domain-specific simulation packages by extending the upper ontology. Ontologies in this layer are the inputs to the SIMONTO Engine, as illustrated in Figure 6.2. The third layer contains the ontology-based simulation models created by the SIMONTO Engine. In this layer, each simulation model from the proprietary model file is represented as an ontology-based model. The rules represent an addition to ontology-based simulation models and act upon them.

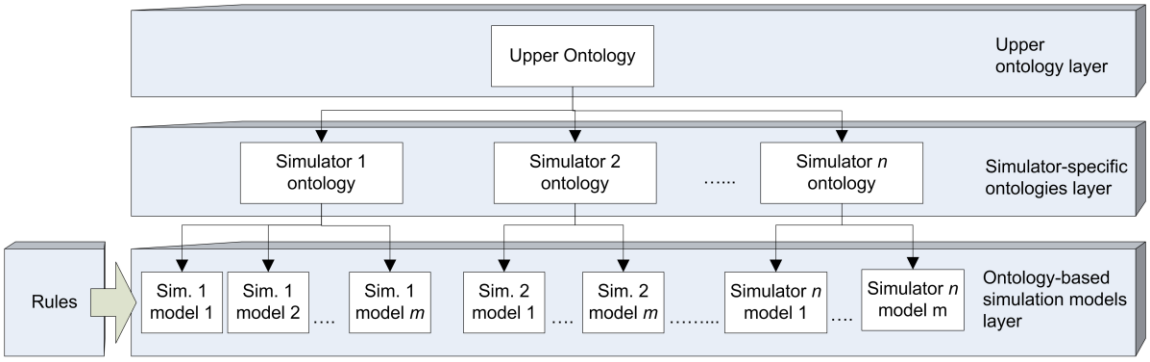

Figure 6.3: SIMONTO ontologies

Upper Ontology Layer

The top layer consists of the upper ontology, which contains generic concepts which are common to all simulation engines. The upper ontology's purpose is to provide a set of concepts on which other ontologies can be constructed and to support broad semantic interoperability among other ontologies. Based on Definition 1, the upper ontology can be defined as follows:

Definition 2: The *upper ontology* is the set:

$$O^U = \{C^U, P^U\} \tag{6}$$

where:

C^U is the set of upper ontology classes and

P^U is the set of upper ontology properties.

The classes C^U of the upper ontology are *E*-vertices, while the properties P^U are arcs of the simulation model graph from Definition 1.

Example: Figure 6.4 [52] portrays the *upper ontology* classes. The *cell* is an entity that transforms inputs into outputs. The *channel* transports entities between *cells* and/or *controls*, while *controls* are responsible for distributing the flow of entities among channels. *Meters* are responsible for performance measures, while *other* serves as a category for entities that cannot be assigned to any of the other four categories.

The only properties, or arcs in the graph representation included in the upper ontology, are object properties *hasInput*, *hasOutput*, and their inverse properties *hasStartNode* and *hasEndNode*.

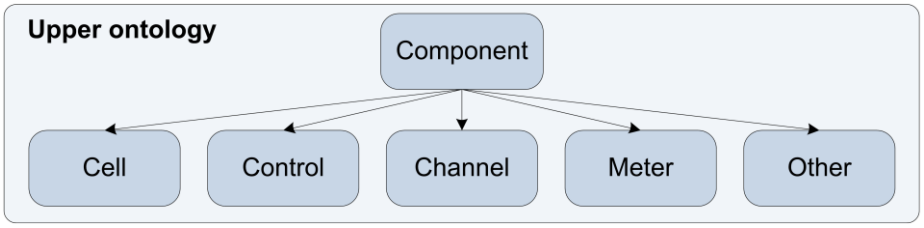

Figure 6.4: Upper ontology classes

Simulator-Specific Ontologies Layer

The simulator-specific ontologies layer consists of ontologies that are specific to the actual simulators. This layer provides the simulator-specific entities needed to describe individual simulation models. Thus, the terminology matches that of the simulators, making it easier for domain experts to understand the ontologies as well as enabling automated creation of ontology-based representations from proprietary simulation models. In this layer, there is one ontology for each simulation package. A simulator-specific ontology is defined as:

Definition 3: The *simulator-specific ontology* for the *i*-th simulation package is the set:

$$O_i^S = \{C_i^S, P_i^S\} \tag{7}$$

$$C_i^S = \{c_i^S \mid c_i^S \sqsubseteq x \wedge x \in C^U\} \tag{8}$$

where:

C_i^S is the set of the *i*-th simulation package classes such that each class is a subclass of an upper ontology class. ($\sqsubseteq$ indicates class/subclass relation: 'c_i^S is subclass of x')

P_i^S is the set of the *i*-th simulation package properties.

This definition provides limitations on the class definitions in this layer: each class defined in this layer must be a subclass of a class in the upper ontology layer. Because the upper ontology contains highly generic simulation concepts, this definition enables further division of classes in the simulator-specific ontologies layer.

Example: An example of a simulator-specific ontology, specifically the EPANET water distribution simulator ontology, with its relation to the upper ontology is illustrated in Figure 6.5. To keep the illustration simple, this figure includes the EPANET ontology classes, but not their properties. It can be observed that each EPANET ontology class is a subclass of the upper ontology class.

Although there are class restrictions at this layer, limitations on properties are not imposed. Therefore, at this level, properties can be independently defined, eliminating the need to identify properties as sub-properties of the upper ontology layer. This approach to property identification was chosen because properties vary greatly across domains and even among simulators in the same domain. As a result, the process of assigning each property into an upper-ontology category might cause implementation challenges. Although it is still possible to define properties as sub-properties of the upper ontology, properties can also remain independent of the upper ontology. In the case of the EPANET ontology, its properties are not sub-properties of the upper ontology.

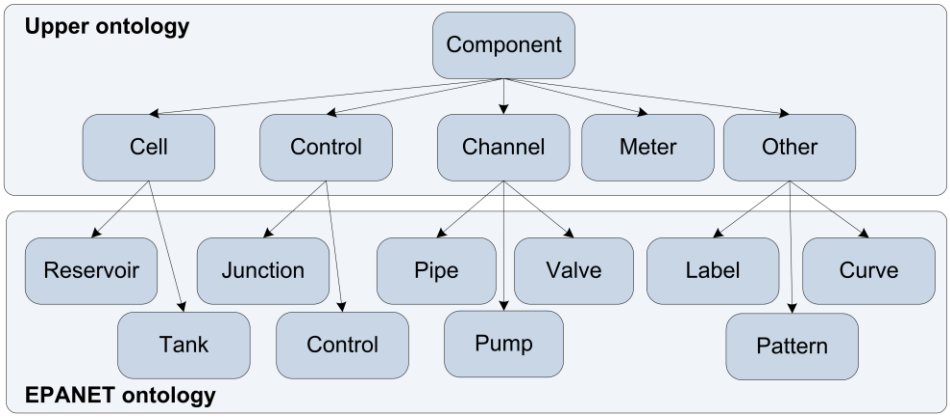

Figure 6.5: EPANET ontology with relations to the upper ontology

Furthermore, it is important to highlight the distinction between datatype properties *DP* and object properties *OP*:

$$P_i^S = OP_i^S \cup DP_i^S \,. \tag{9}$$

With respect to the simulation model graph in Definition 1, datatype properties are *A*-arcs, while object properties are *R*-arcs. The significance of distinguishing between datatype and object properties in simulation models is that the datatype properties of a single ontology individual can be established without the knowledge or existence of other individuals, while object properties require knowledge about another individual. This has a major impact on formulating an algorithm for creating ontology-based simulation models from proprietary simulation models.

Ontology-Based Simulation Model Layer

The ontology-based simulation model layer contains ontology-based simulation models that are represented as instances of simulator-specific ontologies. More specifically, each simulation model, usually contained in a simulation engine proprietary model file, is represented as an ontology-based simulation model consisting of interconnected instances of the simulator-specific ontology. Different simulation models from distinct proprietary

files correspond to the various models in this layer. Consequently, the ontology-based simulation model can be defined as follows:

Definition 4: The *ontology-based simulation model* for the *j*-th simulation model of the *i*-th simulation package is the set:

$$O_{ij}^{M} = \{I_{ij}^{M}, PI_{ij}^{M}\}, \tag{10}$$

$$I_{ij}^{M} = \{\, a \mid a{:}\, c \wedge c \in C_i^S \,\}, \tag{11}$$

$$PI_{ij}^{M} = \{p(x,y) \mid p \in P_i^S \wedge x \in I_{ij}^{M}, y \in (I_{ij}^{M} \cup AV)\}, \tag{12}$$

where:

AV is the set of data values.

I_{ij}^{M} is the set of *individuals a*. Each individual *a* is an instance of a class *c* from the set of simulator-specific ontology classes C_i^S.

PI_{ij}^{M} is the set of all *instantiated properties p*. Each property *p* is instantiated from the properties P_i^S defined in the simulator-specific ontology layer.

Example: An example of an ontology-based simulation model is portrayed in Figure 6.1. The set of *individuals* includes the actual objects from the simulation model: reservoir 264, pipes 36, 37, and 218, and junction 37. The set of *instantiated properties* includes individual occurrences of properties defined in the simulator-specific ontology. For example, the property *hasStartNode* is defined in the EPANET ontology, and in Figure 6.1 it appears twice, indicating two occurrences of the *hasStartNode* relation: *hasStartNode(pipe218, junction40)* and *hasStartNode(pipe37, junction40)*.

As shown in Definition 4, the ontology-based simulation model consists of individuals and instantiated properties. Since this definition does not permit the formation of new classes or properties in this layer, all classes and properties must be defined in the simulator-specific ontologies layer. Consequently, once a simulator-specific ontology has been created for each simulation engine, the creation of ontology-based simulation models can be automated.

As in the simulator-specific ontology case, in the ontology-based simulation model, object and datatype properties are distinguished from one another. In the example from Figure 6.1, all occurrences of *A*-arcs compose the datatype properties set, while the object properties set includes all occurrences of *R*-arcs. The set of instantiated properties is:

$$PI_{ij}^{M} = DPI_{ij}^{M} \cup OPI_{ij}^{M} \tag{13}$$

$$DPI_{ij}^{M} = \{p(x,y) \mid x \in I_{ij}^{M} \wedge y \in AV\}, \tag{14}$$

$$OPI_{ij}^{M} = \{\, p(x,y) \mid x,y \in I_{ij}^{M} \,\}, \tag{15}$$

where:

DPI_{ij}^{M} is the set of instantiated datatype properties assigning attribute values AV to individuals I_{ij}^{M},

OPI_{ij}^{M} is the set of instantiated object properties establishing relations between two individuals *x* and *y*.

Example: In the example from Figure 6.1, all occurrences of *A*-arcs compose the set of instantiated datatype properties DPI_{ij}^{M}, while the set of instantiated object properties OPI_{ij}^{M} includes all occurrences of *R*-arcs. An example of an instantiated datatype property is *hasDiameter(pipe218, 300)*, while *hasStartNode(pipe218, junction40)* is an instantiated object property.

Rules

Although ontologies establish a way of describing knowledge with defined semantics, they do not provide a method for defining procedures to extract new knowledge from existing assertions. Consequently, in Berners-Lee's Semantic Web Stack [66], rules are the next hierarchical layer after ontologies.

Accordingly, SIMONTO includes the rules which act upon ontology-based simulation models created by the SIMONTO Engine. Rules are intended for situations in which ontology-based specifications are not sufficient and additional expressiveness is required

to represent a complete simulation model. Additionally, they can also express rules and constraints to which the simulation model should conform.

6.2.2 The SIMONTO Engine

The SIMONTO Engine is responsible for the creation of an ontology-based simulation model representation. As illustrated in Figure 6.6, the SIMONTO Engine inputs consist of the simulator-specific ontology and the proprietary simulation model. The simulator-specific ontology is simulation package-specific and captures simulation package components, vocabularies, and functionalities. On the other hand, the proprietary simulation models are model-specific, with each model stored in a separate model file. The output of the SIMONTO Engine is the ontology-based simulation model represented as interconnected instances of the simulator-specific ontology.

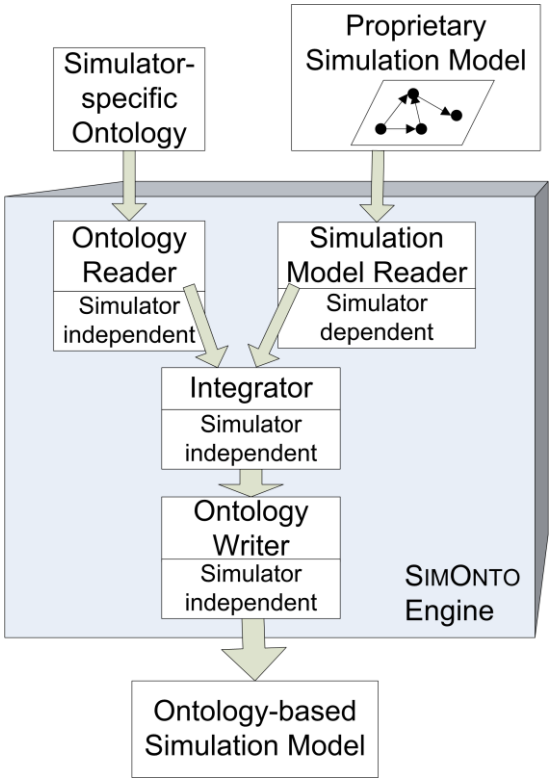

Figure 6.6: The SIMONTO Engine

The four SIMONTO Engine components are: Ontology Reader, Simulation Model Reader, Integrator, and Ontology Writer.

