Agent-based analysis and mitigation of failure for cyber-physical systems

Jing Lin

AGENT-BASED ANALYSIS AND MITIGATION OF FAILURE FOR CYBER-PHYSICAL SYSTEMS

by

JING LIN

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

2011

ABSTRACT

Techniques exist for assessment, modeling, and simulation of physical and cyber infrastructures, respectively; but such isolated analysis is incapable of fully capturing the interdependencies that occur when they intertwine to create a cyber-physical system (CPS). The first contribution of this doctoral research includes qualitative representation of the operation of a CPS in a single multi-agent model. Dependable operation of a CPS is contingent upon correct interpretation of data describing the state of the system. To this end, we propose agent-based semantic interpretation services that extract useful information from raw sensor data. We utilize the summary schemas model to reconcile differences in data resolution, syntax, and semantics; and to facilitate imprecise query of databases that maintain historical information, including failure mitigation techniques.

Another contribution of the research is in developing ontologies that enable automated reasoning in the classification and mitigation of failures in CPS operation. As a measure of dependability, we quantify the effectiveness of our proposed ontology-based approach in identifying correct mitigation techniques. Our methodology and models are applicable to a broad range of CPSs; however, they are described in the context of intelligent water distribution networks (WDNs), which are cyber-physical critical infrastructure systems responsible for reliable delivery of potable water. We illustrate the use of game theory in agent-based decision support for allocation of water. As a precursor to empirical validation with field data, we developed an integrated cyber-physical WDN simulator using EPANET and MATLAB, and illustrate the use of this simulator in validating our agent-based model and ontology-based approach to automated mitigation of failure.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude for the academic guidance from my advisor Dr. Sahra Sedigh; and to the late Dr. Ann Miller, who served as my co-advisor until her passing in December 2010. Each of them served as example for me in professionalism and helped me as an international Ph.D. student during the past 3.5 years. The research and life experience in Rolla will become a fortune to me and will help me to overcome obstacles in research and work in the near future.

I would also like to thank Dr. Choi, Dr. Hurson, Dr. McMillin, and Dr. Zawodniok for serving on my committee. The knowledge and technical skills I gained from their courses have greatly contributed to my research work, and their insightful technical feedback helped to improve the quality of my dissertation.

I am grateful to all of my Pine Building colleagues, past and present; and all of the friends whom I met in Rolla. Their company made a challenging time go by faster, and their cordial friendship broadened my perspectives towards the world.

Finally, I would like to extend my deepest appreciation to my parents - my father Youyi and my mother Xinzhong. What they have given to me is the greatest and most selfless love in this world. By their own behavior, they taught me how to be a person of integrity and how to be a good parent. My husband Fei helped me to make an easier transition to life in this country, and always supported my studies. Without the consistent support and care of my family, I would not have been able to complete my studies in such a smooth fashion.

The research presented in this dissertation was supported by grants from the National Science Foundation, the Missouri S&T Intelligent Systems Center, and the United States Department of Transportation.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1 INTRODUCTION

Cyber-physical systems (CPSs) are the integration of computation, as manifested by embedded computers and communication networks, with physical processes [1], [2]. In CPSs, sensors collect information about the physical operation of the system, and communicate this information in real time to the computers and embedded systems used for intelligent control. These cyber components use computational intelligence to process the information and determine appropriate control settings for physical components of the system, such as devices used to control the flow of a physical commodity, e.g., water or electric power, on a line.

A fundamental challenge in research related to CPSs is modeling of these systems. Accurate representation of a CPS encompasses three aspects: computing, communication, and the physical infrastructure. Fundamental differences exist between the attributes of cyber and physical components, significantly complicating representation of their behavior with a single comprehensive model (or simulation tool) [3]. Specialized models and simulation tools exist for the engineering domains represented in critical infrastructure; including power, water, and transportation [4]. These models and tools have been created with the objective of accurately reflecting the operation of the physical system, at high spatial and temporal resolution. Intelligent control is not captured, leaving these models incapable of representing CPSs.

Ideally, a single model would encompass both the physical and cyber system semantics of a CPS in a meaningful way, such that the effects of a specific event are reflected in the reaction of either a cyber or a physical component of the system. Interdependencies among the cyber and physical components, in operation and failure, present a major challenge, as they invalidate simplified models that assume components fail independently [5]. This hampers study of the reliability of CPSs - an urgent task, given the increasing use of cyber control in critical infrastructures.

The research presented in this dissertation rises to the challenge of developing models and simulation techniques that capture cyber-physical interdependencies, while accurately reflecting the operation and attributes of the cyber and physical infrastructures. More specifically, the goal is to develop techniques for qualitative and quantitative characterization of the effect of introducing "intelligence" to physical infrastructure systems, in terms of reliability.

Agent-based modeling is the foundation of this doctoral research, which began with qualitative representation of the operation of a CPS, as a precursor to quantitative modeling. An *agent* is defined as an independent entity capable of making decisions based on information from its environment [6]. Agents can bridge the gap between the cyber and physical layers of a CPS, by serving as stewards for information representing attributes of both layers. An agent-based model can represent diverse characteristics and behaviors at high resolution, a feature essential to capturing the intricacies of a CPS [7].

In this dissertation, a CPS has been modeled as a multi-agent system, where each agent is an independent entity that manages resources within its local scope. The CPS used as a case study is an intelligent water distribution network (WDN), where physical components; e.g., valves, pipes, and reservoirs, are coupled with the hardware and software that support intelligent water allocation. Figure 1.1 depicts a sample WDN.

In brief, the goal of this doctoral research is to model, analyze, and mitigate failures in CPSs, with water distribution selected as the application domain. Based on investigation of studies on hydraulics, as well as knowledge of potential computing failures, we have identified and categorized potential failures and corresponding mitigation techniques for a WDN, as depicted in Figs. 1.2 and 1.3, respectively.

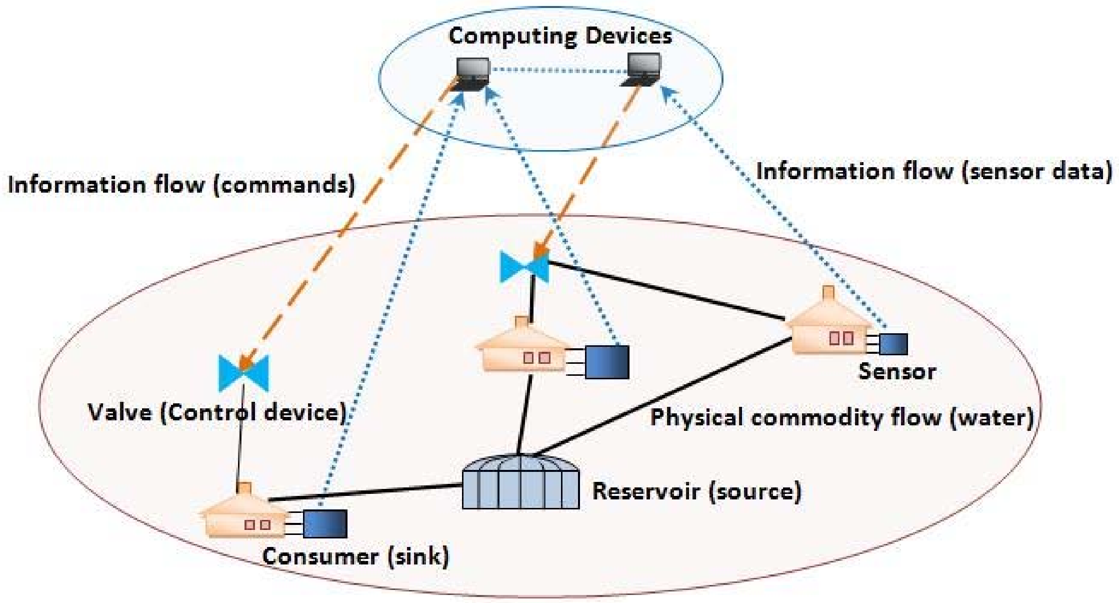

Computing Devices
Information flow (commands)
Information flow (sensor data)
Sensor
Valve (Control device)
Physical commodity flow (water)
Reservoir (source)
Consumer (sink)

Figure 1.1: Cyber and physical components of a WDN.

This dissertation describes the following original research contributions (listed in chronological order), which have been published in seven refereed conference publications, two book chapters, and two journal papers.

1. Development of an integrated CPS simulator for WDNs, using EPANET [8] and MATLAB to represent the physical infrastructure and the decision support algorithms used to control the allocation of water, respectively [9].

2. Development of a qualitative multi-agent model for WDNs, which represents both physical and cyber infrastructures of the CPS [10].

3. Refinement of the multi-agent WDN model to represent semantic interpretation of raw data [11, 12].

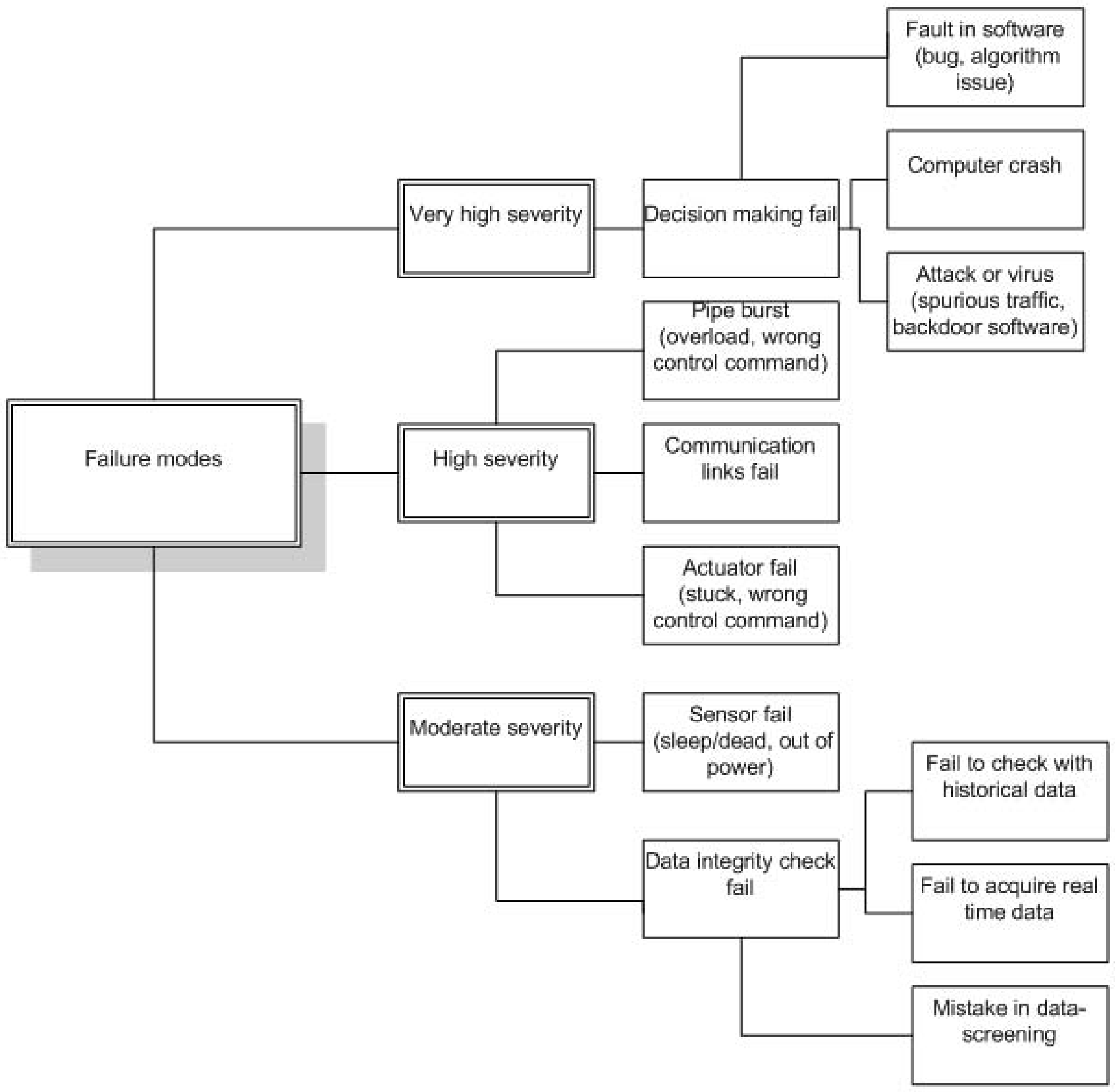

Failure modes
Very high severity
Decision making fail
Fault in software (bug, algorithm issue)
Computer crash
Attack or virus (spurious traffic, backdoor software)
High severity
Pipe burst (overload, wrong control command)
Communication links fail
Actuator fail (stuck, wrong control command)
Moderate severity
Sensor fail (sleep/dead, out of power)
Data integrity check fail
Fail to check with historical data
Fail to acquire real time data
Mistake in data-screening

Figure 1.2: Failure modes of a WDN.

4. Development of a Markov chain model for WDNs, and use of this model in quantitative analysis of non-functional system properties; e.g., reliability, mean time to failure [13].

5. Enhancement of the qualitative model to address semantic heterogeneity and facilitate imprecise query of data sources [14].