Ontology Reader is responsible for reading simulator-specific ontologies. Although ontologies are simulator-specific, they are always represented using the common ontology language, which allows a simulator-independent reader. Specifically, the Ontology Reader is responsible for acquiring information about simulator-specific classes and their properties, including datatype and object properties. The Ontology Reader is not aware of individuals because the simulator-specific ontologies contain only classes and their properties; however, individuals will be extracted by the SIMONTO Engine.

Simulation Model Reader is responsible for reading the second SIMONTO Engine input, the proprietary simulation model. Since the format of a proprietary simulation model depends on a specific simulator, a separate Simulation Model Reader has to be created for each simulator having models that require transformation to an ontology-based representation. Therefore, there will be one Simulation Model Reader for each simulator. However, once a Simulation Model Reader has been created for a specific simulator, the reader can transform any model represented in that format. The Simulation Model Reader design depends on the model being read, and the reader can use the simulator's API interface, directly read the model file, or employ external model readers.

The **Integrator** receives the data from the Ontology Reader and the Simulation Model Reader and creates the ontology-based simulation model. Specifically, the Integrator receives information about simulator-specific classes from the Ontology Reader. For each class, the Integrator obtains knowledge about its individuals and their data properties from the Simulation Model Reader. After acquiring information about all individuals of all classes and their datatype properties, the Integrator proceeds to determine object properties. Because object properties connect individuals of the same or different classes, all individuals must be determined before object properties are defined. Subsequently, the Integrator sends information about classes, individuals, and properties to the Ontology Writer.

The **Ontology Writer** is responsible for writing an ontology-based simulation model in an ontology language such as OWL. Rather than re-creating classes, the output ontology imports the simulator-specific ontology to acquire domain-relevant concepts and properties. The Ontology Writer then identifies individuals and properties using information received from the Integrator and records the output in an ontology language. As its purpose is to write ontologies from the Integrator's information, the Ontology Writer is simulator-independent.

Algorithm 6.1 illustrates the process of creating ontology-based representations of simulation models performed by the SIMONTO Engine. The result of this algorithm is the Ontology-based Simulation Model O_{ij}^{M} consisting of *individuals* and *instantiated properties*, as per Definition 4. For simplicity, the algorithm omits the following subscripts: $I{\sim}I_{ij}^{M}$, $OPI{\sim}OPI_{ij}^{M}$, $DPI{\sim}DPI_{ij}^{M}$.

Step 1: The Ontology Reader acquires classes, object properties and datatype properties from the simulator-specific ontology as specified in lines 1-3. At this point, there are no ontology individuals, and the only existing knowledge pertains to classes and properties.

Step 2: Individuals and their datatype properties are acquired, as shown in lines 4-14. For each class from the simulator-specific ontology (the loop starting with line 6), the Simulation Model Reader acquires the set of all individuals (line 7). Then, for each individual, all datatype properties are obtained, as specified in lines 8-12. After this step, all instances of all classes are known.

Step 3: Object properties are instantiated and the ontology-based simulation model finalized, as described in lines 15-24. Because object properties establish relations between individuals, their instantiation happens after all individuals have been acquired. For each object property (the loop starting with line 16), the Simulation Model Reader acquires all pairs of individuals related by that property (line 17). The union of all pairs of individuals related by object properties makes up a set of instantiated object properties, or OPI. Finally, lines 23 and 24 finalize the ontology-based simulation model. This ontology-based simulation model is written by the Ontology Writer in the ontology language of choice.

Algorithm 6.1: Creating ontology-based simulation models with SIMONTO (*i*-th simulation package)

1:	C_i^S := OntologyReader.getAllClasses(O_i^S)	**Step 1**
2:	OP_i^S := OntologyReader.getAllObjectProperties(O_i^S)	
3:	DP_i^S := OntologyReader.getAllDataProperties(O_i^S)	
4:	I := ∅ //set of all individuals	**Step 2**
5:	DPI := ∅ //set of all instantiated datatype properties	
6:	**for each** class $c_{ij}^S \in C_i^S$	
7:	I_j: = ModelReader. getIndividuals(c_{ij}^S)	
8:	**for each** individual x_{ij}^k , $x_{ij}^k \in I_j$, of the c_{ij}^S class	
9:	// x_{ij}^k - k-th individual of the j-th class	
10:	DPI_{ij}^k: = ModelReader. getDataPropValues(x_{ij}^k)	
11:	DPI := DPI ∪ DP_{ij}^k	
12:	**end for**	
13:	I := I ∪ I_j	
14:	**end for**	
15:	OPI := ∅ //set of all instantiated object properties	**Step 3**
16:	**for each** object property op_{in}^S, $op_{in}^S \in OP_i^S$	
17:	$S(op_{in})$: = ModelReader. getIndividualPairs(op_{in}^S)	
18:	// $S(op_{in})$ - set S of all individuals satisfying op_{in}	
19:	**for each** pair of individuals $(a, b) \in S(op_{in})$ **do**	
20:	OPI := OPI ∪ $OPI_{in}(a, b)$	
21:	**end for**	
22:	**end for**	
23:	PI = OPI ∪ DPI //set of all instantiated properties	
24:	$O_{ij}^M = \{I, PI\}$ //ontology-based simulation model	

Consequently, the Simulation Model Reader is the only SIMONTO Engine component that is simulator-dependent. However, this reader can be replaced with readers from different simulators to represent specific proprietary simulation models in an ontology-based representation. Once a reader has been created for a simulator, it will read all models constructed using that simulator.

It is important to note that Algorithm 6.1 is suitable for parallel processing because it can be divided into parts which can be executed independently on different processing devices. For example, processing for each outer loop in step 2 (lines 7-12) can be executed on different devices as there are no interdependence among loops. Similarly,

each outer loop in step 3 (lines 17-21) can be executed simultaneously. However, step 2 must be completed and its results must be aggregated (line 13) before the step 3 can start.

6.2.3 Storage for Ontology-Based Simulation Models

Disaster-CDM is designed to enable the choice of a storage solution that corresponds to data requirements in terms of data structure as well as access patterns. With respect to simulation models, the task at hand determines data access patterns and consequently influences the choice of storage solution. Therefore, this work addresses the two main storage approaches for simulation models:

- **Storage focused on integration with other data**. This approach was described in Chapter 5 and includes storage of ontology-based simulation models in the document data store alongside all other file-style data. Because it supports full-text search and querying pertaining to a variety of data sources, as will be demonstrated in the case study, this approach is very successful in integrating simulation models with other file-style data**.** Nevertheless, it provides limited capabilities for querying the simulation model itself or for validating that the simulation model complies with rules and constraints.

- **Storage focused on querying within simulation models.** Ontology-based models are represented in OWL, which is characterized by an abstract ontology structure that can be perceived as a graph. On the other hand, graph databases use graph structures with nodes, edges, and properties to represent and store data. They are optimized for efficient management and storage of graph-like data. Consequently, because ontologies can be perceived as graphs, it is apparent that graph databases are a good choice for storing ontologies as well as ontology-based simulation models. Another characteristic that makes a graph database a good choice is its query capabilities; graph database implementations typically offer advanced query capabilities using different query languages. However, this approach imposes challenges in integrating simulation models with other data sources.

Consequently, to facilitate integration services and simulation-specific services, Disaster-CDM stores simulation models twice: in a document data store to facilitate integration with other data sources and in a graph database to enable simulation model querying. In a traditional approach to database design this redundancy is undesired and must be avoided. However, this work adopts a NoSQL approach which allows data redundancy in order to achieve performance and scaling benefits. By storing simulation models in a document data store and in a graph database, Disaster-CDM can take advantage of both, and therefore it can support integration services and simulation-specific services.

6.2.4 Simulation Services

Ontology-based simulation models created by the SIMONTO Engine enable integration, simulation model querying, and rule and constraint validation. These services are external to SIMONTO as they exploit existing methods, approaches, and technologies to carry out simulation-related tasks. However, they act on SIMONTO ontology-based simulation models. The following paragraphs introduce the three categories of services observed in this study:

- **Integration**. This category involves any task that needs to be carried out across a variety of sources, including simulation models. Examples include full-text search and querying over data from a variety of sources. To support integration services, data are stored using integration-focused storage, as described in Section 6.2.3. Specifically, for integration with other file-style data typical in the disaster management domain, Disaster-CDM takes advantage of document data stores.
- **Simulation Model Querying.** Ontologies, and therefore ontology-based simulation models, can be queried using ontology querying languages such as Semantic Query-Enhanced Web Rule Language (SQWRL) [67]. In addition, OWL ontologies can be serialized as RDFs, and therefore they can be queried using RDF query languages such as SPARQL [68]. Queries can be executed directly against OWL ontologies; however, disaster management deals with a large number of simulation models, which makes use of a database preferable to storing ontologies as OWL files. Consequently, to support simulation model querying, Disaster-CDM stores ontology-based simulation models in a graph

database and takes advantage of the querying capabilities provided by the database.

- **Rule and Constraint Validation.** SIMONTO enables validation of model compliance with rules and constraints. Since simulation models are represented as ontologies, validation of simulation models can be performed using ontology approaches. Two approaches for rule validation are considered: genuine rule language and querying.
  - **The genuine rule language approach.** In the genuine rule language approach, the rules represent an addition to the ontology-based model and act upon the ontology. Ontology rule languages express antecedent/consequent relations: if the conditions expressed in the antecedent hold, then the conditions in the consequent also must hold. When the ontology is represented using OWL, a possible choice of rule language is the Semantic Web Rule Language (SWRL) [69]. The main disadvantage of this approach is that the complete ontology must fit into computer memory; therefore, instead of a rule language, this research has used a querying approach for rule and constraint validation.
  - **Querying approach.** This approach to rule and constraint validation involves querying ontologies to identify entities that do not conform to rules or constraints. Although querying is not actually a rule engine, it can identify entities that violate rules. Once the violating entities are identified, corrections are performed on the originating proprietary simulation model, which is also used for simulation execution. The advantage of this approach is that it can be carried out on an ontology-based simulation model stored in a graph database and therefore can take advantage of graph database querying capabilities. Moreover, the ontology-based model does not need to fit into computer memory as is the case with the rule language approach. A querying approach, unlike the rule language approach, cannot take advantage of inferences performed by ontology reasoners. However, in the context of Disaster-CDM, this drawback is outweighed by the advanced querying capabilities provided by graph databases.

This section has introduced the main services that SIMONTO enables in the context of Disaster-CDM; their application is demonstrated in the case study. A querying approach to rule and constraint validation transforms the validation problem into a querying task; however, this study examines them separately due to their different objectives and the presence of an alternative approach to rule and constraint validation. Future work will explore the possibility of combining the advantages of the two rule and constraint validation approaches.

6.3 Summary

This chapter has proposed SIMONTO, an ontology-based representation of simulation models, which represents proprietary simulation models as interconnected instances of simulator-specific ontologies. SIMONTO transforms existing proprietary simulation models into their corresponding ontology-based models with the objective of facilitating integration, simulation model querying, and rule and constraint validation. In the context of Disaster-CDM, SIMONTO is responsible for simulation model processing. For the purpose of integration with other file-style data, ontology-based simulation models are stored in the document data store along with other file-style data sources. For the purpose of simulation model querying and rule and constraint validation, ontology-based simulation models are stored in a graph database to take advantage of the advanced querying capabilities provided by the database and to enable querying within ontology-based simulation models.

Chapter 7

7 Evaluation: Case Study 1

The objective of the presented case studies is to demonstrate Disaster-CDM benefits on data collected during the Disaster Response Network Enabled Platform (DR-NEP) project [70]. Public databases, such as Emergency Events database (http://www.emdat.be) and a number of databases from Global Risk Information Platform (http://www.gripweb.org/gripweb/?q=disaster-database) were considered; however, those databases contain only public information. In contrast, data set from DR-NEP project includes public data as well as sensitive data which are not accessible to the general public.

Because this research focuses on knowledge acquisition and storage, the presented case studies show how knowledge from DR-NEP data set is acquired and stored; the benefits are demonstrated through the knowledge delivery services. Specifically, case study 1 presented in this chapter demonstrates how knowledge is acquired from a variety of file-style data sources including simulation models, how it is stored, and illustrates the Disaster-CDM benefits through the integration knowledge delivery services. In contrast, case study 2 presented in Chapter 8 focuses on simulation models; it shows how knowledge is acquired from simulation models and stored in a graph database, and demonstrates Disaster-CDM benefits through examples of simulation-specific knowledge delivery services.

Section 7.1 describes the DR-NEP data set which is used in both case studies. The Disaster-CDM implementation is presented in Section 7.2 and knowledge acquisition and delivery in Sections 7.3 and 7.4 respectively. Finally, Section 7.5 discusses the findings and concludes case study 1.

7.1 Data Set

This work was evaluated on data collected by Western University during the two-year period of the CANARIE sponsored Disaster Response Network Enabled Platform (DR-

NEP) project [70]. The DR-NEP project combined the expertise of a number of research groups, industries, government agencies, and response teams in multiple geographical locations with the aim of improving the capability to prepare for and respond to large disasters. To achieve this objective, close collaboration among partners was essential, and the case study presented here demonstrates how Disaster-CDM can facilitate this collaboration. Disaster modelling and simulation played a major role in the project, with a special focus on critical infrastructure (CI) simulation.

The participation of Western University in the DR-NEP project involved the investigation of critical infrastructure interdependencies in an incident that happened on its campus. As the event involved various infrastructures, it was simulated using several simulators including EPANET [65] water distribution simulator and the I2Sim [71] interdependency simulator. Different disaster response strategies were explored and compared with decisions made during the event. Western University collected information directly related to the event such as the event reports and timelines, data pertaining to the involved infrastructures and a variety of other data that could help in better understanding and modeling the event.

The data set is heterogeneous and includes data sources such as disaster plans from different institutions, reports of previous incidents and their timelines, minutes of DR-NEP team meetings and various other disaster response meetings, information about different critical infrastructures, risk analysis documents, and information about a number of disaster-related stakeholders. These data sources are owned by various participants who had to collaborate and share the information they own to achieve successful disaster management.

Because the simulation of critical infrastructures was of special interest in the DR-NEP project, the data set includes a number of simulation models that were used to explore interdependencies of critical infrastructures, including EPANET water-distribution models and I2Sim interdependency models.

With respect to format, the data set includes image files in a variety of formats, text and PDF files, and MS Office documents, including Word, Excel, PowerPoint, and Visio. The

simulation model file formats are simulator-specific: I2Sim models are stored in a Simulink-style .mdl file format, while EPANET models are stored in .NET or .INP files.

7.2 Disaster-CDM Implementation

The Web application was implemented to provide access to the Disaster-CDM system using a Web browser. Specifically, this Web application provides access to KaaS, including knowledge acquisition and knowledge delivery services. Moreover, this approach enables users to access Disaster-CDM from anywhere and from a variety of devices. The following sections describe the implementation of the two main Disaster-CDM knowledge acquisition components: data processing services and data storage.

7.2.1 Implementation: Data Processing Services

Disaster-CDM, according to the KaaS approach, provides data processes as services. The framework of the data processing component was implemented using Web services, in which each data processing component was treated as a separate Web service. In this case study Web services were deployed on a local machine; nevertheless, this choice of implementation enables flexible deployment of services in the cloud environment and their composition for the provision of knowledge acquisition services according to the KaaS approach. Specifically, the RESTful (Representational State Transfer) Web service architecture was used.

This work focuses on data stored in a variety of file formats, and therefore case study 1 implements the data processing services required for such data sources. Implementations of most of the generic file-style data processing services mentioned in Section 4.2.2 are available either as open source or commercial products. This case study used open source products, adapted them when needed, and wrapped them as RESTful Web services. The following data processing services for generic file-style data were implemented:

- File metadata separation service used the Apache Tika Toolkit [72] to detect and extract file metadata. Tika supports a large number of formats, including MS Office, PDF, and a variety of image formats.

- Text extraction service for MS Office documents was also performed by Apache Tika; however, Tika is incapable of extracting text from images. Therefore, text extraction service for image files was performed using the Tesseract [73] optical character recognition (OCR) software. Text from images embedded in MS Office files was also extracted using Tesseract OCR.
- Simulation model service applied the SIMONTO approach. SIMONTO was implemented as described in Section 8.1, and additional services required to prepare ontology-based simulation models for tagging were implemented in Java 1.6.
- Tagging service was carried out using the General Architecture for Text Engineering (GATE) tool suite [74,75]. Specifically, an information extraction system called ANNIE (A Nearly-New IE system), which is distributed with GATE, was used. ANNIE offers great flexibility by enabling customization of its components for the information task at hand; however, in this case study, customizations were not performed.

The data processing rules at this stage of the research were predefined, even though extensions are planned which would provide dynamic rule specification. The challenge with such dynamic rules is that they may result in very similar files being processed in different ways, thus resulting in inconsistent system performance.

7.2.2 Implementation: Data Storage

This case study addresses generic file-style data, and accordingly the storage model chosen was the document data store, as presented in Section 5.3. The data model portrayed in Table 5.1 is designed for document data stores and can be realized in any document data store implementation. This case study used the Apache CouchDB document data store [76].

CouchDB is designed for Web applications. It uses JavaScript Object Notation (JSON) to represent documents and HTTP for an API. The primary reasons for choosing CouchDB for this case study were its scalability, high availability, and partition tolerance. Its ability to scale over many commodity servers enables CouchDB to store large amounts of data,

while its high availability ensures system operation even when a region is affected by a disaster and a local data centre fails. Partition tolerance refers to the ability of the system to remain operational in the presence of network partitions, which is especially relevant in disaster-related applications because it can be expected that parts of the network will fail. CouchDB achieves partition tolerance using an asynchronous replication approach. Multiple replicas placed on geographically distant locations have their own copies of data, and in case of network partition, each replica modifies its own copy. At a later time, when network connectivity is restored, the changes are synchronized.

The primary way of querying and reporting on CouchDB documents is through views which use the MapReduce [77] model with JavaScript as a query language. In the MapReduce model, the Map function performs filtering and sorting, while the Reduce function carries out grouping and aggregation operations.

The Apache Lucene library [78] provides full-text search of data stored in CouchDB. In general, Lucene is an open-source, high-performance text search engine library written in Java. It is suitable for almost any application which requires full-text search and has been recognized for its utility in Internet search engines. With respect to Disaster-CDM, Lucene enables ranked searches and field-specific searches such as searching for a specific file name or an author. This case study takes advantage of the CouchDB-Lucene project [79], which integrates Lucene with CouchDB.

7.3 Knowledge Acquisition Services

Western University stored the data collected and produced as part of the DR-NEP project on a server in a dedicated area. It was the responsibility of the individual participants to place data that needed to be shared among participants onto the server. Therefore, this case study uses data from this DR-NEP server as its data source. In the knowledge acquisition stage, these data were processed by data processing services described in Section 7.2.1 and loaded into the Disaster-CDM system, specifically into CouchDB in the cloud environment.

During the knowledge acquisition process, a total of 1129 files were successfully loaded into the Disaster-CDM system in the cloud environment, resulting in the same number of documents in the data store. A number of files failed to load; however, further review revealed that they were in file formats which are outside the scope of this case study, including pub, zip, mat, dll, and exe. Nevertheless, the number of these files was small, and including them in the knowledge acquisition process would not have resulted in a major system improvement.

Table 7.1 shows a number of files of each type loaded into the system together with their size. As expected, there were many MS Word and PDF files. Furthermore, the number of PowerPoint presentation files (pptx) was large, which may be explained by the nature of the DR-NEP project, which was a multidisciplinary project involving a large number of stakeholders in which presentations were often used to transfer knowledge or convey findings. In addition, a large number of .m and .h text files were found, but their significance in knowledge delivery is minor because they are MATLAB and C-language program files. As for simulation data, there were 20 EPANET model files (.net) and 12 MATLAB model files (.mdl).

Table 7.1: Loaded file types

File Type	# of Files	Size (MB)
pdf	247	321.08
m	149	0.5
pptx	104	197.84
h	73	0.49
jpg	64	71.46
docx	60	13.74
txt	54	0.49
png	51	1.5
.	.	.
.	.	.
.	.	.
net	20	1.24
mdl	12	11.42

Updates to existing knowledge are outside the scope of this study. In other words, the knowledge from each file is acquired once, and the system does not keep track of subsequent updates to the file. New files can be loaded into the system at any time. Nevertheless, updates to existing knowledge will be addressed in future work.

7.4 Knowledge Delivery Services

This case study demonstrated knowledge delivery services on two examples of integration services: full-text search and querying. The two are complementary approaches for accessing data stored in a cloud data store, with each one exhibiting strengths for specific data access tasks.

7.4.1 Full-text Search

Storing data in a document data store as described in Section 5.3 enables variants of full-text search. Three variants of full-text search have been observed:

- **Searching attached documents**. This search relies solely on document attachments in the CouchDB data store. Because original files are attached to the CouchDB document in their original form, this search is somewhat similar to using an indexing and search engine, Lucene in this case, directly on the original files. This strategy does not take advantage of any data processing performed during knowledge acquisition and is the baseline for comparison with other strategies.
- **Searching extracted text**. This strategy includes only the contents of *docText* field. Because text extracted from images is in *docImageText* fields, this strategy ignores text contained in images as well as text in images embedded in other documents. Note that ontology-based simulation models are stored in *docText* fields and therefore are included in this strategy.
- **Searching extracted text, including text from images**. This approach takes full advantage of text extraction service described in Section 7.2.1, including Tika text extraction and OCR text extraction, by engaging both fields, *docText* and *docImageText*, in the search strategy. This strategy also takes full advantage of the data processing performed in the knowledge acquisition stage.

A full-text search screen from the implemented Web application is displayed in Figure 7.1. This application enables users to choose among the three described search strategies; on the screen in Figure 7.1, the extracted text strategy is selected. The result of searching for the term “power house” are displayed in the table with two columns: document and last modified. The document column displays the file name, and it can be noted that the search result is made up of various file types, including pdf and text files, MS Word, PowerPoint, and simulation model files. Some of the files appear several times with different last modified date. This is caused by files residing in different folders, but having the same name. Disaster-CDM does not check whether files with the same name have identical content, but rather creates a new document in the data store for each loaded file.

Table 7.2 provides an overview of different full-text search strategies with respect to the main file categories addressed in this study. For the three file categories, PDF, text and I2Sim model files, all three search strategies were virtually the same. Even though searching I2Sim models produced the same results set, the ranking of the documents was different because the searches were based on different text content. The attached document strategy searched mdl files, which are text files, directly, while the other two strategies searched the ontology-based simulation models. Consequently, the attached document strategy ranked simulation models lower than the other two strategies.

With regard to MS Office files, the difference among the various searches depended on whether or not they were using text extracted from images. The data set for this case study contained 82 MS Word files (doc and docx), of which only 8 contained images from which text was successfully extracted. In contrast, out of 140 PowerPoint files (ppt and pptx), only 6 did not benefit from the OCR service. Therefore, the OCR service had a greater impact on processing PowerPoint files than on processing Word files. With respect to image files, out of 116 images, text was successfully extracted from 75; however, some of the extracted text did not contain readable words and therefore was not beneficial for searching. Therefore, the OCR service had a greater impact on PowerPoint files than on image files, which can be explained by the common use of diagram-style graphs in PowerPoint presentations.