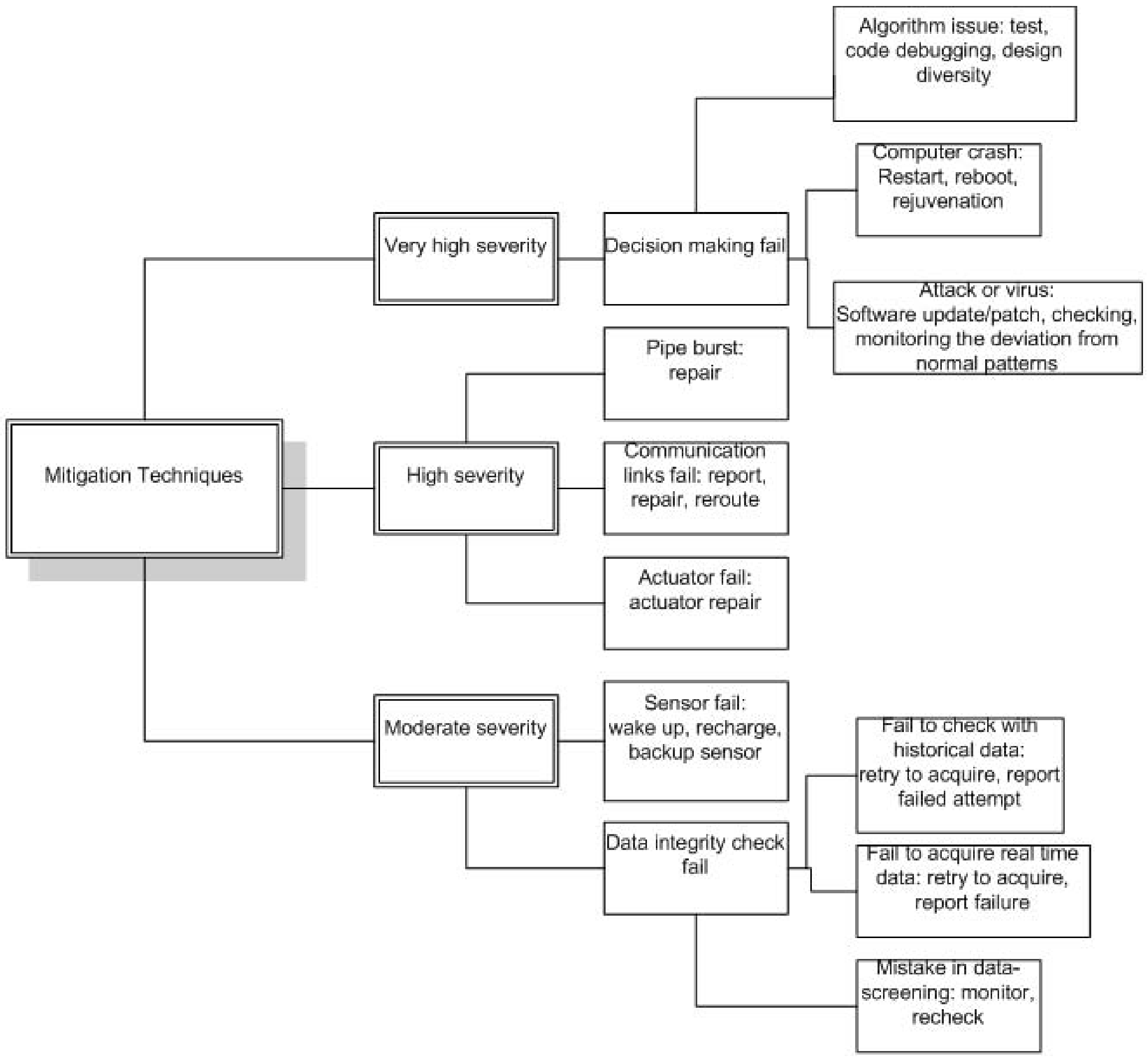

Mitigation Techniques
Very high severity
High severity
Moderate severity
Decision making fail
Algorithm issue: test, code debugging, design diversity
Computer crash: Restart, reboot, rejuvenation
Attack or virus: Software update/patch, checking, monitoring the deviation from normal patterns
Pipe burst: repair
Communication links fail: report, repair, reroute
Actuator fail: actuator repair
Sensor fail: wake up, recharge, backup sensor
Data integrity check fail
Fail to check with historical data: retry to acquire, report failed attempt
Fail to acquire real time data: retry to acquire, report failure
Mistake in data-screening: monitor, recheck

Figure 1.3: Failure mitigation techniques for a WDN.

6. Investigation and simulation of game theory as an algorithmic tool for decision support in WDNs [15], [16].

7. Definition of an ontology to facilitate automated decision support for WDNs [17], [18].

8. Investigation of the efficacy of failure mitigation using the defined ontology [19].

The remainder of this dissertation is organized as follows. Section 2 presents a review of related literature. Section 3 describes our approach to agent-based modeling of WDNs and presents quantitative analysis and simulation techniques for these systems. Ontologies that reflect various aspects of the semantic relationships among components in a WDN are presented in Section 4, where we also present the use of these ontologies in automated failure mitigation. As an illustration of the utility of our proposed technique, Section 5 describes and validates an agent-based environmental decision support system that employs game theory to guide water allocation. We conclude with Section 6, which proposes avenues for future extensions to this research.

2 BACKGROUND AND RELATED WORK

A significant problem in the study of dependability in CPSs in general, and critical infrastructure systems in particular, is characterizing the interdependencies between their cyber and physical components. System complexity has been cited as the main challenge [20]. Other challenges include the low probability of occurrence of critical events, differences in time scales associated with various events, and the difficulty of gathering the data needed for accurate modeling.

A number of modeling and simulation techniques for critical infrastructure are enumerated in [4]. Among these techniques, agent-based models are of particular interest to this doctoral research, as they are capable of capturing component interactions in an accurate, yet simple, fashion. The need for agent-based modeling of distributed complex systems has been investigated in [7]. The availability and reliability of agent-based systems - both significant concerns - are discussed in [21]. The Unified Modeling Language (UML), as a formal specification language with precise semantics, has been adopted to model agent-based systems, as demonstrated in detail in [22]. *In this doctoral research, we have utilized UML 2.0 to construct an agent-based model that qualitatively captures static and dynamic aspects of a CPS.*

Existing modeling techniques for CPSs rely upon semantics to represent the relationship between the cyber and physical components of a CPS. The majority of existing models for CPSs are qualitative in nature. One of very few existing quantitative models is presented in [23], where the Markov imbedded systems technique is used to model reliability of a smart (power) grid, based on knowledge of cascading failures in the system. The existence of data about these cascading failures allows the study to begin from a more advanced stage than was possible for the research described in this dissertation. However, our work shares the goal of characterizing and predicting the operation and failure of a CPS, based on understanding the domain-specific semantics of the system.

The reliability of WDNs, from a purely physical point of view, has long been a topic of interest to the civil engineering community, and is critical to semantic understanding of the physical side of the CPS in focus for this dissertation. Salient studies include work on inoperability modeling [24], which we have used as a basis for analyzing the reliability of the physical infrastructure. *Our research encompasses both cyber and physical aspects of WDNs, and supplements the probabilistic models developed for the physical layer with quantitative data gathered by the cyber components.*

From the engineering implementation perspective, semantic agent technologies are typically closely associated with sensor networks, and several prototype systems and software architectures have been proposed based on the combination of the two. A prototype for battlefield information systems has been described in [25], where the stated goal is to dynamically integrate sensor networks with information fusion processes to support real-time sensing, interpretation, and decision-making in a tactical environment. In [26], an architecture and programming model has been presented for a semantic service-oriented sensor information platform. In contrast to [26], our work expands the semantic service model to a semantic agent framework, whereas [26] focuses on how to use the semantic model to query the system for high-level events without processing raw sensed signals. The use of autonomous semantic agents in developing a new software architecture for distributed processing environments has been proposed in [27]. The discussion in [27] involves software architecture in general, and utilizes semantic web technologies; whereas our work is tailored to the specific requirements of CPSs.

The complexity of CPSs, as well as the necessity of capturing embedded computing and communication capabilities motivate the use of distributed agents and semantic services for representing the relationship between the cyber and physical infrastructures. In our work, the distributed semantic agent model represents the augmentation of data acquisition by sensors in the CPS with decision-making intelligence. *To our knowledge, our work is the first application of semantic agents to modeling of CPSs.*

Several challenges to the development of a generic framework for the design, modeling, and simulation of CPSs are articulated in [28]. Features described as desirable for such a framework include the integration of existing simulation tools, reusability of software, and graphical representation of the modeling and simulation environment. *Our proposed integrated simulator meets these criteria.*

The study most closely related to our proposed simulation method is [29], where a method is proposed for integration of the ns-2 network simulator with the Modelica framework, a modeling language for large-scale physical systems. The study highlights the challenge of two-way synchronization of the simulators. The key difference between this study and our work is that *we link to a specialized simulator capable of accurately representing the operation of the physical infrastructure - in this case a WDN, at high resolution.*

In a WDN, heterogeneous sensor networks and related databases create a multidatabase platform that provides data to the semantic services, which in turn provide information to decision support algorithms. One of the goals of multidatabase platforms is to provide transparent and uniform access to heterogeneous data sources [30]. The summary schemas model (SSM) has been designed to fulfill this objective at low cost [30, 31, 32], while preserving local autonomy and offering scalability. SSM enables automatic identification of semantically similar/dissimilar data that have different/same names and representations. This identification is carried out very efficiently, to the point where in some instances, in spite of their higher complexity, imprecise queries can be carried out faster than precise queries [31]. *Among the many benefits* *of SSM, its support for imprecise queries is of greatest relevance to our work, as it relaxes constraints on the form and vocabulary of database queries.*

Experts differ on their definitions of “ontology,” but every definition we have encountered concurs that an ontology is a representation of entities and the relationships among them [33]. A definition given in [34] characterizes an ontology as the specification of conceptualizations that are used to help computers and humans share knowledge. The semantic web and social network research communities have been especially prolific in their use of ontologies [35], [36]. Two well-known examples are Friend of a Friend (FOAF) [37] and Flink [38], which have been used to analyze social networks, discover communities of practice [39], and explore “hot” topics [40]. Recent applications of ontologies in automated reasoning include their use in improving situational awareness [41]. *We adopt ontologies to reflect various aspects of the semantic relationships among components in the WDN and apply automated reasoning on these ontologies to classify and mitigate failures.*

3 MODELING, ANALYSIS, AND SIMULATION OF WDNS

This section lays the groundwork for the remainder of the dissertation by presenting qualitative modeling and quantitative analysis of WDNs. We propose a multi-agent qualitative model, based on knowledge of the composition and functionality of a WDN. This model was validated using behavior-based simulation, and served as a basis for subsequent quantitative analysis of WDN reliability. The extension of the initial qualitative model includes semantic interpretation of sensor data in the WDN, and the use of SSM to reconcile semantic heterogeneity and facilitate imprecise query of data sources. We also developed an integrated WDN simulator capable of reflecting both cyber and physical aspects and utilized this simulator in validation of our models and methods. In the interest of readability, an overview of the qualitative and quantitative models and a simple simulation are presented here; details are deferred to Appendices A through C.

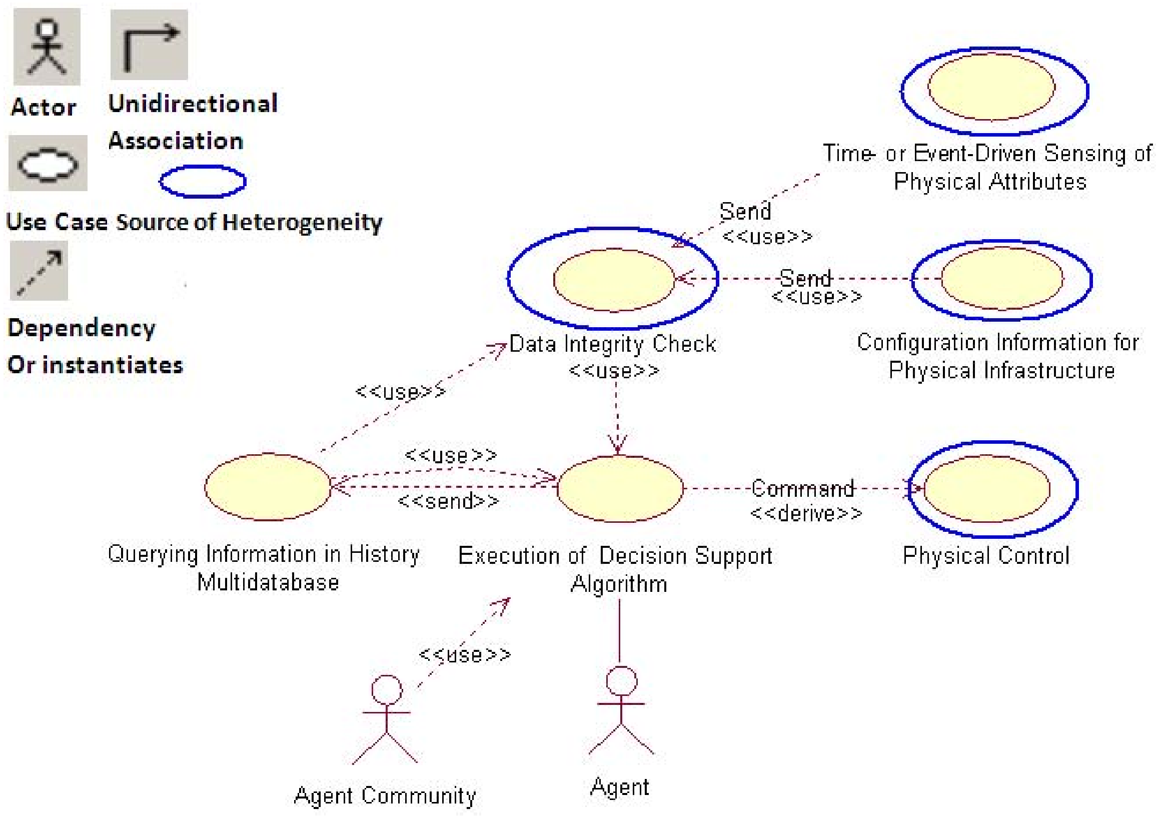
The qualitative analysis was published in the Proceedings of the 5th IEEE Workshop on Engineering Semantic Agent Systems (ESAS) [11]; and as a book chapter [12]. An expanded version of the quantitative analysis will appear in the International Journal of Performability Engineering [13]. The integrated cyber-physical simulator was published in the Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC '09) [9]. The application of SSM was published in the Proceedings of the 20th IEEE International Heterogeneity in Computing Workshop (HCW) [14].

3.1 QUALITATIVE MODELING OF A WDN

For any system, qualitative modeling is a necessary precursor to quantitative analysis. In this subsection, we present a qualitative model for a typical WDN, with the goal of capturing the interaction between the cyber and physical infrastructures. We use UML to represent this model, due to the precise semantics offered by this formal specification language.

Our work aims to accurately model a CPS as a multi-agent system, where each *agent* is an independent entity that manages resources within its local scope. The fundamental feature of an agent is its capability of making independent decisions, which defines agents as active, rather than purely passive entities [42]. Typically, an agent has the attributes of discreteness, autonomy, speed, repeatability, intelligence, flexibility, and situation-awareness [42, 43]. An agent is capable of perceiving its environment, acting on that perception, and interacting with other agents. Agents are diverse, heterogenous, and dynamic in their behavioral rules and attributes. Behavioral rules for agents vary in granularity, sophistication, information load for decision making, and the extent of memory of past events retained by the agent for its future decision making. As a result, the agent-based paradigm is very well-suited to representation of complex heterogenous systems. In a WDN, the agent-based approach offers a dependable, distributed method for managing water resource allocation, enforcing system rules, and responding to unexpected events.

Creating a use case diagram is the first step for qualitative system analysis. A *use case* captures the interaction of a number of external actors with the system towards accomplishment of a goal, which in our case is the provision of potable water. Figure 3.1 shows an actor and use cases involved in a WDN, where each use case represents one functionality in the model. The blue circles highlight the sources of heterogeneity in the WDN, which are of particular relevance to the discussion of Section 3.4, where we elaborate on the use of SSM for reconciling this heterogeneity. This use case diagram can be readily generalized to other CPSs whose main goal is management of a physical commodity. Examples include smart power grids and intelligent transportation systems.