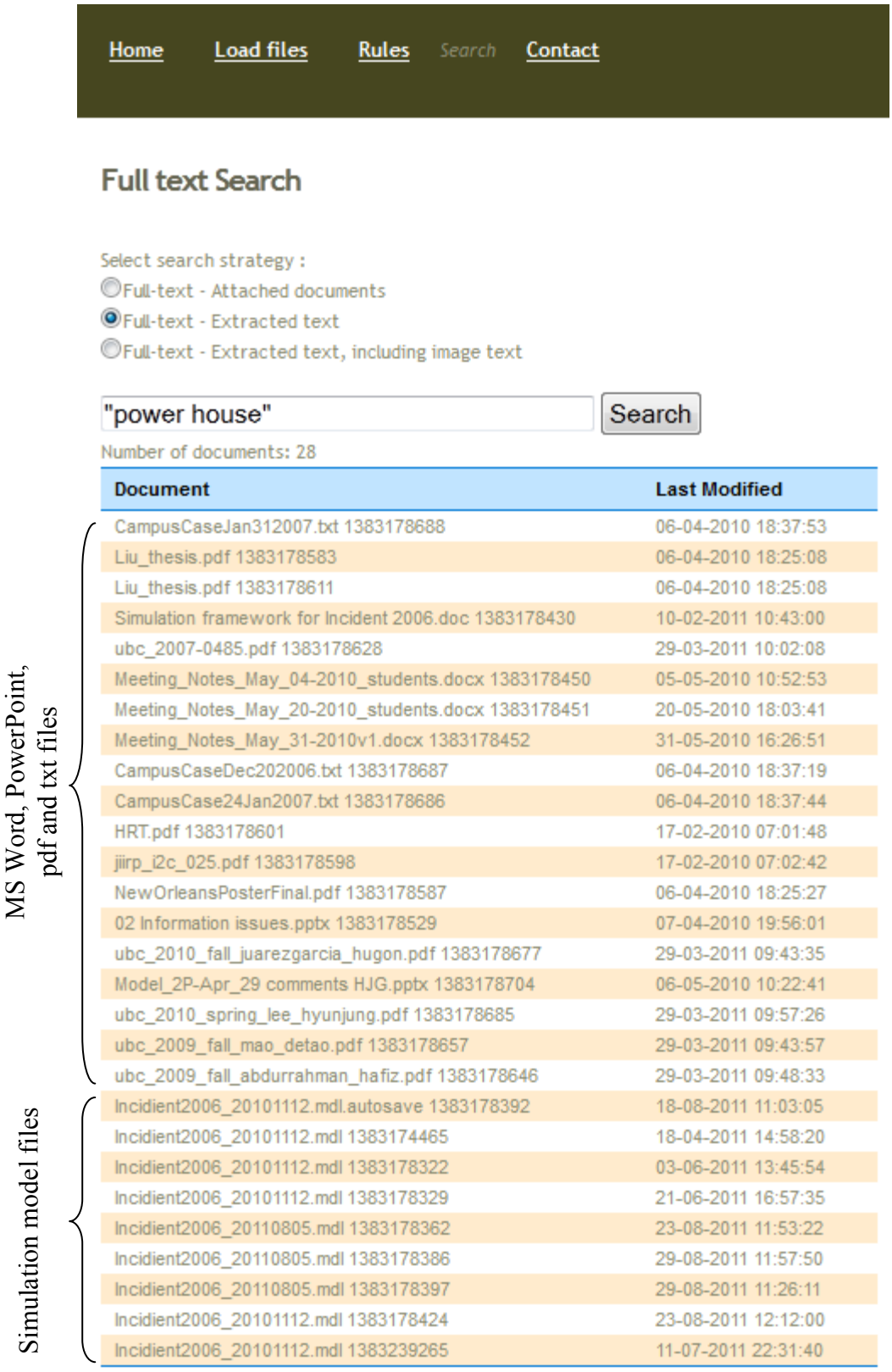

Figure 7.1: Full-text search

Table 7.2: Search strategies

File type	Search Strategy		
	Attached document	**Extracted text**	**Extracted text including text from images**
PDF files	✓	✓	✓
MS Office files	✓ Does not include text from images	✓ Does not include text from images	✓
Image files	×	×	✓
Text files	✓	✓	✓
Simulation model files			
I2Sim model files (.mdl)	✓ (mdl file are text files)	✓	✓
EPANET model files (.net)	×	✓	✓

Transforming simulation models into their corresponding ontology-based representations did not change the result set with respect to I2Sim models, but was essential for including EPANET models in the full-text search. The attached document strategy did not search EPANET models because they are represented in .net binary files; however, the extracted text strategies searched EPANET models by taking advantage of the ontology-based simulation models stored in *docText* fields.

Note that the attached document search strategy took advantage of CouchDB-Lucene [79], which uses Apache Tika [72] to search the attached documents. This case study also used Tika to extract text from files, and therefore the only major difference between the attached-document and the extracted text strategies was with respect to EPANET model files. Only the extracted text strategy included EPANET model files.

Full-text search can also be achieved by applying text search engine such as Lucene directly on the file system containing disaster-related data; however, such search ignores text contained in images as well as text in images embedded in other documents. In contrast, full-text search in Disaster-CDM includes image text because OCR performed in knowledge acquisition stage extracted text from images. Moreover, direct full-text search on the file system does not include EPANET .net model file as they are binary file. Disaster-CDM transforms EPANET model files into ontology-based representation, and consequently includes them in full-text search. Additionally, storing data in NoSQL data

store facilitates querying file-style data and allows Disaster-CDM to take advantage of scaling and replication capabilities provided by NoSQL store.

7.4.2 Querying File-Style Data

The documents contained in the document store are semi-structured: the data within a document are encoded, but each document can have a different structure. Such a data model enables document data stores to index documents based on primary keys as well as on document content fields. Consequently, this data model provides querying abilities.

The data model designed for storage of file-style data, as presented in Table 5.1, was flexible enough to enable storage of diverse data, but at the same time was relatively standardized to support querying abilities. In this case study, querying was used to obtain various kinds of aggregate information about the contents of the data store, such as the number of documents of each type or the number of documents containing images. Aggregate querying is illustrated in this case study on a simple example, that of counting the documents of each type. In CouchDB, this is achieved by views which make use of the MapReduce approach. The Map function extracts the value of the *fileExtension* field from within each document, while the Reduce function groups by *fileExtension* (which is in the *key* argument passed to the Reduce function) and counts the entries for each *fileExtension*.

Map function:

```
function(doc) {
      emit(doc.fileExtension, 1);
}
```

Reduce function:

```
function (key, values) {
      return sum(values);
}
```

The data presented in Table 7.1 were obtained by executing this query. As illustrated, obtaining such information from the Disaster-CDM system is very simple; however,

doing this without the Disaster-CDM system would require extensive manual efforts or use of specialized (custom or off-the-shelf) software.

The full-text search described in the previous section did not take full advantage of the tagging performed during data acquisition. When text was extracted from documents, tagging was performed, and the results were stored within different *tag* fields. Because the tag fields are encoded within the document, they facilitate querying. For example, as part of the DR-NEP project, Western University explored an incident on the university campus which involved a local power plant. During data acquisition, the text extracted from documents was forwarded to the tagging services. If a power plant was mentioned in a document, the ANNIE tagging service used in this case study recognized "power plant" as an organization and therefore tagged it as *organization='power plant'*. Consequently, the resulting document in the data store contained the following entry: `tag: {organization: ["power plant"]}`. This document structure can be used to find all documents referring to power plants. To enable searching by organization tag, a view with the *organization* tag as its first column was created. In CouchDB, this results in indexing on *organization* tag, thus enabling fast data access by *organization* tag. Listing 7.1 illustrates the Map and Reduce functions for this CouchDB view. The Map function outputs the organization tag as the first array element because this is a search criterion. In addition, this view includes *fileName* to identify the original file and *creationDate* to distinguish more recent documents. In this view, the Reduce function eliminates duplicates produced by the Map function. After this view has been created, data can be queried by specifying *organization* tag values in HTTP calls. A few rows of the search results for the organization tag "power plant" are displayed in Table 7.3.

Listing 7.1: Querying for "Power Plant"—Map and Reduce functions for CouchDB view

Map function

```
function(doc) {
  if (doc.tag.Organization && Array.isArray(doc.tag.Organization)) {
    doc.tag.Organization.forEach( function (organizationTag) {
      var creationDate = doc.metaData["dcterms:modified"];
      if (creationDate == null) {
        creationDate = doc.metaData["dcterms:created"]
      }
      emit([organizationTag.toLowerCase(), doc.fileName, creationDate], null);
    });
  }
}
```

Reduce function

```
function (key, values) {
  return null;
}
```

Table 7.3: Query results for "power plant"

Organization tag	File Name	Creation Date
power plant	11_02_17_DR_NEP_Audit.pptx	2011-02-17T15:21:42Z
power plant	11_09_08_DR_NEP_Audit_Final.pptx	2011-09-08T20:36:29Z
power plant	DeltaV-Chillers-a.jpg	2010-07-19T10:49:52Z
power plant	Disaster_phase2_Aug9.xlsx	2011-08-11T20:14:06Z
power plant	DisasterTable_phase2_Aug11_v1.xlsx	2011-08-12T15:48:49Z

In this case study, only automated tagging was used, and therefore tags typically resembled phrases found in text extracted from documents. In this situation, querying as described in the example gave similar results to the full-text search described in the previous Section 7.4.1. However, Disaster-CDM was designed to allow manual tagging by end users in addition to automated tagging. In a manual tagging scenario, the effectiveness of queries similar to the *organization* tag example would be increased.

7.5 Discussion

The case study presented in this chapter has illustrated the use of Disaster-CDM on the data collected during the Disaster Response Network Enabled Platform (DR-NEP) project. In the knowledge acquisition stage, stakeholders share disaster-related data;

specifically, knowledge is acquired from data owned by various stakeholders. In the knowledge delivery stage, the KaaS approach delivers the knowledge as a service to collaboration participants.

The presented case study focused on data formats commonly present in the disaster domain, e.i. file-style data sources, and implemented the services required for knowledge acquisition from such sources. Processed data were stored in a document data store, specifically CouchDB store.

Two knowledge delivery services were explored: full-text search and querying:

- Various full-text search approaches were investigated, which made it possible to analyze the effects of data processing performed during knowledge acquisition on the full-text search results. Overall, the benefits of data processing services vary by file format as well by file content. For example, as expected, the OCR service had a major impact on image file searching; however, experiments showed that searches of PowerPoint files also benefited greatly from this service. Full-text search does not take advantage of automated tagging, and therefore, if knowledge delivery relies only on full-text search, the automated tagging service can be omitted.
- The querying service proved advantageous in obtaining various types of aggregate information about the stored contents. Some of the query tasks explored in this case study, such as searching for a word or a phrase, can also be achieved by full-text search. In these circumstances, full-text search has an advantage over querying because of its simple call interface and the ability to rank documents according to their relevance. However, the querying approach is promising with respect to manual tagging as it provides fast and easy access to tagged data.

Consequently, the two knowledge delivery services explored in this case study, full-text search and querying, are complementary services which are suitable for different tasks. Knowledge delivery services, together with knowledge acquisition services, facilitate collaboration by providing a platform for sharing and integrating disaster-related information.

7.6 Summary

This chapter has presented an evaluation of the proposed Disaster-CDM framework on data collected by Western University during the CANARIE sponsored Disaster Response Network Enabled Platform (DR-NEP) project. The presented case study applied the Disaster-CDM framework on file-style data sources including simulation models. First, the Disaster-CDM implementation was presented, including its two main knowledge acquisition components: data processing services and data storage. Disaster-related knowledge was acquired from the DR-NEP data set using a variety of knowledge acquisition services and stored in a document data store. Finally, the benefits of Disaster-CDM were demonstrated on two knowledge delivery services: full-text search and querying.

Chapter 8

8 Evaluation: Case Study 2

While case study 1 addressed the application of Disaster-CDM on a variety of file-style data including simulation models with the objective of integrating diverse data sources, the case study presented in this chapter focuses on simulation models with the goal of illustrating how Disaster-CDM enables simulation model querying and rule and constraint validation.

Within the Disaster-CDM framework SIMONTO is responsible for processing simulation models and creating their ontology-based representations. All simulation models from the DR-NEP data set were transformed to ontology-based representations and stored. However, to illustrate Disaster-CDM use with simulation models, this chapter focuses on two specific models: the Western University campus water distribution network modelled in EPANET, and the I2Sim model developed as part of the DR-NEP project for the investigation of infrastructure interdependencies.

The SIMONTO implementation, including SIMONTO ontologies and the SIMONTO engine, is described in Section 8.1. The two ontology-based models, EPANET and I2Sim models, created by SIMONTO from the two selected proprietary simulation models are presented in Section 8.2. Knowledge acquisition services and the storage of ontology-based simulation models are included in Section 8.3. Finally, knowledge delivery services are demonstrated in Section 8.4 and discussion is provided in Section 8.5.

8.1 SIMONTO Implementation

The SIMONTO approach is generic, meaning that it is independent of any specific simulation engine; however, its implementation requires the creation of two simulation engine-specific components: a simulator-specific ontology and the Simulation Model Reader. The remaining SIMONTO components are independent of simulation engines or simulation packages.

The SIMONTO implementation consists of two parts: the SIMONTO ontologies and the SIMONTO Engine.

8.1.1 SIMONTO Ontologies

The four SIMONTO ontology components can be described as follows:

- **Upper ontology (top ontology layer)**: This case study used the upper ontology depicted in Figure 6.4. In compliance with Definition 2, the upper ontology contains concepts and properties that are common across all domains. As depicted in Figure 6.4, the concepts include *cell*, *control*, *channel*, *meter*, and *other*. The "other" category serves as a container for entities that cannot be assigned to any of the other four categories and is needed because in Definition 3 of the simulator-specific ontology each class of the simulator-specific ontology must be a subclass of an upper ontology class.
- **Simulator-specific ontologies (second ontology layer):** A simulator-specific ontology is created once for each simulator. Hence, in this case study, simulator-specific ontologies for the two simulators are created: the I2Sim ontology and the EPANET ontology. The main classes of the EPANET ontology and their mapping to the upper ontology are presented in Figure 6.5, while the I2Sim ontology classes with their mapping to the upper ontology are shown in Figure 8.1 [52]. Complying with Definition 3, the EPANET ontology contains classes specific to the EPANET simulator. As required by Definition 3 and illustrated in Figure 6.5, each class of the EPANET ontology is a subclass of the upper ontology class. The properties contained in the EPANET ontology are not sub-properties of the upper ontology. Likewise, in the I2Sim ontology, each I2Sim class is a subclass of the upper ontology, as illustrated in Figure 8.1, but the properties are not defined as sub-properties of the upper ontology.
- **Ontology-based simulation models (third ontology layer)**: This case study explored two proprietary simulation models, one EPANET model and one I2SIm model. Therefore, the SIMONTO engine created two corresponding ontology-based simulation models, which are described in further detail in Section 8.2.

- **Rules**: This component adds rules to the ontology-based models with the objective of increasing representation expressiveness and for validation of rules and constraints. This case study did not take advantage of the rules since the simulation models were expressed in OWL and the querying approach was used for rule and constraint validation.

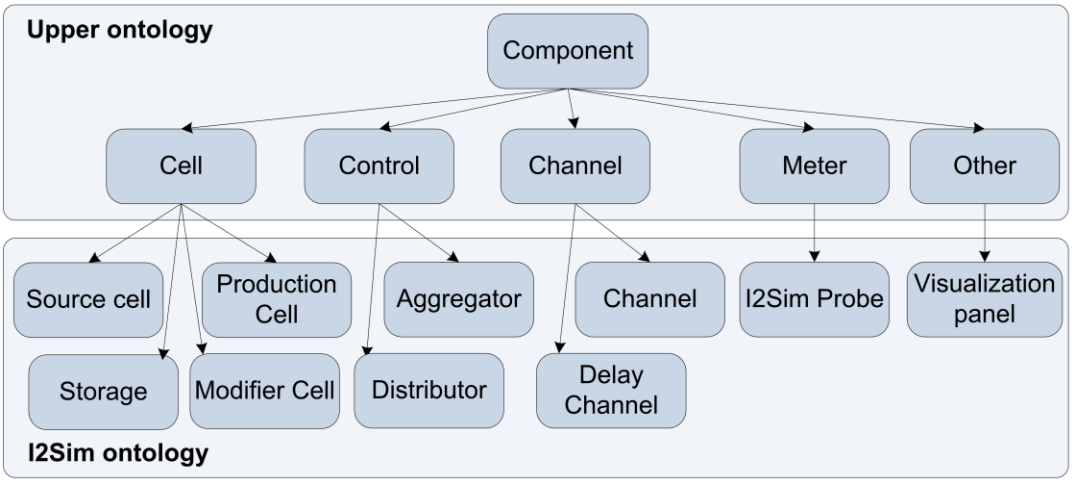

Figure 8.1: I2Sim ontology with relation to the upper ontology

8.1.2 SIMONTO Engine

The simulation models are saved in simulation engine-specific proprietary file formats. EPANET models are saved in .NET and .INP files, while I2Sim models are saved in .mdl files. When working with EPANET, the SIMONTO Engine inputs include the EPANET ontology and the EPANET simulation model as represented in the .NET or .INP file formats. For I2Sim, which is based on MATLAB's Simulink engine, the inputs include the I2Sim ontology and the I2Sim simulation model, which is stored in a Simulink style .mdl file.

In this case study, the SIMONTO Engine was implemented as follows:

- OWL is the representation language of the upper and simulator-specific ontologies and the ontology-based simulation models.
- The Ontology Reader and the Ontology Writer reads and writes OWL ontologies respectively. They are implemented using the Protégé OWL API [80] and Java 1.6.
- The Integrator is implemented using Java 1.6.
- Two Simulation Model Readers are implemented: one each for the EPANET and I2Sim simulators. The EPANET Reader employs the EPANET API to read the simulation model, while the I2Sim Reader uses the Simulink Java library from Technische Universität München [81].

8.2 Ontology-Based Simulation Models

To illustrate the SIMONTO transformation, this case study considers two proprietary simulation models: one EPANET model and one I2Sim model. Consequently, this section portrays the two corresponding ontology-based models. Section 8.3 describes how ontology-based models are loaded into a graph database and Section 8.4 demonstrates the benefits of Disaster-CDM on the two knowledge delivery services: simulation model querying and rule and constraint validation.

8.2.1 The EPANET Model

The observed water distribution network consists of 802 junctions, 836 pipes, 6 valves, and 9 reservoirs. This simulation model has been transformed into an ontology-based representation; Figure 8.2 shows this representation displayed in the Protégé ontology editor [82]. The left pane shows the EPANET classes, such as *pipe*, *pump*, and *valve*. Because the *pipe* class is selected, the middle pane shows all the individual pipes from the EPANET model. In the right pane, the object properties and the datatype properties for the selected pipe, pipe 831, are displayed. The object properties *hasStartNode* and *hasEndNode* indicate that pipe 831 starts from junction 362 and ends at junction 837. In the EPANET ontology, *hasStartNode* and *hasEndNode* are asserted properties because the EPANET model specifies the pipe start and end nodes.

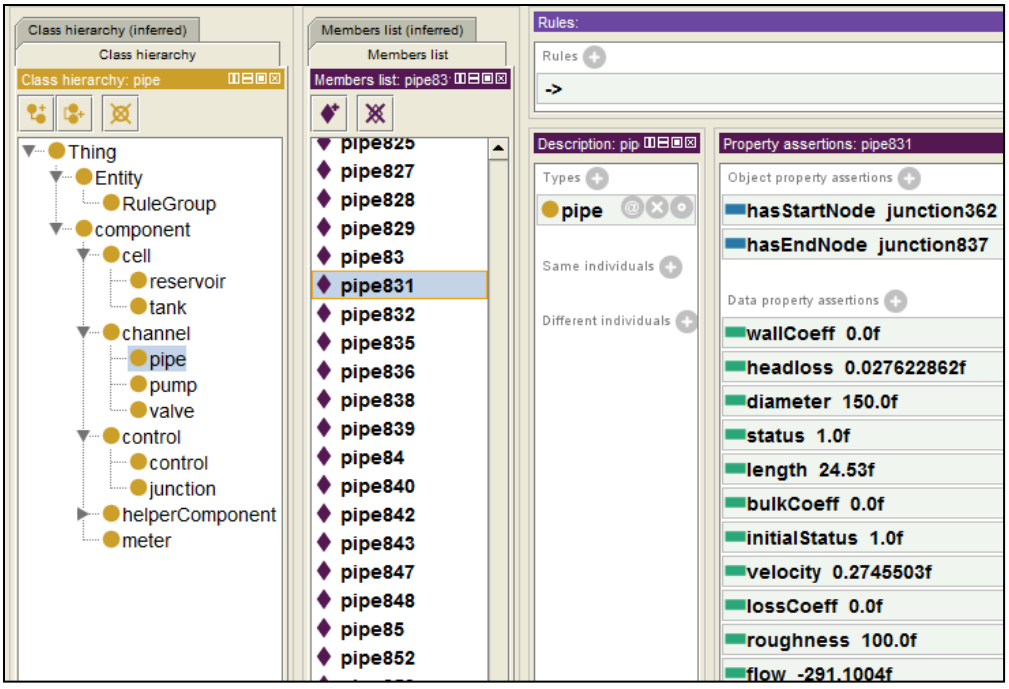

Figure 8.2: Ontology-based representation of the EPANET model

8.2.2 The I2Sim Model

The transformation of the I2Sim model into its ontology-based representation was similar to EPANET model transformation, but a few differences needed to be addressed. I2Sim is built upon Simulink [83] by customizing Simulink blocks and providing entities specific to infrastructure interdependency simulation. Like Simulink [83], I2Sim can divide models into hierarchies of sub-models, as illustrated in Figure 8.3, to make complex system modelling easier. The model hierarchies are represented in the ontology using the *parentSystem* object property. For each child model, the *parentSystem* property links the model to its direct parent. The set of assigned *parentSystem* properties establishes the model hierarchy. A fragment of a hierarchy depicted in Figure 8.3 is represented as *modelE.parentSystem(modelB)* and *modelB.parentSystem(modelA)*. Since the sub-model entities do not belong to any of the simulator-specific ontology classes, a new class, *parentSystem*, was established to contain entities that serve as containers for other entities.

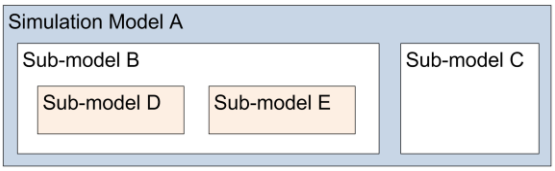

Figure 8.3: Simulation model hierarchy

Initially, it was expected that the I2Sim model would contain only I2Sim blocks. However, when the model was transformed to its ontology-based representation, many entities belonged to the *other* class. Analysis of these entities revealed that they were Simulink blocks. Because I2Sim is constructed based on Simulink by customizing and extending Simulink blocks, it allows Simulink blocks to be used in conjunction with I2Sim blocks. Accordingly, the observed I2Sim model actually contained both I2Sim and Simulink blocks. Therefore, the *non_i2sim* class was created, and the transformation process was allowed to create *non_i2sim* subclasses representing Simulink block categories used in the observed I2Sim model, as illustrated in Figure 8.4.

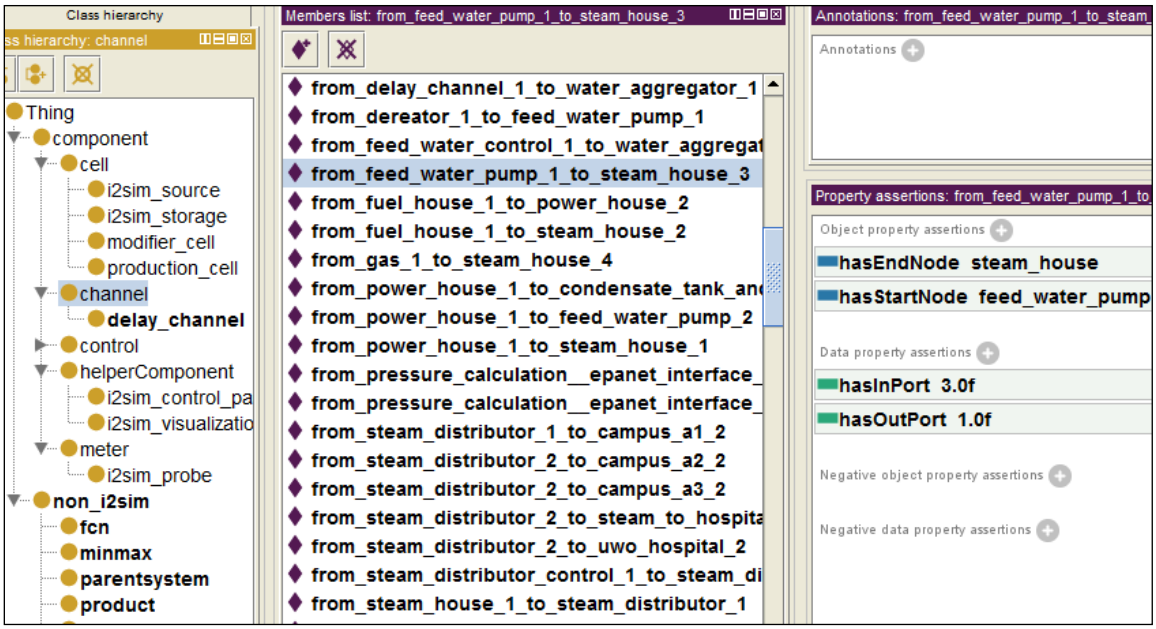

Figure 8.4: Ontology-based representation of the I2Sim model

8.3 Knowledge Acquisition Services

The objective of this case study is to demonstrate how Disaster-CDM facilitates simulation model services, specifically simulation model querying and rule and constraint validation. Those services cannot be achieved in a straightforward manner by the document data store approach. As described in Section 6.2.3, the simulation-specific storage model recognizes that both OWL representations of simulation models and graph databases are graph-based and therefore store ontology-based simulation models in a graph database. Simulation model services can then take advantage of the advanced querying capabilities provided by a graph database. As a result, this storage model supports simulation model querying and rule and constraint validation.

Specifically, this case study uses the Neo4j graph database [84]. Neo4j is an open source graph database implemented in Java with fully ACID transactions and REST as the API interface. It provides powerful and diverse querying capabilities: Neo4j can be queried using Cypher, a property graph query language developed by Neo4j; using Gremlin, a graph traversal language; or even using the RDF query language, SPARQL. Querying examples in this case study are written in the SPARQL query language.

In this case study knowledge acquisition services are responsible for processing simulation model files and storing them in a graph database. Specifically, SIMONTO transforms proprietary simulation models into corresponding ontology-based models, which are then loaded into a graph database, in this case study the Neo4j database. Because Neo4j is a graph database and OWL ontologies are forms of graphs, loading ontologies into the database proved to be straightforward. The loading process was implemented in Java 1.6 using TinkerPop Blueprints [85], a property graph model interface with provided implementations.

The DR-NEP data set contains 20 EPANET models and 12 I2Sim models; however, for the purpose of demonstrating simulation model services, this case study focuses on two models: one EPANET model and one I2Sim model. First, SIMONTO transforms the two simulation models to their corresponding ontology-based representations which are

described in Section 8.2. Next, the two ontology-based models are loaded into the Neo4j database.

Loading the EPANET case study model into the database resulted in a graph with 7,542 vertices and 22,555 edges, while loading the I2Sim model generated a graph with 2,533 vertices and 9,724 edges.