Actor
Unidirectional Association
Use Case
Source of Heterogeneity
Dependency Or instantiates
Time- or Event-Driven Sensing of Physical Attributes
Send <<use>>
Send <<use>>
Data Integrity Check
Configuration Information for Physical Infrastructure
<<use>>
<<use>>
<<use>>
<<send>>
Command <<derive>>
Querying Information in History Multidatabase
Execution of Decision Support Algorithm
Physical Control
<<use>>
Agent Community
Agent

Figure 3.1: Use case diagram for a WDN.

The CPS agent is the actor in the use case diagram, and associated with the decision support algorithm. The agent accesses the system on behalf of an entity and queries the various data sources available, e.g., sensor networks and databases with historical records. For simplicity, only use cases associated with one agent are shown in Fig. 3.1; all other agents have similar use cases associated with them. As shown in Fig. 3.1, sensors collect information about the physical operation of the system on a time- or event-triggered basis. As sensors collect data from different areas, the events may occur sporadically, and the data may be represented in different formats, significant heterogeneity is likely to exist in the data. Heterogeneous data with differences in format and scale is collected by sensors and sent for *Data Integrity Check*, which is a stage of intelligent semantic inference.

The *Data Integrity Check* use case utilizes three main data streams to identify corrupt or invalid sensor data; specifically, i) real-time data from nearby sensors for the same or related physical attributes, ii) information about the physical infrastructure, and iii) data from a (multi)database that maintains historical sensor data. The second and third data streams mentioned are used for corroboration of the first data stream, by checking for discrepancies in the values, whether in variation or in conformance to physical (hydraulic) laws that govern the operation of the physical infrastructure of the WDN. If no data is available from nearby sensors, as would be the case if all nearby sensors are in sleep mode, the history multidatabase will serve as the only source of data for corroboration.

The semantic interpretation service is incorporated in the *Data Integrity Check* use case to organize the information into a meaningful hierarchy. The SSM provides the ability to perform imprecise queries on the aforementioned data sources, by facilitating the identification of semantically similar/dissimilar data. In concert, use of the semantic interpretation service and the SSM while checking data integrity provides transparent and uniform access to heterogeneous data sources. The SSM maintains a hierarchical (logical) meta-data structure based on access terms imported from various local databases, and can be implemented using existing multidatabase technologies, without requiring update or reconfiguration of the local databases. This feature is critical in WDNs, where modifying legacy databases is often infeasible. In this fashion, local autonomy is preserved, while supporting scalability. This approach is very well-suited to large WDNs, which are composed of multiple autonomous districts, each of which can potentially have a different local configuration.

After the integrity of the sensor data is confirmed, it is passed on to the *Decision Support Algorithm*, which utilizes data from two additional sources, each of which can exhibit significant heterogeneity. One of these data sources is the history multidatabase, which maintains information about past results of decision support, e.g., valve settings. Information transmitted from other agents serves as a second data source. For most cases, the other agents are the neighboring agents whose geographical locations are close to the local agent. The *Decision Support Algorithm* is typically implemented in distributed fashion, and its goal is to facilitate intelligent management of physical commodities - in this case, water. The algorithm can make use of legitimate (corroborated) data whose integrity has been checked, and can also resort to the history multidatabase for adjustment (rectification) of the calculated values in determining an appropriate strategy for resource allocation. Meanwhile, the local agent interacts and negotiates with other agents by sharing real-time information that provides a global perspective of resources in the system, and adjusts its own strategy accordingly.

Construction of a state diagram is the next step in qualitative analysis of the WDN, as it describes dynamic operation of the system. For simplicity, at this stage, we consider only the quantity water and do not represent actions related to controlling its quality (chemical composition). Figure 3.2 depicts the state transition diagram of a WDN during one data processing period, which begins when the data is collected and ends when intelligent control has been exerted on the water flow. The condition that can trigger entry to or exit from a particular state has been specified on the corresponding arc.

In Fig. 3.1, we have highlighted the states where significant heterogeneity is likely to be encountered. The *Data Integrity Check* is the most critical state for processing data from heterogeneous sources, and for using the capabilities of the SSM to achieve interoperability. When applying the SSM atop the semantic interpretation, the heterogeneous data sources are reconciled by extracting the essence of the data semantics and aggregating similar data into a more compact semantic entity for further processing. The resulting interoperability among autonomous areas in the WDN

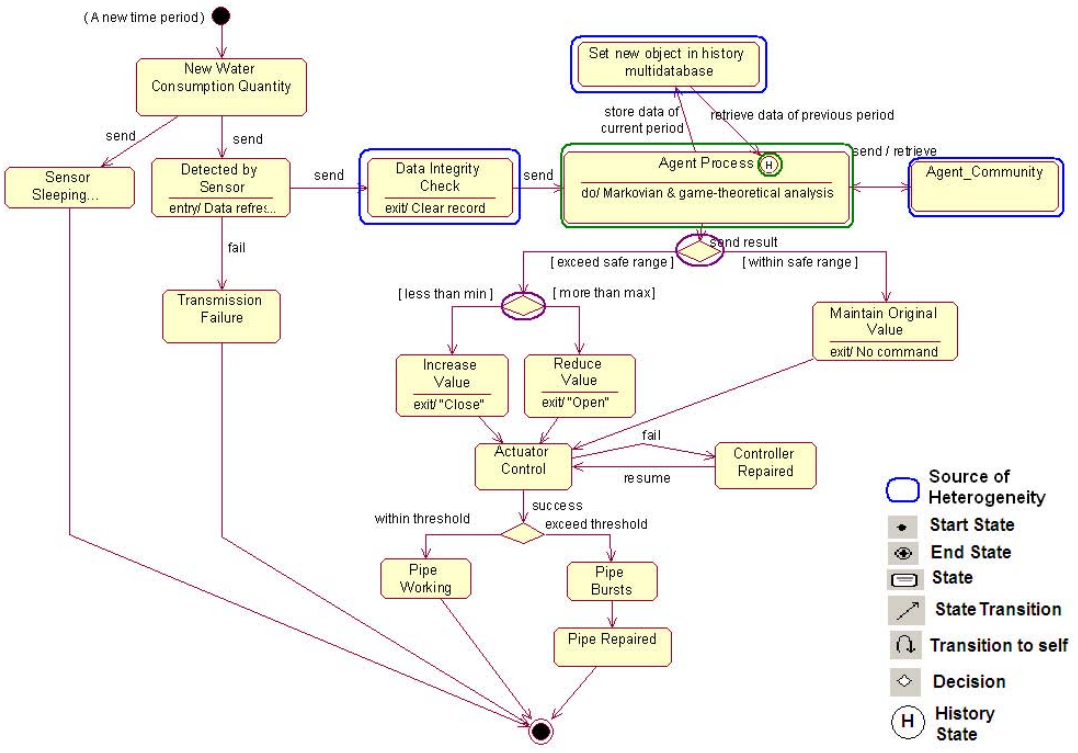
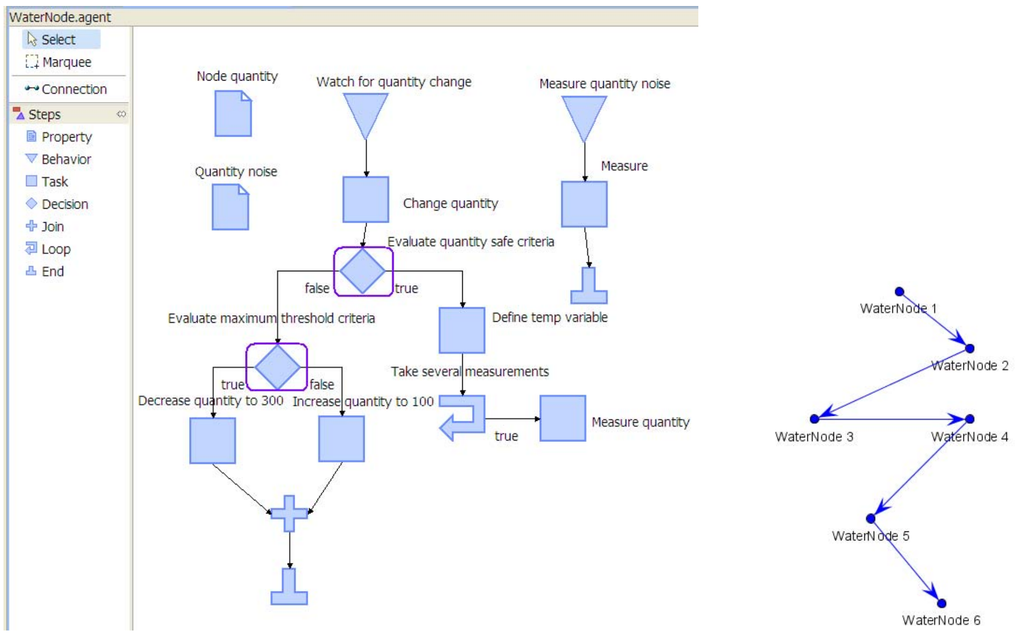


Figure 3.2: State transition diagram for a WDN.

presents the decision support algorithm with transparent and unified access to more compact information.

In Fig. 3.2, the encircled "H" denotes the point where the agent process is making a decision based on a combination of historical and current data (collected by the sensors and checked for integrity). The collection of sensor data is disabled while multidatabase data is being retrieved; afterward, both the newly-collected sensor data and the retrieved multidatabase data are used by the agent in an effort to improve the efficacy of the decision made. Countermeasures have been abstracted as transition states for evaluation of the reliability of the system after remedial actions.

Repast was used for validation of the qualitative WDN model, due to its ability to factor agents, relationships, and behaviors into separate components [44]. In Fig.3.3, Fig.3.3(a) depicts how the qualitative model is translated into component behavior definitions by Repast. Figure 3.3(b) depicts a very simple WDN, where six nodes are connected in a linear topology. The size of each node reflects the quantity of water at the node - initially set to 100 gallons per minute (gpm). An agent is associated with each node. The associated behavior is “Watch for quantity change”, which represents the negotiation underlying a change in a node’s water quantity, which can occur in reaction to changes in the water quantity of other nodes.



(a) Flow chart of the water node agent. (b) Node deployment.

Figure 3.3: System behavior and topology assumed for Repast validation.

As a simple example, we increased the quantity of the first node to 300 gpm, representing an injection of water into this node. This change should incrementally propagate to other nodes. Figure 3.4 depicts the WDN after the increase in water has propagated to three downstream nodes, as reflected by the increased size of the nodes. Figure 3.5 illustrates the gradual propagation of this change to all six nodes of the WDN.

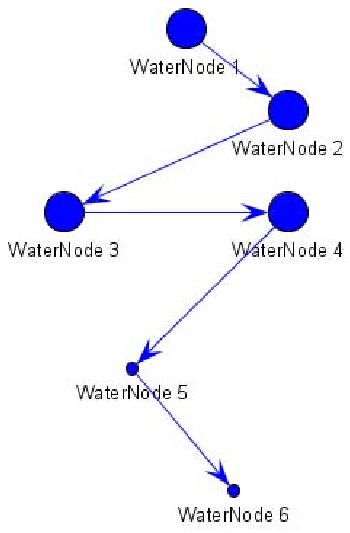


Figure 3.4: Node deployment during simulation.

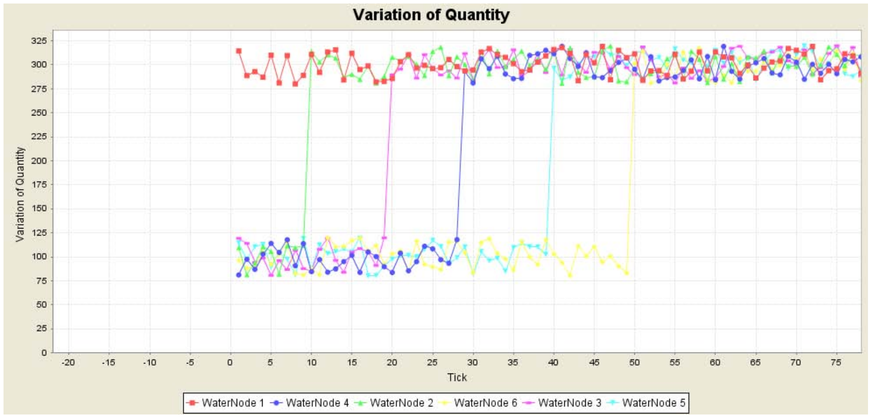


Figure 3.5: Reaction to increase in water quantity of Node 1.

To reflect the intelligent decision-making process of the agent, we add two decision blocks to the flow chart of 3.3(a). The first decision block evaluates whether the current quantity is within the safe range (50-400 gpm in this example). If the quantity is outside the safe range, then the decision flow goes to the second decision block, which prevents this unsafe change from propagating to other nodes. If the quantity is too low, the agent will limit the change propagated, i.e., will raise the quantity of water at other nodes (for example to the default value of 100 gpm) to mitigate the damage caused by the unsafe decrease. This can be accomplished by injecting water from a conduit to the node immediately downstream. This remedial action is shown in Fig. 3.6. Similar remedial action can be taken for a quantity that is dangerously high, as shown in Fig. 3.7.

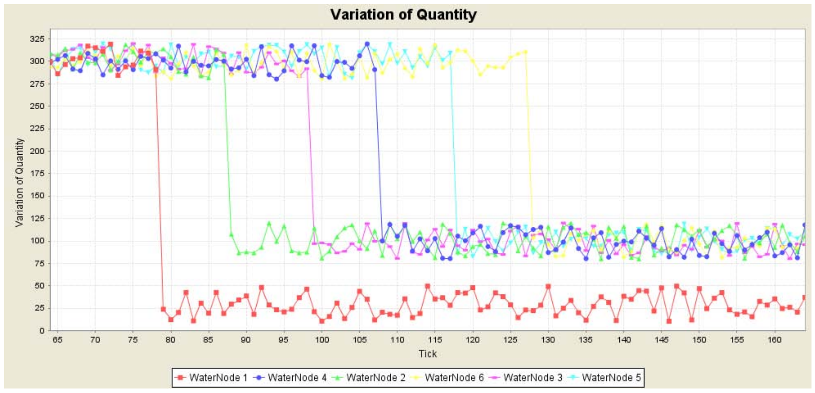


Figure 3.6: Isolating the unsafe decrease in water quantity of Node 1.