8.4 Knowledge Delivery Services

Knowledge delivery services are illustrated on two examples of simulation-specific services: simulation model querying and rule and constraint validation. Both services operate on simulation-specific storage, specifically on ontology-based models stored in a graph database. Even though the two have different objectives, both use querying approaches to achieve their goals.

8.4.1 Simulation Model Querying

SIMONTO ontology-based simulation models stored in a Neo4j graph database can be queried using different approaches, including SPARQL, Gremlin, and Cypher. This case study illustrates the querying ability on an EPANET model example scenario using the SPARQL query language.

Scenario: A new water distribution network has been modelled in EPANET. To plan network construction, analysts need to find out the total length of all pipes of each diameter in the simulation model. The EPANET simulator cannot directly provide this information.

However, the proposed Disaster-CDM system can provide such information because the ontology-based representation of an EPANET model stored in a graph database can be queried. The following SPARQL query obtains, for each pipe diameter, the number of pipes and their total length. Results are sorted in ascending order of diameter.

```
PREFIX epanet: <http://www.semanticweb.org/ontologies/Simulators/EPANET.owl#>
PREFIX SimModel:
<http://www.semanticweb.org/ontologies/Simulators/EPANETnetwork.owl#>
SELECT ?diameter (COUNT(?pipe) as ?pipeCount) (SUM(?length) as ?pipeLength)
```

```
WHERE { ?pipe  a epanet:pipe.
      ?pipe SimModel:diameter ?diameter.
      ?pipe SimModel:length ?length
}
GROUP BY ?diameter
ORDER BY DESC(?diameter)
```

The results of this query for an ontology-based representation of the EPANET simulation model are displayed in Table 8.1. The first column shows the pipe diameter, the second the number of pipes, and the third the total length of pipes of each diameter.

Table 8.1: SPARQL query output

diameter	pipeCount	pipeLength
600.0	6	2975.27
300.0	32	7072.42
250.0	134	16092.216
200.0	195	11829.779
150.0	270	23219.268
100.0	118	6737.2803
75.0	22	1564.59
62.5	15	997.55005
50.0	25	1259.47
32.5	11	391.91998
25.0	8	398.26

8.4.2 Rule and Constraint Validation

Rule and constraint validation is illustrated on an example from the Ontario Ministry of the Environment document, *Watermain Design Criteria for Future Alterations Authorized under a Drinking Water Works Permit* [86]. Table 8.2 shows a fragment of this document consisting of the Hazen-Williams C-factors that should be used in watermain designs when data from field tests are not available. The Hazen-Williams C-factors specify pipe roughness.

Table 8.2: Watermain design recommendation [86]

Table 1: Hazen-Williams C-Factors	
Diameter – Nominal	**C-Factor**
150 mm	100
200 mm to 250 mm	110
300 mm to 600 mm	120
Over 600 mm	130

The objective of validating rules and constraints, such as those presented in Table 8.2, on an ontology-based simulation model is to identify which entities violate rules and constraints, not to change attribute values. After the entities have been identified, the attribute values should be changed in the original simulation model, in this case the EPANET model, rather than in the ontology-based model because the original simulation model is used for simulation execution. Therefore, querying can achieve rule and constraint validation because it can identify entities that are violating rules without introducing changes to the ontology. This approach transforms rule and constraint validation to a querying problem in which the query itself contains rules or constraints.

The following SPARQL query identifies the entities that violate recommendations in Table 8.2:

```
PREFIX net:
<http://www.semanticweb.org/ontologies/Simulators/EPANETnetwork.owl#>
SELECT *
WHERE {
   {?x net:diameter ?d .
   ?x net:roughness ?r
   FILTER (?d <= 150) FILTER (?r != 100)}
UNION
   {?x net:diameter ?d .
   ?x net:roughness ?r
   FILTER (?d >= 200) FILTER (?d <= 250) FILTER (?r != 110)}
UNION
   {?x net:diameter ?d .
   ?x net:roughness ?r
   FILTER (?d >= 300) FILTER (?d <= 600) FILTER (?r != 120)}
```

```
UNION
   {?x net:diameter ?d .
   ?x net:roughness ?r
   FILTER (?d > 600) FILTER (?r != 130)}
   }
```

The results of this query for an ontology-based representation of the EPANET simulation model are displayed in Table 8.3. Specifically, this SPARQL query identified two pipes violating recommendations: pipe38 and pipe612. The two pipes are modelled with diameter 150 and roughness 110, while the recommendations from Table 8.2 suggest that pipes of diameter 150 should be modelled with roughness 100. Consequently, to comply with recommendations, the two pipes' attributes need to be corrected in the EPANET simulation model. Moreover, the simulation experiments might need to be repeated because the change in the two pipes could impact the simulation results.

Table 8.3: Result of validating rules from Table 8.2

x (pipe)	d (diameter)	r (roughness)
pipe38	150.0	110.0
pipe612	150.0	110.0

8.5 Discussion

The case study 2 focused on simulation models and demonstrated how Disaster-CDM facilitates simulation model querying and rule and constraint validation. For this purpose, proprietary simulation models were first transformed by SIMONTO to their corresponding ontology-based representations and then stored in a graph database.

Simulation-specific knowledge delivery services operate on ontology-based simulation models stored in a graph database and take advantage of the querying capabilities provided by the database. Two knowledge delivery services were demonstrated: simulation model querying and rule and constraint validation.

Simulation model querying was demonstrated using SPARQL, an RDF querying language. However, this case study did not explore other querying languages which

potentially could have advantages over SPARQL. For example, it can be expected that a graph traversal language, such as Gremlin in Neo4j, would show performance benefits in the presence of join operations.

Because the querying approach was chosen in this work for rule and constraint validation, as explained in Section 6.2.4, the presented case study demonstrated validation with SPARQL queries. This is similar to simulation model querying as both deal with query data stored in a graph database, but rule and constraint validation actually expresses rules in the form of queries. The case study presented here demonstrated rule and constraint validation, but did not explore the potential limitations of the approach used. A thorough comparison of the genuine rule language and querying approaches to rule and constraint validation would provide a better insight into the limitations, advantages, and disadvantages of each approach; however, such a comparison is outside the scope of this work.

8.6 Summary

This chapter, like Chapter 7, has presented an evaluation of the proposed Disaster-CDM framework; however, in contrast to Chapter 7 which addressed file-style data sources, this chapter was concerned with simulation models. Because SIMONTO is the Disaster-CDM component responsible for processing simulation models, the SIMONTO implementation and the ontology-based models created by SIMONTO were discussed first. In the presented case study knowledge acquisition service, specifically SIMONTO, transformed simulation models to their corresponding ontology-based representations and stored them in a graph database. Finally, the benefits of Disaster-CDM were demonstrated on two simulation-specific knowledge delivery services: simulation model querying and rule and constraint validation.

Chapter 9

9 Conclusions and Future Work

In recent years, we have witnessed an increase in the number and severity of extreme weather events and natural disasters around the globe. Consequently, disaster impacts on human life and property have risen as well, escalating the importance of minimizing disaster impacts and making an effective response imperative in today's society.

The main goal of disaster management is to minimize disaster impact, and a crucial element for achieving this goal is effective decision-making through all four disaster phases: mitigation, preparedness, response, and recovery. Successful and effective disaster decision-making requires information gathering, sharing, and integration by means of collaboration on a global scale and across governments, industries, communities, and academia. A large quantity of disaster-related data is available, including response plans, records of previous incidents, simulation data, social media data, and Web sites; however, current data management solutions offer few or no integration capabilities and limited potential for collaboration.

At the same time, changes in software and hardware have created opportunities for new solutions in the disaster management domain. In particular, recent advances in cloud computing, Big Data, and NoSQL have opened doors for new solutions in disaster data management.

Consequently, this research proposed a Knowledge as a Service (KaaS) framework for disaster cloud data management (Disaster-CDM). The ultimate goal of Disaster-CDM is to facilitate improved and informed disaster decision-making and consequently to reduce the impact of disasters on human lives and property. Disaster-CDM facilitates information gathering and sharing through knowledge acquisition and delivery; stores large amounts of disaster-related data from diverse sources by taking advantage of cloud computing and NoSQL data stores; and facilitates search and supports interoperability and integration by means of knowledge delivery services.

The case studies presented in this research demonstrated the use of Disaster-CDM on data collected during the Disaster Response Network Enabled Platform (DR-NEP) project. In the first case study knowledge was acquired from diverse file-style data sources such as MS Office documents, images, text and PDF files, and simulation models. In this case study Disaster-CDM contributions were demonstrated on examples of two integration services: full-text search and querying services. The second case study focused on simulation models and illustrated Disaster-CDM benefits on simulation-specific tasks; specifically, two simulation services were presented: simulation model querying and rule and constraint validation.

Section 9.1 discusses the contributions of this research, while Section 9.2 presents future work.

9.1 Contributions

The contributions of this thesis can be summarized as follows:

Disaster-CDM framework

This research has proposed Disaster-CDM, a Knowledge as a Service (KaaS) framework for disaster cloud data management. Disaster-CDM provides a flexible and customizable disaster data management solution which can be expanded and altered according to the needs of the organizations using it. Disaster-CDM achieves the following objectives:

- Information gathering and sharing is facilitated by means of knowledge acquisition and knowledge delivery services. Knowledge acquisition services are responsible for acquiring knowledge from diverse collaboration partners and from heterogeneous data sources, processing it to add structure, and storing it. Knowledge delivery services are responsible for integrating information from different data sources and delivering knowledge to consumers as a service.
- Storing large amounts of disaster-related data from diverse sources is achieved by taking advantage of cloud computing and NoSQL data stores. Specifically, data are stored in a cloud environment in a variety of relational databases and NoSQL data stores. Scalability of cloud and NoSQL solutions makes it possible to start

the system small and expand as needs grow by adding heterogeneous nodes. Within the cloud environment, data stored in NoSQL data stores is replicated, often across large geographic distances. This ensures high availability and system operation in the presence of failures, which in the disaster management domain is particularly important as it can be expected that disasters will cause a variety of failures. NoSQL data stores offer flexible data model and therefore enable storage of diverse disaster-related data. Moreover, Disaster-CDM allows a choice of storage solutions to suit a variety of data structures and access patterns.

- Search, interoperability and integration are supported primarily by means of knowledge delivery services. Data stored in diverse data stores is provided to consumers as services according to the KaaS approach. This work focuses on knowledge acquisition, specifically on data processing services and storage; knowledge delivery services are used to demonstrate the benefits of the proposed framework.

As already mentioned, Disaster-CDM is a flexible and expandable disaster data management solution which can accommodate a variety of data sources. Therefore, this research has defined a process for introducing a new data source into the framework. The process consists of three steps:

- adding new processing services for dealing with the new data source;
- defining data processing rules for the new data source;
- determining appropriate data storage, including choosing the type of data store and designing a data model.

All three steps must be considered when introducing a new data source, but they will not necessary introduce new components. For example, depending on existing processing capabilities, a new data source will not necessarily need a new processing service.

This research applied the proposed Disaster-CDM approach to file-style data because data formats commonly present in the disaster management domain include MS Office files, text and PDF files, images, and simulation model files. The common element among these data sources is that information is typically stored in self-contained and

largely unrelated files. The data processing services required for file-style data were identified, and examples of data processing rules were presented. With respect to storage, two steps were performed: in the first step the type of data store was chosen, specifically, the document data store was selected as it is designed around the concept of a document and provides storage flexibility along with querying capabilities; in the second step a data model for storage of file-style data in a document data store was designed.

Disaster-CDM contributions were demonstrated with two case studies. The first case study illustrated how Disaster-CDM supports integration of diverse file-style data sources on examples of full-text search and querying services. The second case study focused on simulation-specific tasks and demonstrated how Disaster-CDM facilitates querying within simulation models and rule and constraint validation.

SIMONTO

This work has proposed SIMONTO, an ontology-based representation of simulation models, which represents domain simulation models as interconnected instances of simulator-specific ontologies. SIMONTO transforms existing simulation models expressed in simulator-specific model files to their corresponding ontology-based representations. Such ontology-based simulation models facilitate integration with other sources, provide simulation model querying capabilities, and enable rule and constraint validation.

In the context of Disaster-CDM, SIMONTO is responsible for processing simulation models. In this study, the created ontology-based simulation models are stored according to their intended use:

- For integration with other file-style data sources, simulation models are stored in a document database alongside other data. Such storage enables full-text search and querying over data originating from a variety of sources, as demonstrated in Section 7.4.
- For querying within simulation models, and for enabling rule and constraint validation, ontology-based simulation models are stored in a graph database. This

approach takes advantage of the advanced querying abilities provided by graph databases, as demonstrated in Section 8.4.