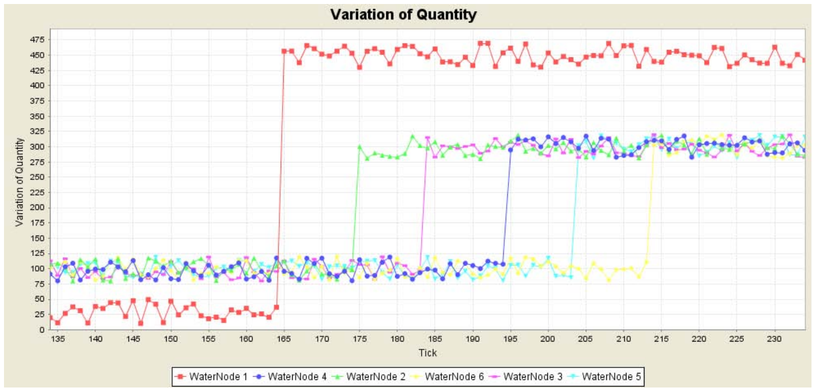


Figure 3.7: Isolating the unsafe increase in water quantity of Node 1.

3.2 QUANTITATIVE ANALYSIS OF A WDN

The qualitative model described in Section 3.1 served as the basis for quantitative reliability analysis. We used the Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE) [45] to develop and simulate a Markov chain model for the WDN. The time-dependent functions describing each component's behavior are restricted by SHARPE to be exponential-polynomial in form.

Figure 3.2 serves as the basis for the Markov chain model shown in Fig. 3.8, where each state is labeled with the functionality of the system in that state, and each transition arc is labeled with the corresponding transition probability. In the absence of field data, these values were estimated based on semantics of the operation of the WDN, as deduced from hydraulics literature. Population of the model with field data is a future task. In Fig. 3.8, the rectangles highlight the states that represent failure of certain components, and the circle highlights the "decision_making" state, which is the state of greatest interest from the CPS perspective. The transition probabilities leaving each state should sum to one; however, SHARPE does not represent self-transitions, hence, the probabilities associated with self-transitions do not appear in the figure.

We applied the severity specification in FMEA [46], to categorize the five types of failures marked by the rectangles in Fig. 3.8 into three groups:

1. The most critical failure is the "decision_fail", as the failure of this component can lead to malfunction of the overall system.

2. The second-level failures are the "pipe_bursts" and/or the "actuator_fail", as both pipes and actuators have repair mechanisms and hence can recover from the failure states.

3. The third-level failures are the "sensor_fail" and/or the "check_fail". The data integrity check is likely to identify faulty sensor data. At this point, we are assuming that the checking mechanism is relatively reliable, as compared to other components of the system. This assumption can be relaxed in the future.

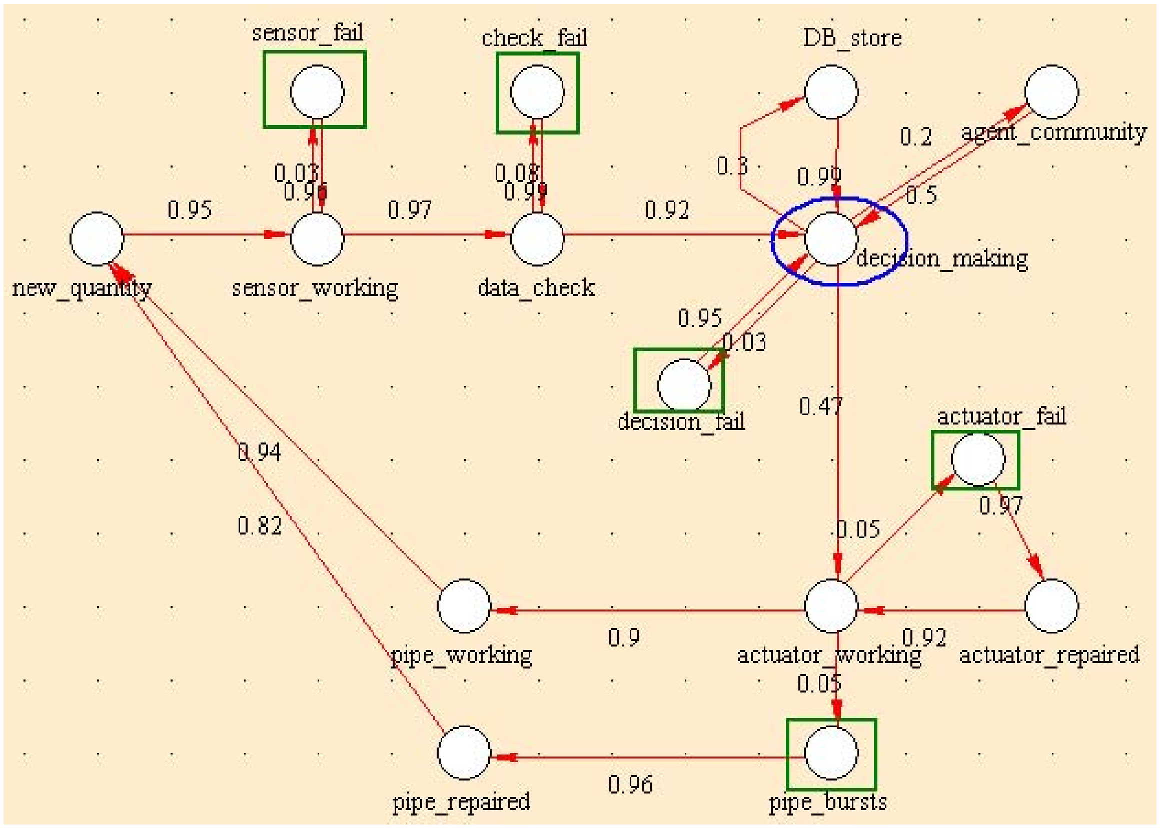


Figure 3.8: Markov chain model for an intelligent WDN.

The UML state transition diagram for a purely physical WDN is shown as Fig. 3.9. Similar to its CPS counterpart - the intelligent WDN, a local node can exchange water with its neighboring nodes, and the neighboring nodes can in turn interact with each other. However, in a purely physical WDN, the water exchange is not controlled by agents, and therefore no intelligent decision support or automated failure mitigation exists.

The Markov chain model for a purely physical WDN is depicted in Fig. 3.10. The “sensor_detection” state (on the far left) represents the collection of data when a “new_water_quantity” occurs. In this state, information is reported for accounting purposes, not for decision support. The “actuator_control” state (circled) is the only state in the system where some measure of intelligence is present in managing the

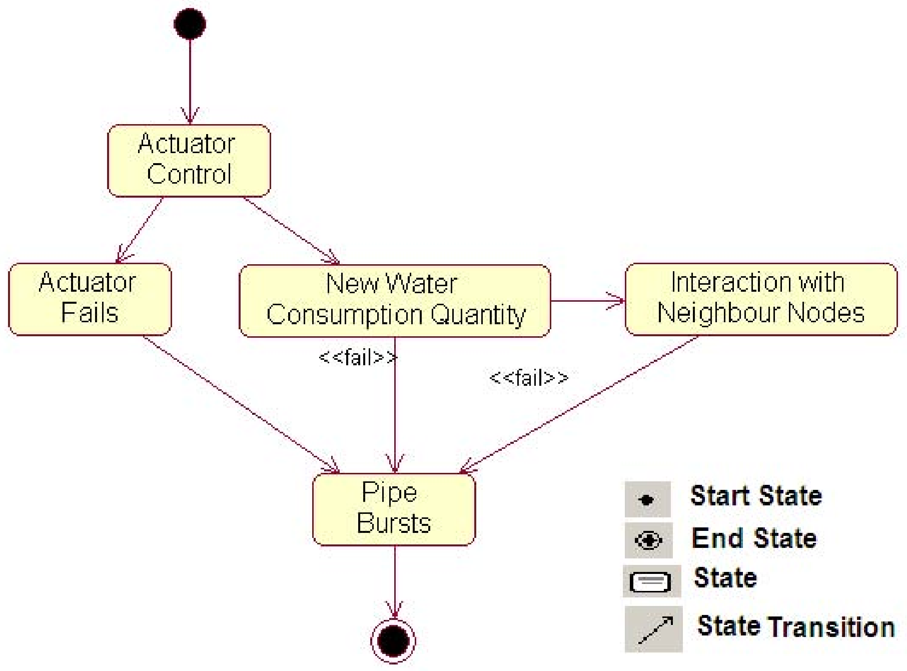


Figure 3.9: State transition diagram for a purely physical WDN.

water on the physical network. However, the actuator control is itself vulnerable to failure, the occurrence of which may lead to the "pipe_bursts" state. This state is considered an absorbing state, as the physical system is assumed to fail if a pipe bursts in the network. This is based on the assumption that the lack of real-time communication prevents timely alert of this failure and considerably delays repair. Similar to the Markov model for the CPS, the transition probabilities have been assigned based on understanding of the semantics of the system. An extension planned to this work is derivation of these probabilities from field data.

The analysis editor provided by SHARPE can be used to compare various reliability attributes, e.g., mean time to system failure ($MTTSF$), of the physical and intelligent (cyber-physical) WDNs. For the CPS, the $MTTSF$ ranges from 86.93 to 175.08 sec, while for the purely physical WDN, it ranges from 3.94 to 4.07 sec. As expected, the cyber infrastructure delays system failure. The expected reward ($Exrt$)

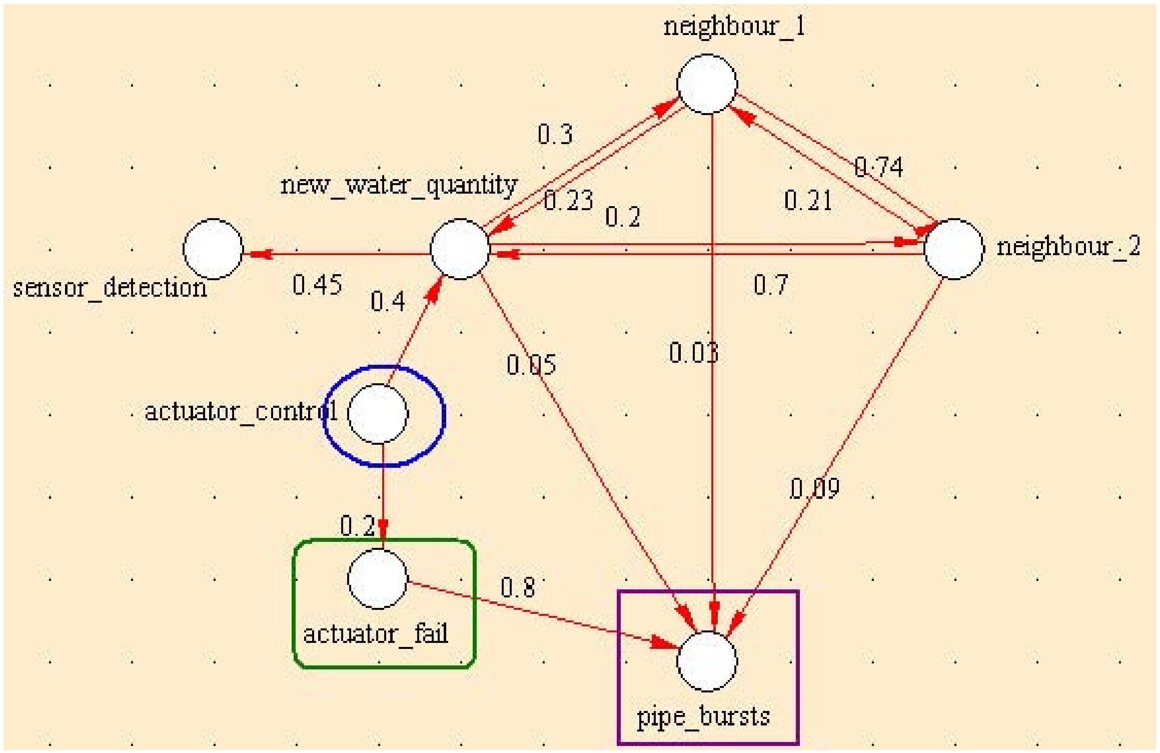


Figure 3.10: Markov chain model for a purely physical WDN.

is a parameter that can reflect the relative intelligence level of the two systems. For the CPS, the *Exrt* at t = 100 sec is 8.66, as compared to 5.22 for the purely physical WDN.

3.3 A SEMANTIC AGENT FRAMEWORK FOR MODELING WDNs

The multi-agent model described in Section 3.1 was extended to incorporate semantic agents, which represent dynamic integration of information from the sensor networks with semantic services to facilitate real-time decision support in the WDN. The resulting semantic agent framework is described below.

3.3.1 Sensor Information Ontology. Semantic interpretation is carried out on semantic streams of verified data, each of which is defined in a domain-specific ontology associated with the agent. Generally, an ontology is a description, e.g., a formal specification of the relationships among a number of entities. The notion of ontology utilized in this dissertation is a model that describes semantic relations among components of the physical and cyber infrastructures, respectively, as well as interdependencies across the cyber-physical boundary.

Each component in the ontology model is a unique class in terms of implementation, with properties and parameters described in the class definition. The relations define how classes can be related to one another. Semantic interpretation is implemented through distributed software with capabilities of extraction, analysis, and processing of the semantic stream. The definition of an ontology for the WDN domain facilitates the extraction of useful information from the heterogenous data, unifies information presentation, and permits software and information reuse, so as to reduce information redundancy during the process of semantic interpretation by the agents.

Figure 3.11 shows the information hierarchy for failure detection through the semantic interpretation process, given the verified sensor data. In this figure, which is a UML class diagram, each block (class) represents one type of semantic stream in the WDN; i.e., the ontology captures the semantic relationships among the heterogeneous data streams.

Figure 3.11 shows how a failure in the WDN can be detected by the agent in the event of physical or cyber failure, the latter of which occurs when data falls outside a pre-defined safety range. Failures in the physical infrastructure of a WDN are of two main types: physical failure due to excessive values of pressure and elevation, or biochemical failure due to excessive quantities of a biochemical substance or discovery of unknown biochemical materials. Failures in the cyber infrastructure can be attributed to either human error (accidental or malicious), or malfunction of computing devices. The ultimate determination of failure is carried out by the agent, which has authority over all sensors deployed within its administrative scope.

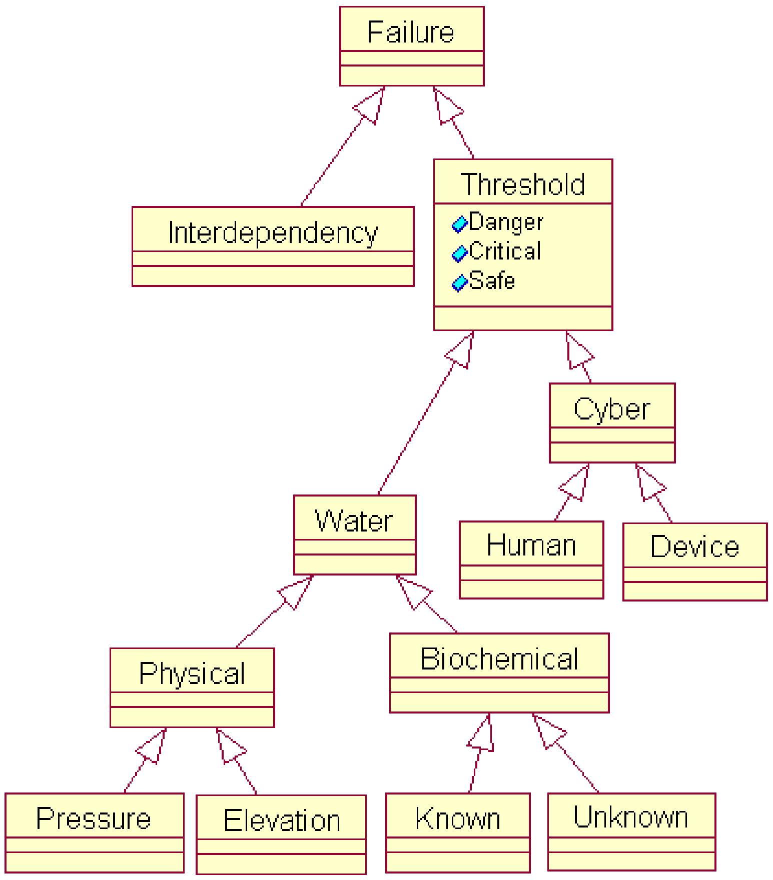


Figure 3.11: Failure detection ontology for a WDN.

The sensor information ontology captures the semantics of entities (classes in the UML diagram) and the relations of events and objects, resulting in intelligent reasoning capability beyond what sensors can provide through detection alone. The ontology proposed in Fig. 3.11 is specific to the WDN domain, but can be readily adapted to other CPSs.

3.3.2 Model for Semantic Services. Based on the sensor information ontology proposed, we can develop components for implementing conversion of semantics between classes in the information processing hierarchy. This conversion is carried out by extracting new semantic information from existing data streams. In other words, the components encapsulate the semantic service into a “black-box” containing the execution method, which takes as input information corresponding to events detected by sensors and generates as output a number of meaningful new events.

A semantic service model is proposed to overlay the ontology defined in Fig. 3.11. The semantic services can be categorized into two types: i) supplementation services, which supplement input events with additional semantic annotation; and ii) transformation services, which produce new semantic streams. Supplementation services can only identify additional properties carried by the input event. For example, a sensor has detected that the water pressure in a certain area has exceeded the safety threshold and reports this event to its semantic service component, which can be a sensor or multiplexer at a higher level of the information hierarchy. The semantic service model associated with this component will add the geographical location as an additional identifier to distinguish this event from events reported from other areas. Such functionality is particularly useful for distributed control and management in the context of CPSs, where a service may not correspond to a centralized component that physically exists on one device; it can be physically implemented on several distributed devices, but logically exist as a single service.

In contrast, a transformation service automatically terminates the input semantic stream, and generates an output semantic stream for propagation on the ontology. The essence of this type of service is semantic transformation, where the input and output events are different classes in the ontology. One typical semantic transformation is generalization. For example, in Fig. 3.11, an excessive pressure quantity will be interpreted as physical failure due to an abnormal pressure value. Later on, the semantic stream of physical failure will be propagated to a higher level for ultimate decision making, instead of the semantic stream of abnormal pressure quantity, which no longer exists. Transformation services can greatly reduce the complexity of the data stream, by extracting only unique and necessary information.

Therefore, transformation is the main semantic service that reconciles heterogeneity, by transparently unifying semantically similar streams.

The benefits of this semantic service model and information ontology include the reduction of information redundancy, pre-processing and abstraction of data for the agent, and the facilitation of semantic queries. A cyber component can issue a query that requests that a certain data stream with desired semantics be provided to a given component device to diagnose whether failure exists at the queried level. The SSM further enhances this ability, processing the query based on semantic similarity to other queries, thereby enabling support of imprecise queries. Greater detail on this feature is presented in Section 3.4.

3.3.3 Semantic Agent Framework. Figure 3.12 illustrates how the agents use the information collected by sensors and the interpreted semantics based on the defined ontology. Raw data is obtained from sensor networks, and since each agent is an independent entity in charge of a particular geographical region, sensors located in distributed areas are managed by different agents (with possible overlap). For a semantic service component, the input semantic events are preconditions of the service. The postconditions, i.e., the processed output semantics are provided to the agents for further computing.

To implement the service in C++, the properties of the service are parameterized, and the execution method of the service becomes the corresponding method of the service class. For example, consider the *Pressure* to *Failure* branch of Fig. 3.11. Sensors are treated as services with only output semantics, which are parameterized into data that can be used by superior service components (those at a higher level of the information hierarchy). Each component has been specified with a service name and associated parameters. Each service takes the output of an inferior component as the input to its execution method, and inherits the parameters to ensure that attributes of a potential failure source (such as pressure, failure time, or location) are

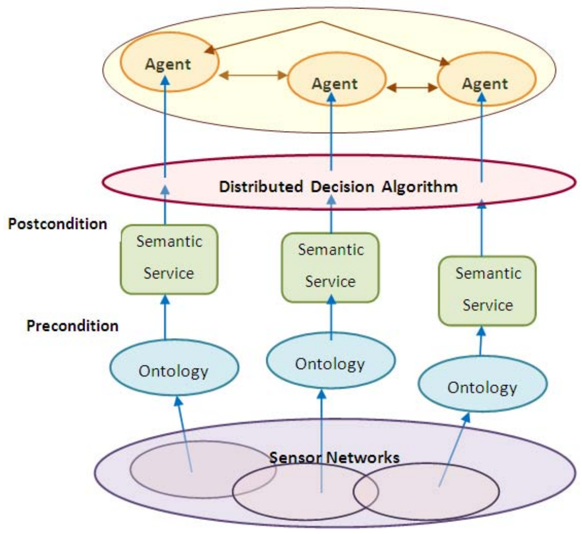


Figure 3.12: Semantic agent framework.

not lost during information propagation on the ontology. Pseudocode for our C++ implementation is shown in Fig. 3.13.

3.4 RECONCILING DATABASE HETEROGENEITY

In developing a multi-agent model for a CPS, our focus in this subsection is on reconciling heterogeneity among the data sources underpinning the intelligent decision support. We utilize SSM to establish semantic interoperability among heterogeneous data sources, while maintaining high performance and local autonomy.

3.4.1 Model Structure. The SSM is an advanced semantic processing model that supports our semantic services by extracting the semantics of access terms from underlying local databases and forming a hierarchical information structure. The SSM increases the efficiency of information retrieval from distributed sources by merging similar semantics. The imprecision allowed in queries as a result of using SSM extends beyond the syntax and linguistics, to the location of data, which is a vital capability for critical infrastructure systems such as CPSs.

```
Sensor{
water_sensor, geoID,[width,length,height],
/* properties of sensors: water detection, geographical ID, location*/
 Outputs(pressure, elevation, biochemical, location);
/*the parameters can be detected by sensor* /
}

Pressure component:
Pressure_Service{
  service(pressure),
  /*service indicates execution method and the parameter is pressure*/
  Inputs (sensor(water_sensor, geoID, [width,length,height]));
   If (pressure > normal range)
    Outputs (pressure_normal (false), detected (pressure,geoID,T));
    /*add the judgment result and time T*/
}

Physical component:
Physical_Service{
  service(physical_failure),
  Inputs(pressure_service(pressure_normal(false), detected (pressure,geoID,T)));
   If ((elevation < normal range) && (pressure_normal = false))
   /*guarantees pressure is the unique reason*/
    Outputs(physical_normal(false), detected(physical_failure,pressure,geoID,T));
   /*inherit inferior attribute*/
}

Water component:
Water_Service{
 service(water_failure),
 Inputs(physical_service(normal(false), (physical_failure,pressure,geoID,T)));
   If ((biochemical_normal = true) && (physical_normal = false))
   /*same as above*/
   Outputs(water_normal(false), detected(water_failure,physical _failure,pressure,geoID,T));
}

Threshold component:
Threshold_Service{
 service(failure),
 Inputs(physical_service(normal(false), detected(physical_failure,pressure,geoID,T)));
   Switch(detected(pressure))
   {
      Case (within range for safe): service terminates;
      Case (within range for critical): send (pressure,geoID,T) to database;
      Case (within range for safe): output system failure alert;
      Default: service terminates;
    }
 Outputs(system_failure_alert, detected( water_failure,physical _failure,pressure,geoID,T));
}
```

Figure 3.13: Pseudocode for semantic service.

By maintaining a hierarchical meta-data structure based on the information retrieved from the underlying local databases, the model can intelligently resolve terminology differences using predefined word relationships from a standard thesaurus or dictionary. The Cambridge Scientific Abstracts Engineering Thesaurus [47] is one such reference for the WDN domain. It defines a set of standard hydraulic access terminologies, the semantic categories to which they belong, and the semantic relationships among them.

A sample SSM for a WDN is shown in Fig. 3.14, which for brevity presents only a few of the numerous potential summary schemas that could potentially exist.

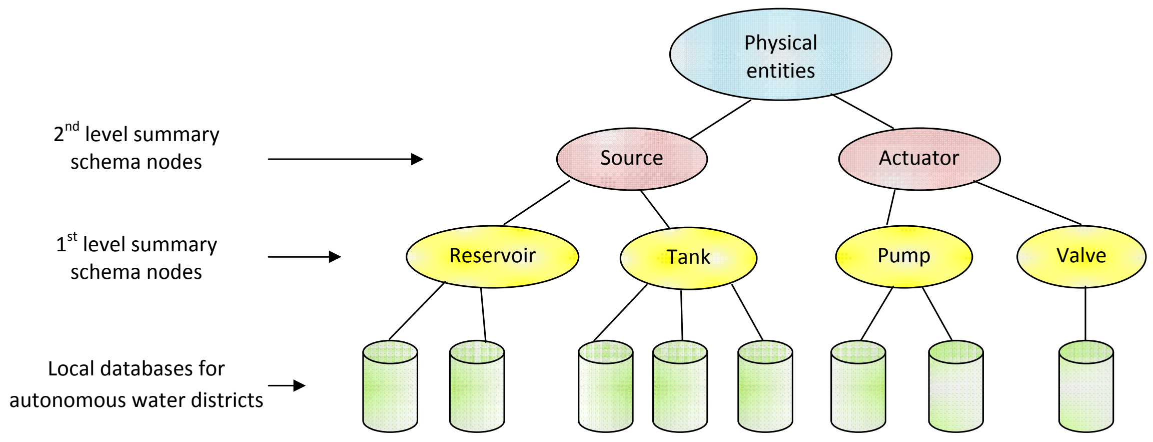

Physical entities
2nd level summary schema nodes
Source
Actuator
1st level summary schema nodes
Reservoir
Tank
Pump
Valve
Local databases for autonomous water districts

Figure 3.14: Summary Schema Model for WDN access terms.

At the bottom of the figure are the local databases (one for each autonomous water district) that represent the sensor data, or databases with physical configuration or historical information. The local schema associated with each local database is a list of access terms, e.g., pump, reservoir, for the data within. Based on the definitions in the engineering thesaurus, the summary schema can be formed by mapping the access terms of lower-level nodes to their hypernyms and resolving semantic similarities among these hypernyms. In linguistics, a hyponym is a word or phrase whose semantic field is included within that of another word, which is a hypernym. In computer science, this relationship is denoted as an "is-a" relationship; i.e., the hypernym describes a semantic relation in which one word is a specific type of another. For instance, the fact that a "reservoir" in the context of a WDN is a type of "source" describes the hyponymic relationship between these two access terms. Therefore, each summary schema node is a logical database that contains meta-data representing the abstract and essential contents of the local schemas of its children. As a result, higher levels of the SSM have very compact schema.

3.4.2 Semantic Distance Metric. Fundamental to operation of the SSM is the ability to quantify the semantic similarity of two terms. The semantic distance metric (SDM) is defined to this end [48]. A general-purpose thesaurus can be used to calculate the SDM for pairs of terms, but is of limited use in specialized applications such as WDNs, as it cannot capture the technical nuances implied. As mentioned in the previous section, a specialized dictionary or thesaurus can serve as a reference for creating a hierarchy of hyponyms with synonym cross references between subtrees of the SSM, facilitating the calculation of the SDM. Each subtree of the SSM is comprised of individual meaning clusters or semantically-linked words.

The SSM is essentially a taxonomy composed of i) pairs of terms and ii) hypernym and synonym links connecting these terms. The SDM is a weighted count of the number of links between two terms, i.e., a large number of links implies that two terms are relatively different in meaning. The weighting is necessary, because different links represent different aspects of semantic similarity. As an example, a link connecting synonyms has a lower weight than a hypernym/hyponym link, because the former provides a more precise description of semantic closeness. Figure 3.15 demonstrates the calculation of the SDM for terms describing potential failures in a WDN. Such a hierarchy could facilitate query of failure history databases. The leaf-level terms are in the local schemas, and the upper-level terms are the summary schemas.