9.2 Future Work

This study has primarily addressed the knowledge acquisition and data storage components of the proposed framework. Directions for future research related to knowledge acquisition and data storage include:

- **Data acquisition from other sources such as Web sites and social media**: Including other sources of information will provide a more comprehensive knowledge base and, when integrated with existing data, will lead to better decision-making.
- **Dynamic data processing rule specification**: This work has considered static and predefined data processing rules. Dynamic rule specification should be explored for rule flexibility and to simplify addition of new data sources.
- **Changes to existing knowledge (knowledge evolution)**: In this research, knowledge from each data source is acquired once, and the system does not keep track of subsequent updates. Support for knowledge evolution would provide for a better, more comprehensive disaster knowledge solution.
- **Knowledge conflicts:** In disaster management, due to large number of participants and the immense diversity of data sources, it is to be expected that knowledge conflicts will occur. Conflicts must first be detected and then resolved or managed so that non-contradicting knowledge can be provided to consumers.
- **NoSQL data store comparison:** In this study, the document data store model specifically CouchDB, was chosen for storage of file-style data. A detailed comparison of different data store implementations would assist in choosing the most suitable NoSQL implementation for the task at hand.
- **Required storage space:** This work did not analyze the storage space requirements for the proposed approach. Disaster-CDM stores original files in addition to data produced by data processing services. Moreover, the full-text

search and CouchDB views presented in the case study require indexes, which also occupy space and must be included in space estimates.

The role of SIMONTO in Disaster-CDM is the transformation of proprietary simulation models to ontology-based representations which are better suitable for integration and querying. With respect to SIMONTO, directions for future research include:

- **SIMONTO limitations:** This work did not explore the limitations of ontology-based representations of simulation models. The completeness of the created model needs to be explored to understand its limitations more fully.
- **Working with large numbers of simulation models**: The case study presented in Chapter 8 transformed a few simulation models and loaded them into a graph database. The behavior of the system with a large number of models loaded into the database remains to be investigated.
- **Stand-alone use of SIMONTO**: This study employed SIMONTO as part of Disaster-CDM; however, SIMONTO also has the potential of being used on its own because Ontology-based simulation models could also be queried directly using an ontology querying language such as SQWRL. Moreover, rules can be added to the ontology base with the help of an ontology rule language such as SWRL. The use of SIMONTO outside the Disaster-CDM framework requires further exploration.
- **Rule languages and querying languages**. When ontology-based models are loaded into a graph database, they can be queried using different approaches. Moreover, SIMONTO ontology-based models can be queried directly. Exploring the advantages and disadvantages of different approaches would provide a better insight into their capabilities and limitations; therefore, it could lead to guidelines for choosing the appropriate approach for the task at hand.

This research has presented the main design of the knowledge delivery component without addressing details. Consequently, directions of future work in knowledge delivery include:

- **Integration of NoSQL data stores**: Since NoSQL data stores were designed for different purposes, they differ greatly in their data models and querying abilities, which presents an obstacle to integration. A major part of the integration challenge is the fact that NoSQL data stores do not support a standard query language.
- **Data analytic services**: The case study presented in this thesis involved query and full-text services, but analytics services were not addressed. Data analytics actually refers to Big Data analytics, where disaster Big Data are analyzed to find meaningful insights which could lead to better decisions.
- **Privacy and security**: Providing adequate security and privacy for such a framework is challenging for a number of reasons, including cloud storage on third-party premises and in a shared multi-tenant environment, diversity of the storage models involved, and the large number of collaboration participants.

The proposed Disaster-CDM framework is designed for use with disaster-related data; however, it could potentially be applied in other domains. For example, Disaster-CDM for file-style data, as presented in Chapter 5 and demonstrated in Chapter 7, could be applied to any file-type data and is not restricted to disaster-related data. Future work will explore the potential of using the same framework, possibly with some adaptations, in other domains. For example, possible use of the proposed framework for geological data management will be explored.

References

[1] V. Hristidis, S. Chen, T. Li, S. Luis, and Y. Deng, "Survey of Data Management and Analysis in Disaster Situations," *The Journal of Systems and Software*, vol. 83, no. 10, pp. 1701-1714, 2010.

[2] A.Y. Chena, F. Peña-Morab, and Y. Ouyang, "A Collaborative GIS Framework to Support Equipment Distribution for Civil Engineering Disaster Response Operations," *Automation in Construction*, vol. 20, no. 5, pp. 637-648, 2011.

[3] K. de Faria Cordeiro, T. Marino, M. L. M. Campos, and M. Borges, "Use of Linked Data in the Design of Information Infrastructure for Collaborative Emergency Management System," *Proceedings of the 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 764-771, 2011.

[4] D. P. Coppola, Introduction to International Disaster Management, Amsterdam, Netherlands: Butterworth-Heinemann, 2011.

[5] S. Sakr, A. Liu, D.M. Batista, and M. Alomari, "A Survey of Large Scale Data Management Approaches in Cloud Environments," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 311-336, 2011.

[6] D. Kossmann and T. Kraska, "Data Management in the Cloud: Promises, State-of-the-Art, and Open Questions," *Datenbank-Spektrum*, vol. 10, no. 3, pp. 121-129, 2010.

[7] R. Hecht and S. Jablonski, "NoSQL Evaluation: A use Case Oriented Survey," *Proceedings of the International Conference on Cloud and Service Computing*, pp. 336-341, 2011.

[8] R. Abdullah, Z. D. Eri, and A. M. Talib, "A Model of Knowledge Management System for Facilitating Knowledge as a Service (KaaS) in Cloud Computing Environment," *Proceedings of the International Conference on Research and Innovation in Information Systems*, pp. 1-4, 2011.

[9] P. Zadrozny and R. Kodali, Big Data Analytics using Splunk, Berkeley, CA, USA: Apress, 2013.

[10] M. Stonebraker, S. Madden, D. J. Badi, S. Harizopoulos, N. Hachem, and P. Helland, "The End of an Architectural Era: (it's Time for a Complete Rewrite)," *Proceedings of the 33rd International Conference on very Large Data Bases*, pp. 1150-1160, 2007.

[11] M. A. Beyer and D. Laney, "The Importance of 'Big Data': A Definition," http://www.gartner.com/id=2057415, 2007.

[12] F. Ohlhorst, Big Data Analytics: Turning Big Data into Big Money, Hoboken, N.J, USA: Wiley, 2013.

[13] S. Mohanty, M. Jagadeesh, and H. Srivatsa, Big Data Imperatives: Enterprise Big Data Warehouse, BI Implementations and Analytics, Berkeley, CA, USA: Apress, 2013.

[14] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud Computing: State-of-the-Art and Research Challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7-18, 2010.

[15] T. Erl, Z. Mahmood, and R. Puttini, Cloud Computing: Concepts, Technology, & Architecture, Upper Saddle River, NJ, USA: Prentice Hall, 2013.

[16] P. Mell and T. Grance, "The NIST definition of cloud computing. NIST special publication 800-145," http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2011.

[17] S. Khoshnevis and F. Rabeifa, "Toward Knowledge Management as a Service in Cloud-Based Environments," *International Journal of Mechatronics, Electrical and Computer Technology*, vol. 2, no. 4, pp. 88-110, 2012.

[18] J. Han, M. Song, and J. Song, "A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing," *Proceedings of the 10th IEEE/ACIS International Conference on Computer and Information Science*, pp. 351-355, 2011.

[19] P. J. Sadalage and M. Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Upper Saddle River, NJ, USA: Addison-Wesley, 2013.

[20] R. Cattell, "Scalable SQL and NoSQL Data Stores," *ACM SIGMOD Record*, vol. 39, no. 4, pp. 12-27, 2011.

[21] E. Brewer, "CAP Twelve Years Later: How the "Rules" have Changed," *Computer*, vol. 45, no. 2, pp. 23-29, 2012.

[22] O. Curé, R. Hecht, C. Le Duc, and M. Lamolle, "Data Integration Over NoSQL Stores using Access Path Based Mappings," *Proceedings of the 22nd International Conference on Database and Expert Systems Applications*, pp. 481-495, 2011.

[23] L. Palen, K. M. Anderson, G. Mark, J. Martin, D. Sicker, M. Palmer, *et al.*, "A Vision for Technology-Mediated Support for Public Participation & Assistance in Mass Emergencies & Disasters," *Proceedings of the ACM-BCS Visions of Computer Science Conference*, pp. 1-12, 2010.

[24] A. Schram and K. M. Anderson, "MySQL to NoSQL: Data Modeling Challenges in Supporting Scalability," *Proceedings of the 3rd Annual Conference on Systems, Programming, Languages and Applications: Software for Humanity*, pp. 191-202, 2012.

[25] S.H. Othman and G. Beydoun, "Model-Driven Disaster Management," *Information & Management*, vol. 50, no. 5, pp. 218-228, 2013.

[26] T. Silva, V. Wuwongse, and H. N. Sharma, "Linked Data in Disaster Mitigation and Preparedness," *Proceedings of the Third International Conference on Intelligent Networking and Collaborative Systems*, pp. 746-751, 2011.

[27] K. M. Anderson and A. Schram, "Design and Implementation of a Data Analytics Infrastructure in Support of Crisis Informatics Research: NIER Track," *Proceedings of the 33rd International Conference on Software Engineering*, pp. 844-847, 2011.

[28] C. Chou, F. Zahedi, and H. Zhao, "Ontology for Developing Web Sites for Natural Disaster Management: Methodology and Implementation," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 41, no. 1, pp. 50-62, 2011.

[29] I. Lai, S. Tam, and M. Chan, "Knowledge Cloud System for Network Collaboration: A Case Study in Medical Service Industry in China," *Expert Systems with Applications*, vol. 39, no. 15, pp. 12205-12212, 2012.

[30] Y. Qirui, "Kaas-Based Intelligent Service Model in Agricultural Expert System," *Proceedings of the 2nd International Conference on Consumer Electronics, Communications and Networks*, pp. 2678-2680, 2012.

[31] S. Kannimuthu, K. Premalatha, and S. Shankar, "Investigation of High Utility Itemset Mining in Service Oriented Computing: Deployment of Knowledge as a Service in E-Commerce," *Proceedings of the Fourth International Conference on Advanced Computing*, pp. 1-8, 2012.

[32] A. Maria, "Introduction to Modeling and Simulation," *Proceedings of the Winter Simulation Conference*, pp. 7-13, 1997.

[33] L. G. Birta and G. Arbez, Modelling and Simulation: Exploring Dynamic System Behaviour, London: Springer, 2007.

[34] E. Abu-Taieh and A. El Sheikh, "Commercial Simulation Packages: A Comparative Study," *International Journal of Simulation*, vol. 8, no. 2, pp. 66-76, 2007.

[35] "1516-2010 - IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules," *IEEE Standard*, 2010.

[36] A. Tofani, E. Castorinia, P. Palazzaria, A. Usovb, C. Beyelb, E. Romeb, *et al.*, "Using Ontologies for the Federated Simulation of Critical Infrastructures," *Proceedings of the International Conference on Computational Science*, vol. 1, no. 1, pp. 2301-2309, 2010.

[37] D. D. Dudenhoeffer, M. R. Permann, and M. Manic, "CIMS: A Framework for Infrastructure Interdependency Modeling and Analysis," *Proceedings of the Winter Simulation Conference*, pp. 478-485, 2006.

[38] J. A. Miller, G. T. Baramidze, A. P. Sheth, and P. A. Fishwick, "Investigating Ontologies for Simulation Modeling," *Proceedings of the 37th Annual Simulation Symposium*, pp. 55-63, 2004.

[39] G. Guizzardi and G. Wagner, "Towards an Ontological Foundation of Discrete Event Simulation," *Proceedings of the 2010 Winter Simulation Conference*, pp. 652-664, 2010.

[40] W. L. Oberkampf and C. J. Roy, Verification and Validation in Scientific Computing, New York, NJ, USA: Cambridge University Press, 2010.

[41] T. J. Barth, F. Graziani, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, *et al.*, Computational Methods in Transport: Verification and Validation, Berlin, Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2008.

[42] B. Lee, T. Critchlow, G. Abdulla, C. Baldwin, R. Kamimura, and N. Tang, "The Framework for Approximate Queries on Simulation Data," *Information Sciences*, vol. 157, no. 1-2, pp. 3-20, 2003.

[43] C. Szabo and Y. M. Teo , "An Approach to Semantic-Based Model Discovery and Selection," *Proceedings of the 2011 Winter Simulation Conference*, pp. 3054-3066, 2011.

[44] "IEEE Standard for Property Specification Language (PSL)," *IEEE Std 1850-2012*, pp. 1-188, 2012.

[45] S. Staab and R. Studer, Handbook on Ontologies, Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2009.

[46] L. Lacy and W. Gerber, "Potential Modeling and Simulation Applications of the Web Ontology Language - OWL," *Proceedings of the Winter Simulation Conference*, vol. 1, pp. 265-270, 2004.

[47] G. A. Silver, L. W. Lacy, and J. A. Miller, "Ontology Based Representations of Simulation Models Following the Process Interaction World View," *Proceedings of the Winter Simulation Conference*, pp. 1168-1176, 2006.

[48] J. A. Miller and G. Baramidze, "Simulation and the Semantic Web," *Proceedings of the Winter Simulation Conference*, pp. 2371-2377, 2005.

[49] P. Benjamin and K. Akella, "Towards Ontology-Driven Interoperability for Simulation-Based Applications," *Proceedings of the Winter Simulation Conference*, pp. 1375-1386, 2009.