If we assign a weight of 1 to all links, then the terms with SDM = 1 as imprecise references to “pipe break” include “pipe burst,” “pipe cutoff” and “pipe corrosion.” Similarly, a term with SDM = 2 from“pipe-break” is “link failure,” and a term with

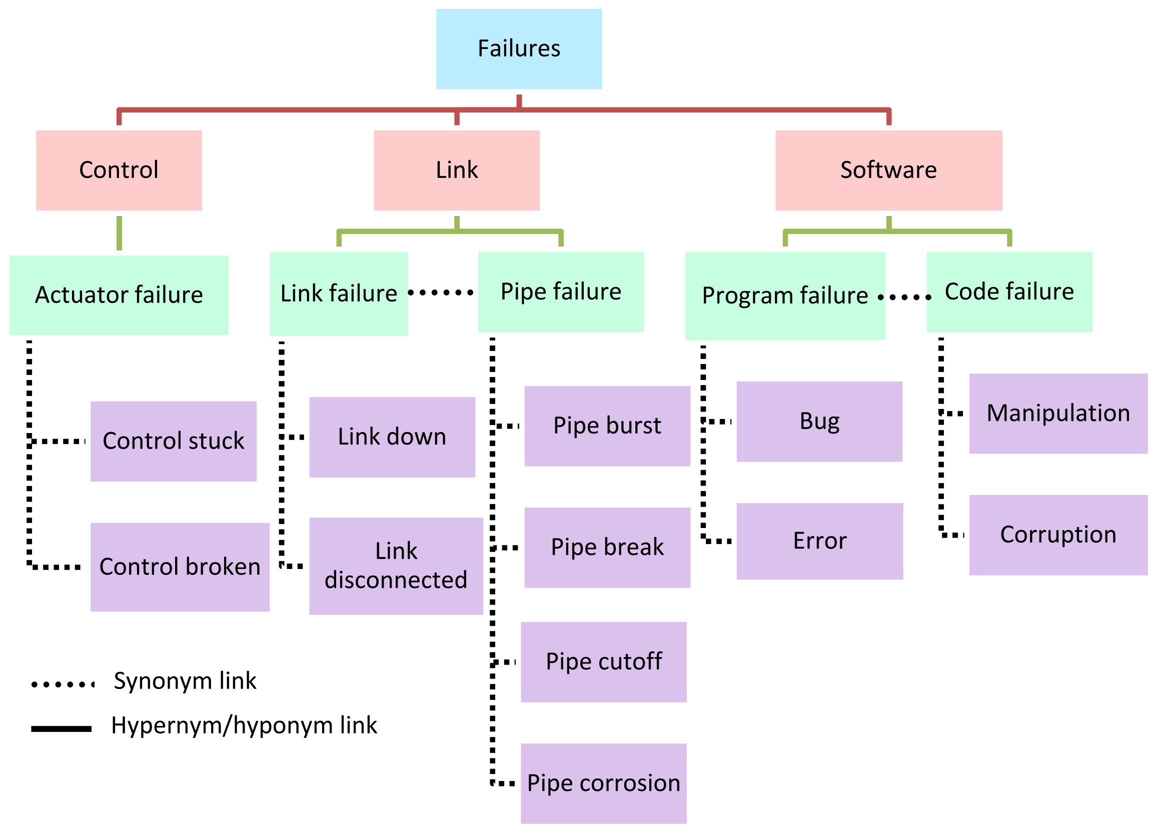


Figure 3.15: Hierarchy of WDN failure terms.

SDM = 3 is "link down." Beyond use of the engineering thesaurus, design of such an SSM taxonomy can be based on knowledge of the application, experience with and statistical analysis of previous queries, and even on the bias of the taxonomy compiler [30], each of which can introduce subjectivity to a different extent.

3.4.3 Imprecise Queries. Imprecise queries allow users to specify data references in their own terms, rather than the system's predefined terms. By using the SDM in the summary schemas hierarchy, imprecise data references to semantically similar system access terms can be quantitatively compared. The SSM can be configured to set the maximum SDM acceptable for a match - the higher this value, the greater the tolerance for imprecision in the query terminology. In multidisciplinary applications such as CPSs, it is prudent to allow for greater values of SDM, as the system design and implementation is carried out by practitioners from a diverse array of backgrounds - each of which uses specific and possibly different terminology.

In the case of a WDN, queries can take place at any layer of the sensor information hierarchy defined on the ontology. Imprecise-query support eliminates the need for knowledge of the location of or local access terms (the leaf nodes in Fig. 3.14) for the data. Imprecise queries are denoted as such by the entity making them. When this notation is identified, the summary schemas structure matches the reference to the semantically closest precise reference, based on the SDM. From this point onward, the imprecise query is processed as if it were a precise query.

More specifically, the SSM processes imprecise queries in a fashion similar to that of precise queries, i.e., parsing the query, sending data access requests to remote data sources, and combining the data accessed according to the operations specified in the query. However, for imprecise queries, a reference resolution phase is added between parsing the query and sending the remote access requests [30]. The resolution involves a search that begins at the origin node of the query, and searches upwards in the SSM hierarchy until a node is encountered that has a potential match in its summary schema. The search continues downwards in the subtree rooted at the potential match node. If an access term is found that is within the maximum SDM, it is considered a match.

As a simple example of imprecise-query processing, Fig. 3.16 depicts the procedure for querying a multidatabase platform (which can include both history and sensor databases) for information about WDN failure. The query issued may specify “link down” as the type of failure sought. The local access term, which is the name used to describe the data in the local database, may be “pipe burst,” which is more typically used in hydraulics. This difference in terminology can be resolved if a maximum SDM of 3 or greater is specified. This enables the use of heterogeneous databases and sensor networks, without compromising the decision support carried out by the cyber infrastructure.

```
Parse:    parse query at origin node;
          if all data references are precise then
              go to Execute;
          else
              send query to immediately higher node in SSM hierarchy;
          end
Resolve:  foreach imprecise data reference do
              calculate SDM for imprecise data reference and local summary schema;
              if any local term is within max SDM of the imprecise data reference then
                  search subtree rooted at this node for access term that is within max SDM;
                  if such a term is found then
                      replace imprecise data reference with precise term;
                  else
                      reference is still considered imprecise;
                  end
              end
          end
Continue: if all data references in query are precise then
              go to Execute;
          else
              else if root of SSM hierarchy has been reached then
                  reject query - nothing found within max SDM;
              end
              send query to immediately higher node;
              go to Resolve;
          end
Execute:  execute query using standard multidatabase facilities;
```

Figure 3.16: Imprecise query processing algorithm for fault detection in a WDN.

As depicted in Fig. 3.2, the data retrieved as the result of the query is sent to the agent for use in the decision support algorithm that will determine settings for physical control devices such as valves. Computational techniques such as game theory can be used to ensure that the settings are configured to prevent future recurrence of the failure [15]. The agent can also trigger a sequence of actions that lead to repair of the pipe, or can reroute water to avoid exceeding the capacity of downstream pipes. Our investigation of real-time feedback-based control carried out by the cyber infrastructure of a WDN is reported in [9].

3.5 INTEGRATED CYBER-PHYSICAL SIMULATION OF A WDN

Simulation was fundamental to understanding the operation of the WDN, and facilitated the development of the qualitative models. However, a simulator capable of representing both the cyber and the physical infrastructures of the WDN could not be identified. We proceeded to develop an integrated cyber-physical simulator for WDNs, using EPANET 2.0 [8] and MATLAB to represent the physical infrastructure and the decision support algorithms used to control the allocation of water, respectively. EPANET can capture the layout of a WDN and track the water flow, pressure, depth of water in tanks, and the concentration of given chemical substances. A simple model created by EPANET is shown in Fig. 3.17, which includes all the necessary elements of a physical WDN. The reservoir is the major component that provides water to consumers that include tanks and junctions. The actuators are pumps (which control the water pressure) and valves (which turn the water flow on or off). The legend of Fig. 3.17 denotes the water level within the nodes or pipes.

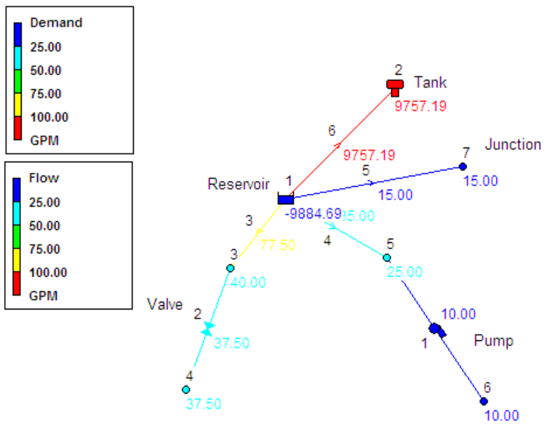


Figure 3.17: A simple topology in EPANET.

In the WDN depicted in Fig. 3.17, the reservoir is providing water to the tank and a number of different junctions. The reservoir in this figure always contributes water to the network, so its demand value is negative - in this case 9884.69 gpm. The tank consumes the highest amount of water. Each junction is also labeled with its demand value, and each pipe with its flow speed. The entire graph is color-coded to simplify the categorization of demand or flow. The demand values of pumps and valves vary in accordance with the nodes they control.

MATLAB has powerful computational capability and can support advanced techniques, i.e., distributed decision support algorithms, for managing the water resources. The procedure for simulation of a WDN with EPANET and MATLAB is depicted in Fig. 3.18, and can be generalized to other CPS domains.

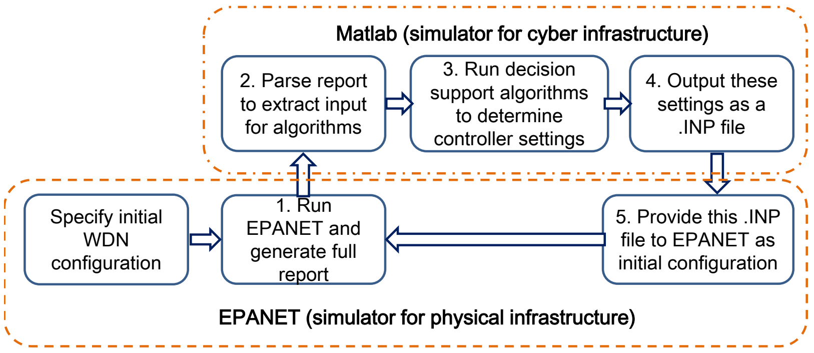

Matlab (simulator for cyber infrastructure)
2. Parse report to extract input for algorithms
3. Run decision support algorithms to determine controller settings
4. Output these settings as a .INP file
Specify initial WDN configuration
1. Run EPANET and generate full report
5. Provide this .INP file to EPANET as initial configuration
EPANET (simulator for physical infrastructure)

Figure 3.18: Procedure for simulation of a WDN.

4 ONTOLOGIES AND DECISION SUPPORT

In this section, we propose and validate a decision support system for WDNs that makes use of cyber infrastructure for automated reasoning. The agent-based paradigm introduced in Section 3 is extended to enable the use of ontologies in classification of failure events and identification of appropriate countermeasures. The work presented in this section has been submitted to the Proceedings of the 45th Hawaii International Conference on System Sciences, and is under review as of Aug. 2011. Extensions to this work planned for the immediate future include generalization of the ontologies to other flow transport CPSs, including smart grids and ground transportation networks.

The premise underlying the automated failure mitigation proposed in this dissertation is that a database is available that associates a set of countermeasures with a set of failure types. As an aside, we investigated data retrieval from such a database, with the goal of determining the most efficacious search techniques. The results of our investigation are presented in Appendix D and have been omitted from this section in the interest of coherence.

4.1 FUNCTIONAL MODEL

The very first step in constructing a CPS ontology model is to identify the major functional components of the system. Figure 4.1 depicts the six main functional components of a CPS used for transporting a physical commodity. WDNs, smart grids, and intelligent transportation systems can be abstracted in this fashion, as they transport water, electric power, and vehicles; respectively. In such transport systems, both discrete and continuous flows (the values of which can be quantized) are carried by passive entities, and controlled, commanded and monitored by actuated components. The cyber components (where the agents reside) control both the actuators (directly) and passive entities (indirectly), and provide intelligent decision support for efficient management of the transport system. Figure 4.2 depicts the instantiation of the functional model of Fig. 4.1 for a WDN.

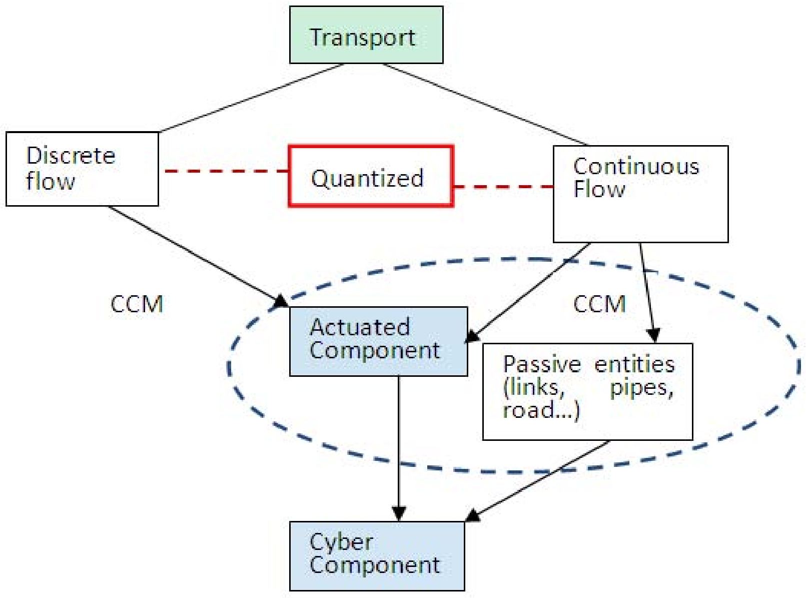


Figure 4.1: Functional model of a CPS for commodity transport.

4.2 COMPONENTS IN WDN ONTOLOGY FRAMEWORK

On the basis of the use case diagram of Fig. 3.1 and the functional components of Fig. 4.2, we present the building blocks of the ontology framework - the classes. We use Protégé 4 [49] as the platform for creation of the WDN ontology.

4.2.1 WDN Ontology Class. The topmost classes of the WDN are shown in Fig. 4.3, according to the functionalities identified in Fig. 4.2 and Fig. 3.1. The

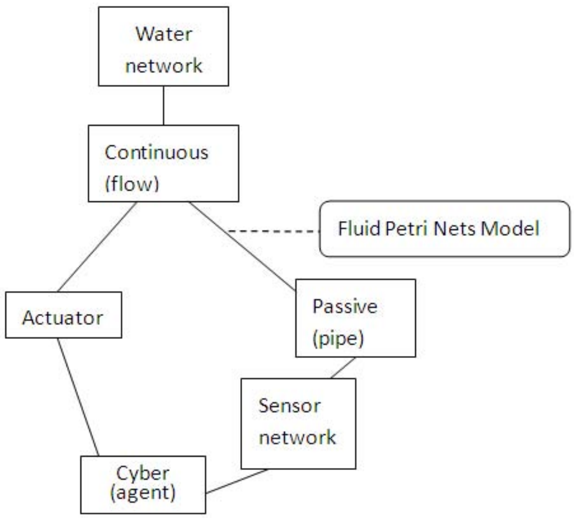
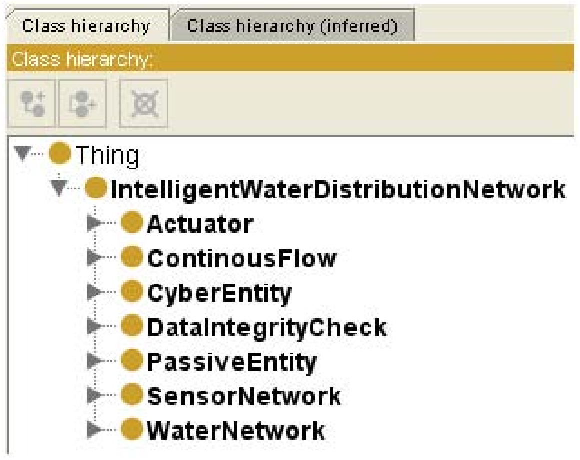


Figure 4.2: Functional model of WDN.

class *Thing* is the set containing all the subclasses, under which all other classes are defined.



Thing
- IntelligentWaterDistributionNetwork
  - Actuator
  - ContinousFlow
  - CyberEntity
  - DataIntegrityCheck
  - PassiveEntity
  - SensorNetwork
  - WaterNetwork

Figure 4.3: Topmost classes of WDN.

Classes can be organized into a superclass-subclass hierarchy, which is quite similar to a taxonomy. We adopt OWL-DL [50] in Protégé to define the superclass-subclass relationships, which can be automatically computed by a reasoner and visualized in diagrams. An automated reasoner can process and parse OWL-DL to understand the relationships among defined classes - specifically determining whether a particular class is a subclass of another. The classes that we have defined in OWL-DL for the WDN are presented in Figs. 4.4 to Fig. 4.9, where the superclass-subclass relationships are clearly depicted.

The failure type class is created based on FMEA [46] and fault tree analysis [51]. The *Computer-System-Vulnerability* class is adopted from an existing ontology developed by the Resilience for Survivability group [52]. The mitigation technique database is designed to address a broad range of failures, and makes reference to [53].

4.2.2 Automated Reasoning Based on Classes. An automated reasoner can utilize the OWL-DL model to compute the inferred ontology class hierarchy, as depicted in Fig. 4.10. The graph has been generated using OWLViz, a visualization plug-in for Protégé. The blue rectangle denotes the selection made when we query the "Decision_MakingFail_Mitigation" class. The hierarchical ontology in OWL-DL facilitates identification of the superclass and subclasses of "Decision_MakingFail_Mitigation." The arcs representing "is-a" relationships have been denoted as such in Fig. 4.10.

As an example, a "PipeOverload" can be automatically identified as a type of pipe failure. Once the failure type has been identified, the associated mitigation technique can be determined, and countermeasures can be actuated for the physical components. Identification of the appropriate mitigation technique takes place in a top-down fashion, with higher-level classes being investigated before lower-level classes [11].

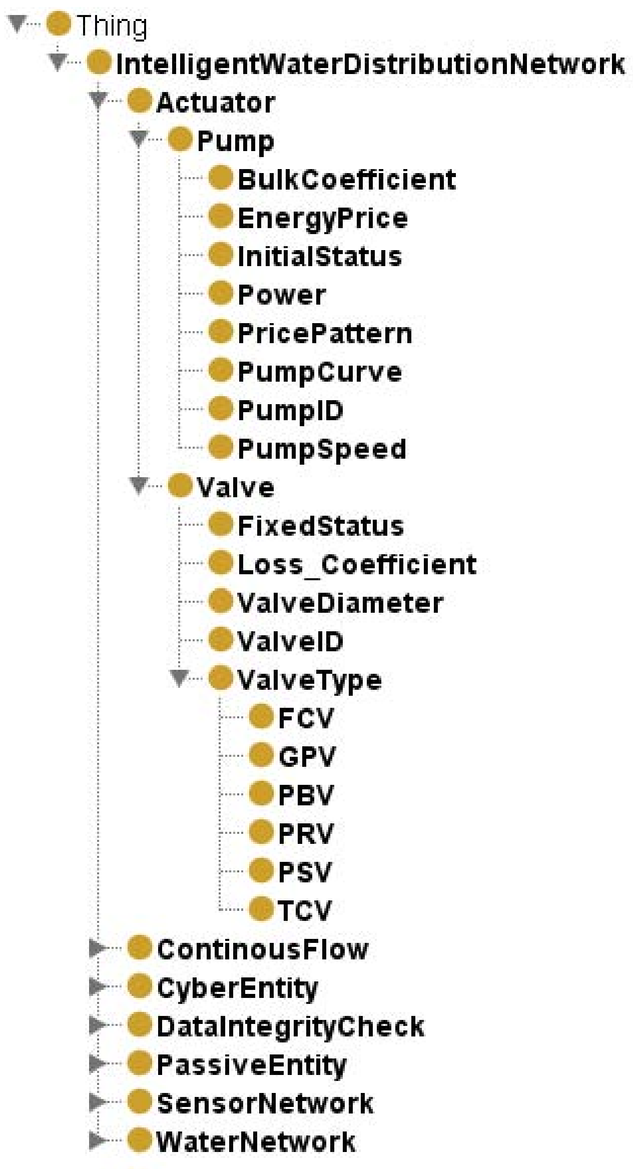


Figure 4.4: Ontologies for actuator class.

4.3 AUTOMATED FAILURE CLASSIFICATION AND MITIGATION

In OWL, properties describe relationships between classes or individuals - instances of the class and the subclass can also be viewed as individuals of the super-class. The two main types of properties in OWL-DL are “object” and “datatype.”

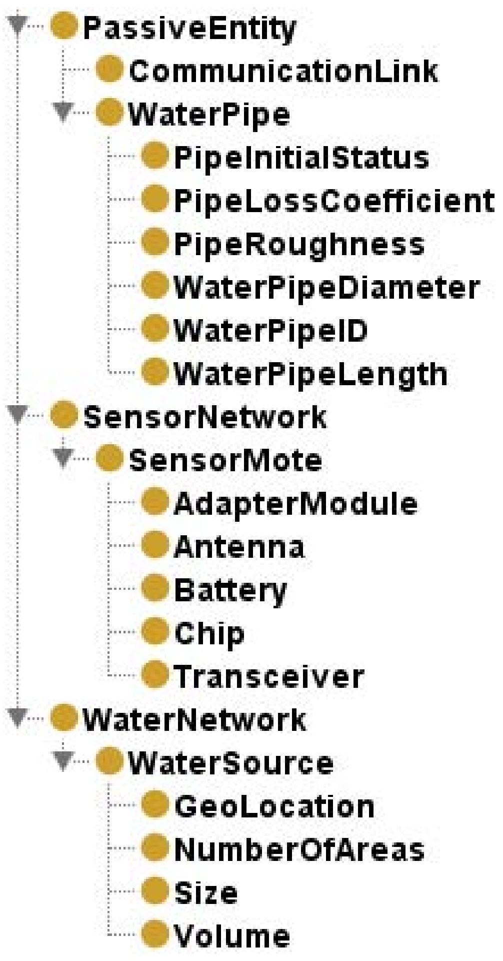


Figure 4.5: Ontologies for passive entity, sensor network, and water network classes.

We illustrate the use of these properties in automatic classification and mitigation of failure.

4.3.1 Object Properties for Behavior Reasoning. The object properties specify relationships between two classes or individuals. By OWL-DL convention, the properties are prefixed with the word "has" or "is" to clarify the meaning of the property for humans; to take advantage of the "English Prose Tooltip Generator", which uses this naming convention where possible to generate more human-readable class descriptions [49]; and to facilitate automated reasoning.

The properties we have defined for the WDN ontology are shown in Fig. 4.12.

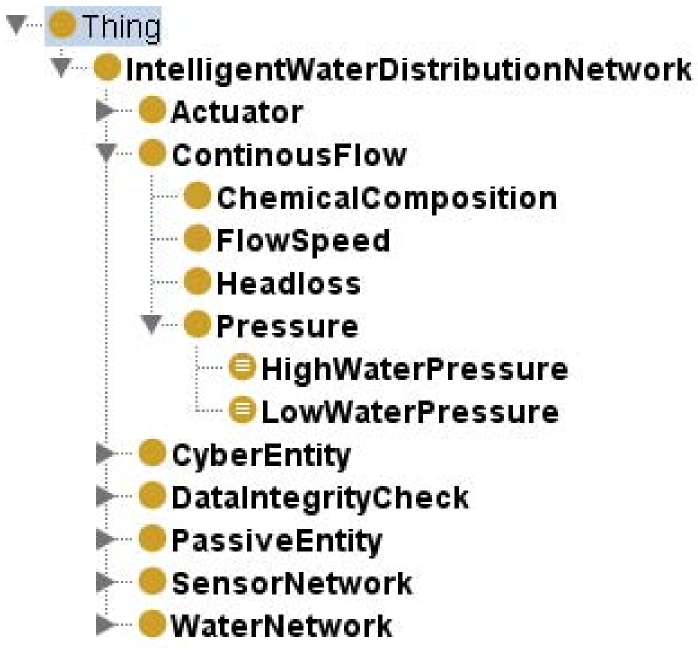


Figure 4.6: Continuous flow class.

We can further define characteristics for each object property to enrich its meaning or to constrain its domain or range. More specifically, a property can be characterized as one of the following [49]:

1. Functional: A functional property relates a given individual to at most one other individual, e.g., "computer has command over actuator", specifies that the computer can send commands to the actuator, as opposed to directly exerting control over the flow transported by the CPS.

2. Inverse functional: Properties that measure the inverse properties are functional. For example, if we defined an "isCommandedBy" property, then it is the inverse functional property of "actuator is commanded by computer."

3. Transitive: If a property relates individual a to individual b, and also individual b to individual c, then we can infer that individual a is related to individual c via the same property. For instance, an actuator can be controlled by a computer,

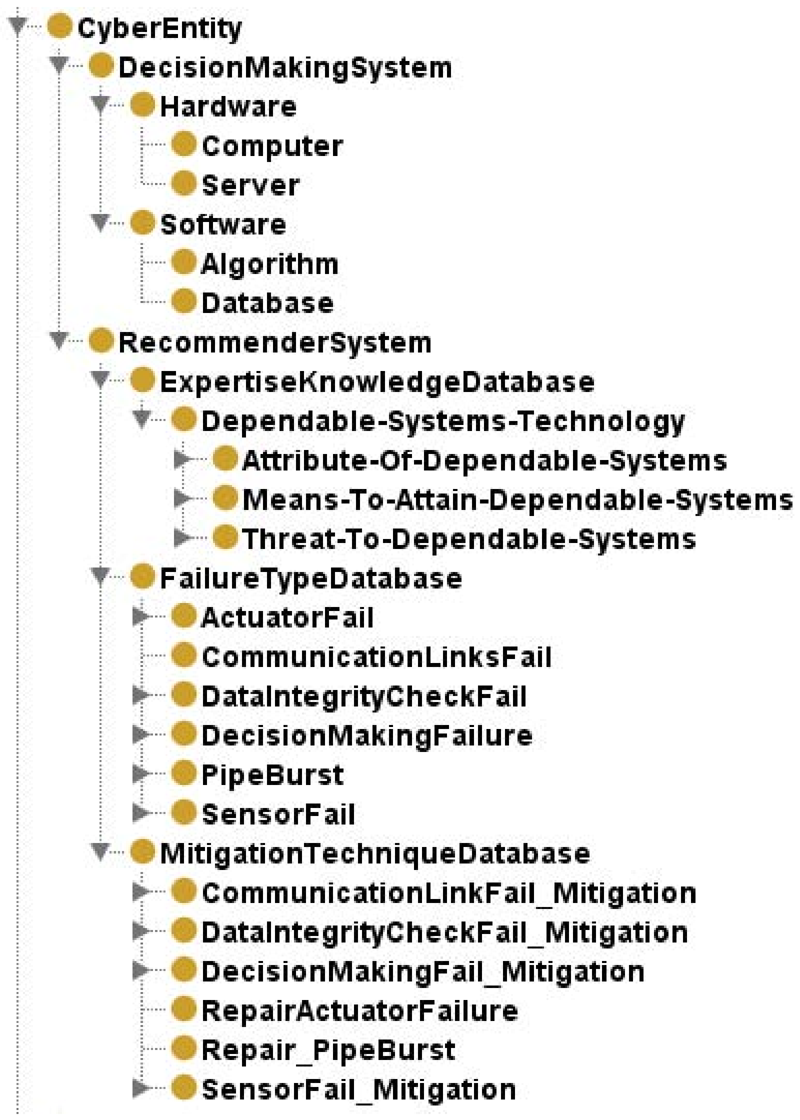

CyberEntity
DecisionMakingSystem
Hardware
Computer
Server
Software
Algorithm
Database
RecommenderSystem
ExpertiseKnowledgeDatabase
Dependable-Systems-Technology
Attribute-Of-Dependable-Systems
Means-To-Attain-Dependable-Systems
Threat-To-Dependable-Systems
FailureTypeDatabase
ActuatorFail
CommunicationLinksFail
DataIntegrityCheckFail
DecisionMakingFailure
PipeBurst
SensorFail
MitigationTechniqueDatabase
CommunicationLinkFail_Mitigation
DataIntegrityCheckFail_Mitigation
DecisionMakingFail_Mitigation
RepairActuatorFailure
Repair_PipeBurst
SensorFail_Mitigation

Figure 4.7: Subclasses of cyber entity class.

and a pipe (passive entity) can in turn be controlled by the actuator, therefore, the "isControlledBy" property can be characterized as transitive.

4. Symmetric: When individual a is related to individual b via property P, and vice versa; P is characterized as symmetric.

5. Asymmetric: Any property that is not symmetric is characterized as asymmetric. Most of the properties in our WDN ontology are asymmetric.