[50] K. Grolinger, E. Mezghani, M. A. M. Capretz, and E. Exposito, "Knowledge as a Service Framework for Disaster Data Management," *Proceedings of the 22nd WETICE Conference*, pp. 313-318, 2013.

[51] C.P. Sumathi, G.Gayathri Devi, and T. Santhanam, "A Survey on various Approaches of Text Extraction in Images," *International Journal of Computer Science and Engineering Survey*, vol. 3, no. 4, pp. 27-42, 2012.

[52] K. Grolinger, M. A. M. Capretz, A. Shypanski, and G. S. Gill, "Federated Critical Infrastructure Simulators: Towards Ontologies for Support of Collaboration," *Proceedings of the Canadian Conference on Electrical and Computer Engineering, Workshop on Connecting Engineering Applications and Disaster Management*, pp. 1503-1506, 2011.

[53] K. Grolinger, M. A. M. Capretz, J. R. Marti, and K. D. Srivastava, "Ontology–based Representation of Simulation Models," *Proceedings of the 24the International Conference on Software Engineering and Knowledge Engineering*, pp. 432-437, 2012.

[54] M. Wang, B. Ni, X. Hua, and T. Chua, "Assistive Tagging: A Survey of Multimedia Tagging with Human-Computer Joint Exploration," *ACM Computing Surveys*, vol. 44, no. 4, pp. 1-24, 2012.

[55] Public Safety Canada, "The Canadian Disaster Database," http://www.publicsafety.gc.ca/cnt/rsrcs/cndn-dsstr-dtbs/index-eng.aspx, 2013.

[56] Center for research on the epidemiology of disasters - CRED, " EM-DAT, The International Disaster Database," http://www.emdat.be/database,.

[57] P. Atzeni, F. Bugiotti, and L. Rossi, "Uniform Access to NoSQL Systems," *Information Systems*, 2013.

[58] J. A. Bondy and U. S. R. Murty, Graph Theory, New York: Springer, 2008.

[59] T. Tran, D.M. Herzig, and G. Ladwig, "SemSearchPro – using Semantics Throughout the Search Process," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 4, pp. 349-364, 2011.

[60] R. Delbru, S. Campinas, and G. Tummarello, "Searching Web Data: An Entity Retrieval and High-Performance Indexing Model," *Web Semantics: Science, Services and Agents on the World Wide Web*, pp. 33-58, 2011.

[61] V. Kashyap, P. Bernstein, C. Bussler, M. J. Carey, S. Ceri, U. Dayal, *et al.*, The Semantic Web, Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2008.

[62] W3C OWL Working Group, "OWL 2 Web Ontology Language," http://www.w3.org/TR/owl2-overview/, 2009.

[63] G. Klyne and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax," http://www.w3.org/TR/rdf-concepts/, 2004.

[64] D. Brickley and R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," http://www.w3.org/TR/rdf-schema/, 2004.

[65] "EPANET, Water Distribution Modeling," http://www.epa.gov/nrmrl/wswrd/dw/epanet.html, 2008.

[66] T. Berners-Lee, "WWW past & future," http://www.w3.org/2003/Talks/0922-rsoc-tbl/, 2003.

[67] M.J. O'Connor and A. Das, "SQWRL: A Query Language for OWL," *OWL Experiences and Directions, 6th International Workshop*, 2009.

[68] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," http://www.w3.org/TR/rdf-sparql-query/, 2008.

[69] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, "SWRL: A Semantic Web Rule Language," http://www.w3.org/Submission/SWRL/, 2004.

[70] "DR-NEP (Disaster Response Network Enabled Platform) project," http://drnep.ece.ubc.ca/index.html, 2011.

[71] H. A. Rahman, M. Armstrong, D. Mao, and J. R. Marti, "I2Sim: A Matrix-Partition Based Framework for Critical Infrastructure Interdependencies Simulation," *Proceedings of the Electrical Power and Energy Conference*, pp. 1-8, 2008.

[72] The Apache Foundation, "Apache Tika toolkit," http://tika.apache.org/, 2013.

[73] R. Smith, "An Overview of the Tesseract OCR Engine," *Proceeding of the Ninth International Conference on Document Analysis and Recognition*, vol. 2, pp. 629-633, 2007.

[74] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham, "Evolving GATE to Meet New Challenges in Language Engineering," *Natural Language Engineering*, vol. 10, no. 3-4, pp. 349-373, 2004.

[75] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, et al, "Developing Language Processing Components with GATE. University of Sheffield Department of Computer Science," http://gate.ac.uk/sale/tao/split.html, 2013.

[76] J. C. Anderson, J. Lehnardt, and N. Slater, CouchDB: The Definitive Guide, Sebastopol, CA, USA: O'Reilly Media, 2010.

[77] K. Hwang and K. Hwang, Distributed and Cloud Computing: From Parallel Processing to the Internet of Things, Waltham, MA, USA: Elsevier/Morgan Kaufmann, 2012.

[78] M. McCandless, E. Hatcher, and O. Gospodnetic, Lucene in Action, Stamford, CT, USA: Manning Publications, 2010.

[79] "CouchDB-Lucene project," https://github.com/rnewson/couchdb-lucene, 2012.

[80] "Protégé OWL API," http://protege.stanford.edu/plugins/owl/api/, 2011.

[81] "Simulink Library, Technische Universität München," http://conqat.in.tum.de/index.php/Simulink_Library, 2011.

[82] Stanford Center for Biomedical Informatics Research (BMIR), "Protégé," http://protege.stanford.edu., 2011.

[83] "Simulink - Simulation and Model-Based Design," http://www.mathworks.com/products/simulink/, 2011.

[84] "Neo4j," http://www.neo4j.org/, 2013.

[85] "TinkerPop Blueprints," http://www.tinkerpop.com/, 2013.

[86] Ministry of the Environment Safe Drinking Water Branch, "Watermain Design Criteria for Future Alterations Authorized Under a Drinking Water Works Permit," http://www.ene.gov.on.ca/stdprodconsume/groups/lr/@ene/@resources/documents/resource/std01_086800.pdf, 2012.

Curriculum Vitae

Name: **Katarina Grolinger**

Post-secondary Education and Degrees:

Doctorate, Software Engineering
Western University
London, Ontario, Canada
2013

Western Certificate in University Teaching and Learning
Western University
London, Ontario, Canada
2012

Master of Engineering, Software Engineering
Western University
London, Ontario, Canada
2009

Master of Science, Mechanical Engineering
University of Zagreb
Zagreb, Croatia
1994.

Bachelor of Science**,** Mechanical Engineering
University of Zagreb
Zagreb, Croatia
1997

Honours and Awards:

The Natural Sciences and Engineering Research Council of Canada,
Alexander Graham Bell Canada Graduate Scholarships – Doctoral NSERC CGS-D
2010 - 2013

Ontario Graduate Scholarship (OGS)
2013

Graduate Student Teaching Award
Western University
2012

Graduate Thesis Research Award

Western University
2012

Department Travel Grant
Western University
2011, 2012, 2013

Best presentation in Software Engineering
ECE Graduate Symposium, Western University
2012

Related Work Experience

Teaching and Research Assistant
Western University
London, Canada
2010 – 2013

Teaching Assistant Mentor (Engineering)
Teaching Support Centre, Western University,
London, Canada
2012 - 2013

Instructor: Software Engineering Summer Academy
Western University
London, Canada
Summer 2011 and 2012

Database Administrator – Consultant
Utilismart Corporation,
London, Canada
Jun 2011

Software Engineer - Database Administrator
Mutual Concept Computer Group Inc.,
London, Ontario
2008-2009

Conversion Team Leader - Database Administrator
Mutual Concept Computer Group Inc.,
London, Ontario
2005-2008

Software Developer - Database Administrator
Mutual Concept Computer Group Inc.,
London, Ontario
1999-2005

Software Developer
Online Business Systems
Winnipeg, Manitoba
1997-1999

Teaching and Research Assistant
University of Zagreb,
Zagreb, Croatia
1995-1997

PUBLICATIONS:

REFEREED JOURNALS:

1. **K. Grolinger,** Wilson A. Higashino, Abhinav Tiwari, Miriam A.M. Capretz, Data Management in Cloud Environments: NoSQL and NewSQL Data Stores, Journal of Cloud Computing: Advances, Systems and Application, Springer Open, Vol. 2, doi:10.1186/2192-113X-2-22, 2013.
2. **K. Grolinger,** E. Mezghani**,** M.A.M. Capretz, E. Exposito, **C**ollaborative Knowledge as a Service Applied to the Disaster Management Domain, International Journal of Cloud Computing, 2013 1st round of review.
3. **K. Grolinger**, Miriam A. M. Capretz, Americo Cunha, Said Tazi, Integration of Business Process Modeling and Web Services: A Survey, Service Oriented Computing and Applications, Springer, pp. 1-24, 2013.
4. B. Muslimi, **K. Grolinger**, M.A.M. Capretz, Mark Benko, EEF-CAS: An Effort Estimation Framework with Customizable Attribute Selection, International Journal of Advanced Computer Technology, Vol. 5. No. 13, 2013.
5. **K. Grolinger**, M.A.M Capretz, A Unit Test Approach for Database Schema Evolution, Information and Software Technology, Elsevier, Vol. 53, Issue 2, pp.159-170, 2011.
6. B. Jerbic, **K. Grolinger**, B. Vranjes, Autonomous Agent Based on Reinforcement Learning and Adaptive Shadowed Network. Artificial Intelligence in Engineering. Vol. 13, Issue 2, pp. 141-157, 1999.
7. B. Jerbic, **K. Grolinger**, B. Vranjes, Autonomous Robotic Task Reasoning in Unpredictable Assembly Conditions, Automatika, Vol. 37 (1-2), Zagreb, Croatia, pp. 37-45. 1996.

REFEREED CONFERENCES:

1. **K. Grolinger,** E. Mezghani**,** M.A.M. Capretz, E. Exposito, Knowledge as a Service Framework for Disaster Data Management, The 22nd IEEE WETICE conference, Hammamet, Tunisia, pp. 313-318, 2013.
2. **K. Grolinger,** M.A.M. Capretz, Ontology–based Representation of Simulation Models, The Twenty-Fourth International Conference on Software Engineering

and Knowledge Engineering, San Francisco Bay, California, USA, pp. 432-437 2012.

3. **K. Grolinger,** M.A.M. Capretz, Autonomic Database Management: State of the Art and Future Trends, 27th International Conference on Computers and Their Applications (CATA), Las Vegas, Nevada, USA, pp.276-, 281, 2012.
4. **K. Grolinger,** K.P. Brown, M.A.M. Capretz**,** From Glossaries to Ontologies: Disaster Management Domain, The Twenty-Third International Conference on Software Engineering and Knowledge Engineering, Miami Beach, Florida, USA, pp. 402-407, 2011.
5. **K. Grolinger**, A. Shypanski, G.S. Gill, M.A.M. Capretz, Federated Critical Infrastructure Simulators: Towards Ontologies for Support Of Collaboration, Workshop on Connecting Engineering Applications and Disaster Management 2011, held in conjunction with the IEEE Canadian Conference on Electrical and Computer Engineering, Niagara Falls, Ontario, Canada, pp. 1503 – 1506, 2011.
6. K.P. Brown, **K. Grolinger**, M.A.M. Capretz, Data Providing Web Service-Based Integration Framework for use in a Health Care Context, Symposium on Biomedical and Health Informatics of the IEEE Canadian Conference on Electrical and Computer Engineering 2011, Niagara Falls, Ontario, Canada, pp. 1069–1072, 2011.
7. **K. Grolinger**, B Jerbic, B. Vranjes. Autonomous Robot Behavior Based on Neural Networks, Proceedings of Applications and Science of Artificial Neural Networks III, Orlando, Florida, USA, SPIE Press, pp. 2038-2046, 1997.
8. B. Jerbic, **K Grolinger**, B. Vranjes, Simulation of Intelligent Robot Behavior Based on Reinforcement Learning and Neural Network Approach, 11th International Conference on Artificial Intelligence in Engineering, Southampton, NY, USA, pp. 450-465, 1996.
9. B. Jerbic, **K. Grolinger**, B. Vranjes, Simulation of Robotic Learning in Assembly Process, Proceedings of 7th International DAAAM Symposium, Vienna, Austria, pp. 185-187, 1996
10. B. Vranjes, **K. Grolinger**, B. Jerbic, Modified Fuzzy ART Neural Network in Group Technologies, Proceedings of 7th International DAAAM Symposium, Vienna, Austria, pp. 185-187, 1996.
11. B. Jerbic, **K. Grolinger**, B. Vranjes, Autonomous Robotic Task Reasoning in Unpredictable Assembly Conditions, 13th Conference BIAM 96, Zagreb, Croatia, pp. B1-B6, 1996
12. B. Vranjes, **K. Grolinger**, B. Jerbic, Cellular Manufacturing Formation with Modified Fuzzy ART Neural Network, 13th Conference BIAM 96, Zagreb, Croatia, pp. J5-J8, 1996.