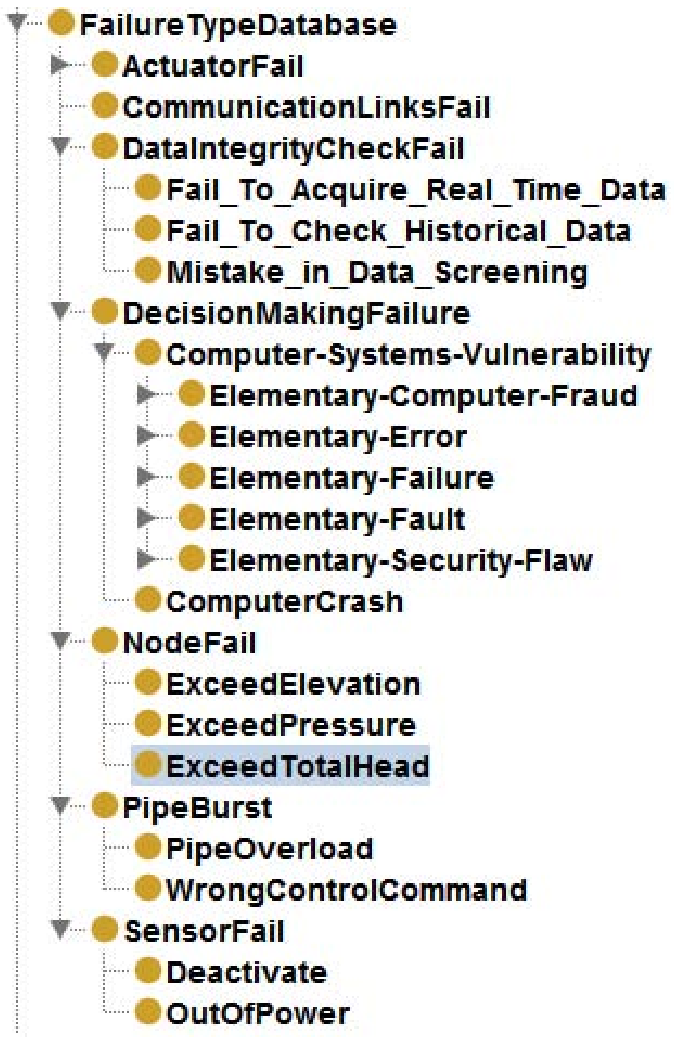
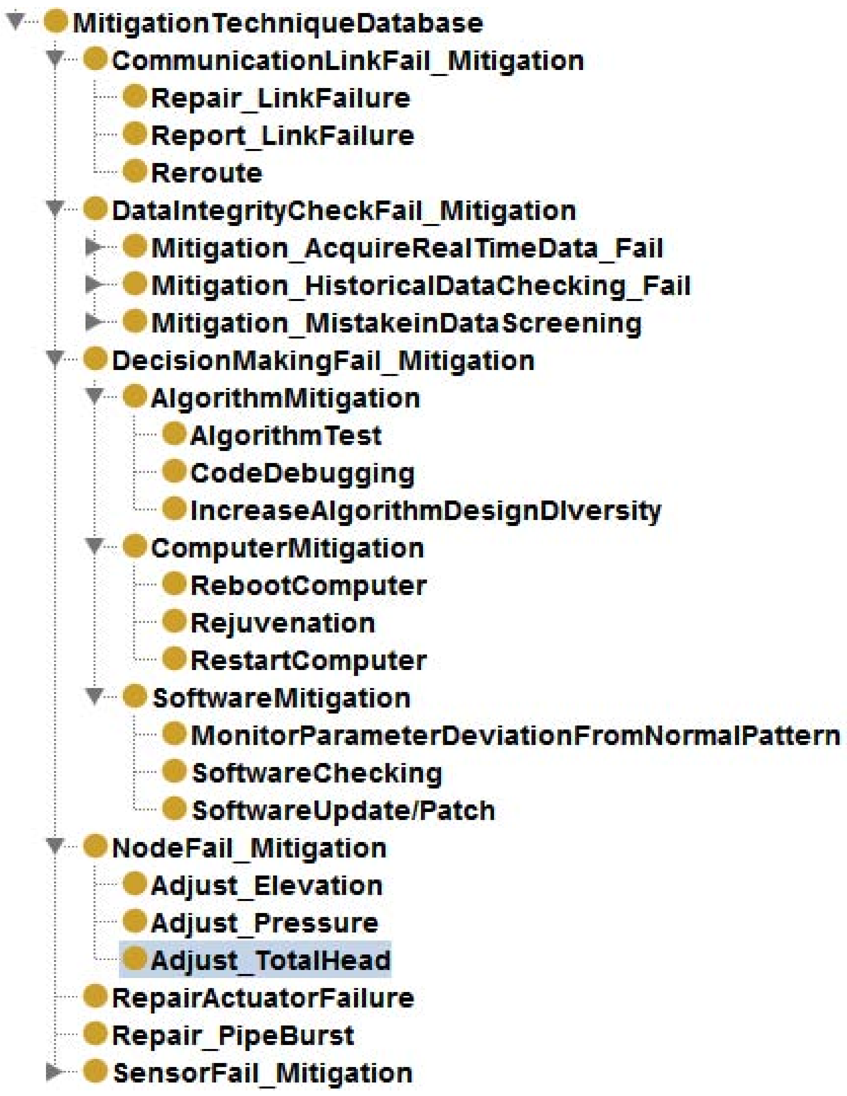


Figure 4.8: Failure type class.

6. Reflexive: A property that relates an individual to itself is characterized as reflexive. For instance, in our ontology, the "MitigationTechniqueDatabase" is reflexively related to "hasMitigationIdentification."

7. Irreflexive: Individuals related by an irreflexive property cannot be the same.

The object properties can be further characterized with a domain and a range, respectively. The object properties link classes (individuals) from the domain to classes (individuals) from the range. In the relationship "computer has command



- MitigationTechniqueDatabase
  - CommunicationLinkFail_Mitigation
    - Repair_LinkFailure
    - Report_LinkFailure
    - Reroute
  - DataIntegrityCheckFail_Mitigation
    - Mitigation_AcquireRealTimeData_Fail
    - Mitigation_HistoricalDataChecking_Fail
    - Mitigation_MistakeinDataScreening
  - DecisionMakingFail_Mitigation
    - AlgorithmMitigation
      - AlgorithmTest
      - CodeDebugging
      - IncreaseAlgorithmDesignDIversity
    - ComputerMitigation
      - RebootComputer
      - Rejuvenation
      - RestartComputer
    - SoftwareMitigation
      - MonitorParameterDeviationFromNormalPattern
      - SoftwareChecking
      - SoftwareUpdate/Patch
  - NodeFail_Mitigation
    - Adjust_Elevation
    - Adjust_Pressure
    - Adjust_TotalHead
  - RepairActuatorFailure
  - Repair_PipeBurst
  - SensorFail_Mitigation

Figure 4.9: Mitigation technique class.

over actuator", the "computer" is the domain and the "actuator" is the range. The domain and the range in OWL-DL are used as "axioms" in reasoning. It is worth noting that an axiom is one of the main components of an ontology; others include concepts, individuals and relationships. Figure 4.13 shows the characteristics and the description of the domain and range of property "hasCommandOver".

Figure 4.14 depicts an overarching map of the object properties that interconnect various classes and subclasses in the WDN ontology. The map is automatically

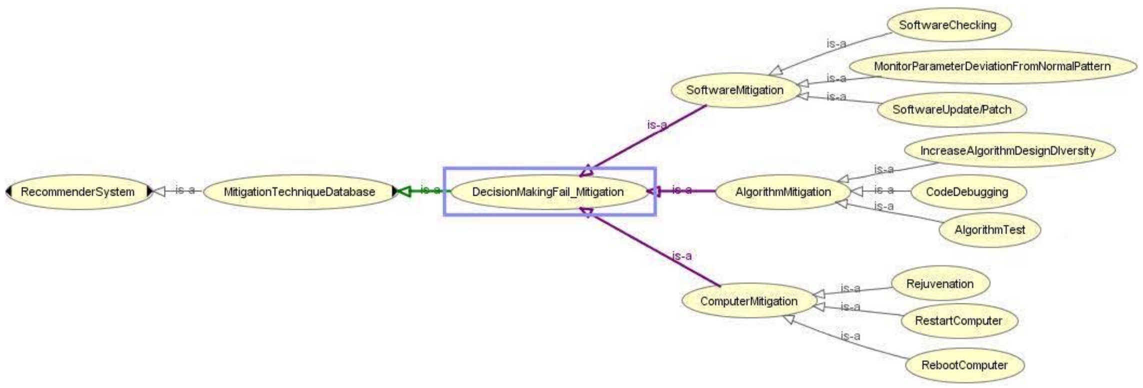


Figure 4.10: Deduced superclass and subclasses of “Decision_MakingFail_Mitigation.”

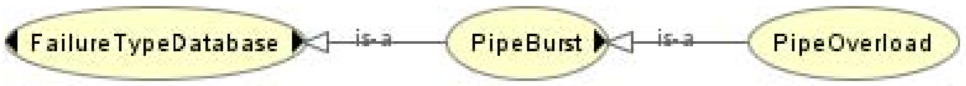
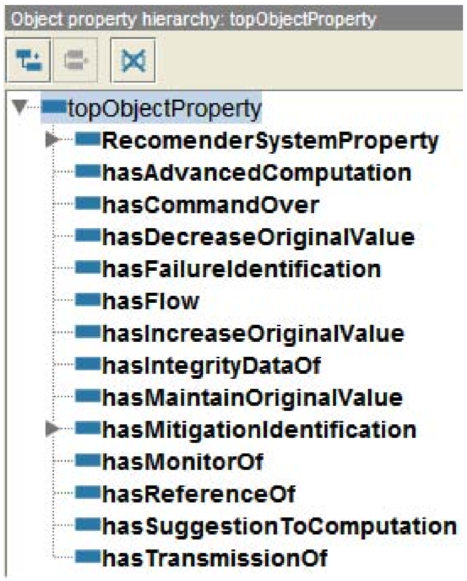


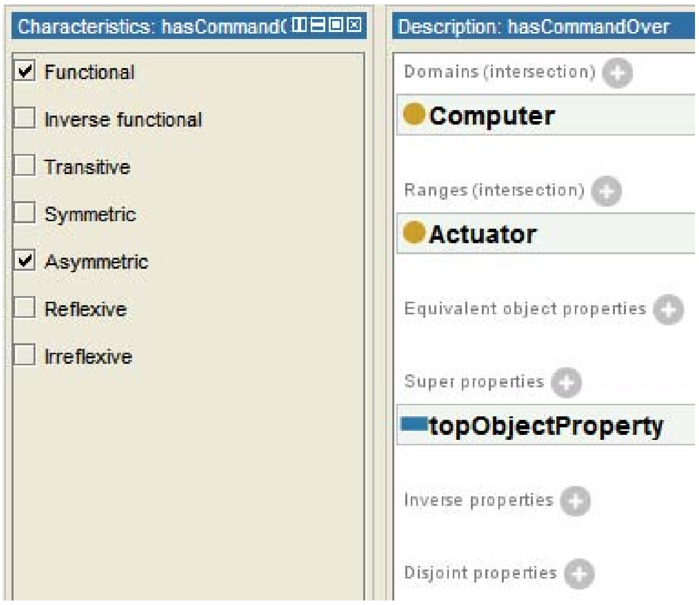
Figure 4.11: Failure type identification.

generated based on the “axioms” used in reasoning. Not all properties have been reflected in this map; it is intended to demonstrate how object properties can be used to infer relationships among different classes. Two types of arcs appear in Fig. 4.14 - solid and dashed. A solid arc represents a “superclass-subclass” relationship, such as “software” and “algorithm.” A dashed arc represents an object property, e.g., if the arrow on the arc between “Software” and “Continuous Flow” is activated, then the object property is highlighted as “Software” hasAdvanced-Computation of the “Continuous Flow”. Similarly, “WaterNetwork” hasFlow of the “ContinuousFlow”; “SensorNetwork” hasMonitorOf the “ContinuousFlow”; “CyberEntity” hasIntegrityDataOf “DataIntegrityCheck”; “Computer” hasCommandOver “Actuator”; “MitigationTechniqueDatabase” hasSuggestionToComputation for “DecisionMakingSystem”; and the “Computer” can send three different types of control


Object property hierarchy: topObjectProperty

- topObjectProperty
  - RecomenderSystemProperty
  - hasAdvancedComputation
  - hasCommandOver
  - hasDecreaseOriginalValue
  - hasFailureIdentification
  - hasFlow
  - hasIncreaseOriginalValue
  - hasIntegrityDataOf
  - hasMaintainOriginalValue
  - hasMitigationIdentification
  - hasMonitorOf
  - hasReferenceOf
  - hasSuggestionToComputation
  - hasTransmissionOf

Figure 4.12: Top object properties.


Characteristics: hasCommand(

- [✓] Functional
- [ ] Inverse functional
- [ ] Transitive
- [ ] Symmetric
- [✓] Asymmetric
- [ ] Reflexive
- [ ] Irreflexive

Description: hasCommandOver

Domains (intersection)
Computer

Ranges (intersection)
Actuator

Equivalent object properties

Super properties
topObjectProperty

Inverse properties

Disjoint properties

Figure 4.13: Characteristics and description of an object property.

commands to the valve and the pump (subclasses of the actuator), including “hasIncreaseOriginalValue”, “hasDecreaseOriginalValue” and “hasMaintainOriginalValue”.

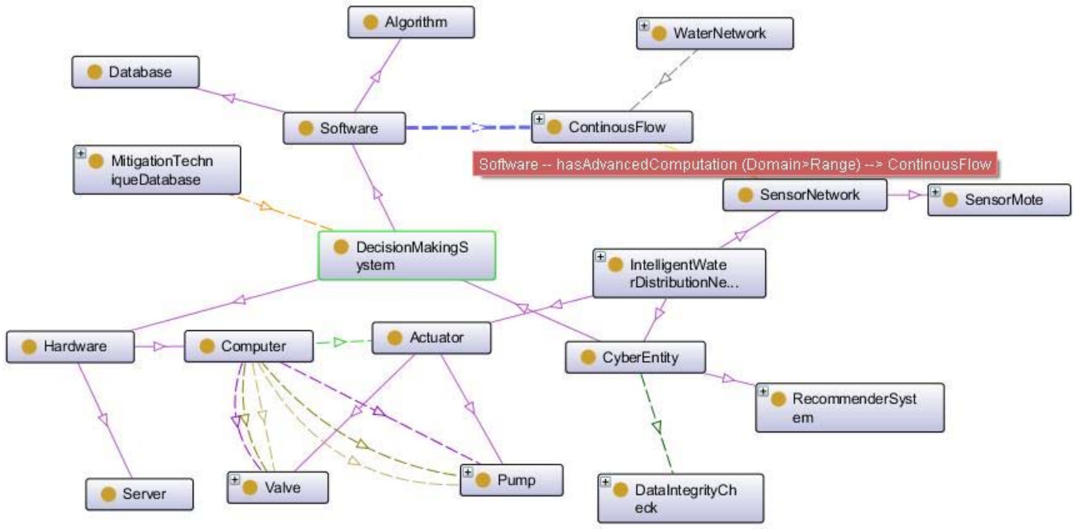


Figure 4.14: Map of object properties in WDN ontology.

Figure 4.15 depicts the interaction of the major components of the cyber entity class. In this example, once a failure is identified as being of the type “PipeOverload”, appropriate countermeasures can be automatically identified and retrieved from the mitigation techniques database. This is reflected by the connection between the “hasMitigationIndentification” property and the failure type database. The mitigation technique database has the reflexive property “hasMitigationIdentification” to facilitate identification of appropriate countermeasures. If a failure is identified as being of type “PipeBurst”, the corresponding mitigation technique is determined to be “Repair_PipeBurst”. This class will mitigate the pipe burst failure through the object property “hasPipeRepaired”. In the meantime, the mitigation technique database will trigger decision support by “hasSuggestionTo Computation”, which leads to computation of updated values for the actuator command.

4.3.2 Data Properties for Value Reasoning. Data properties link an individual to an XML schema datatype value, i.e., they describe the type of relationship between an individual and data values. For instance, we can use data properties

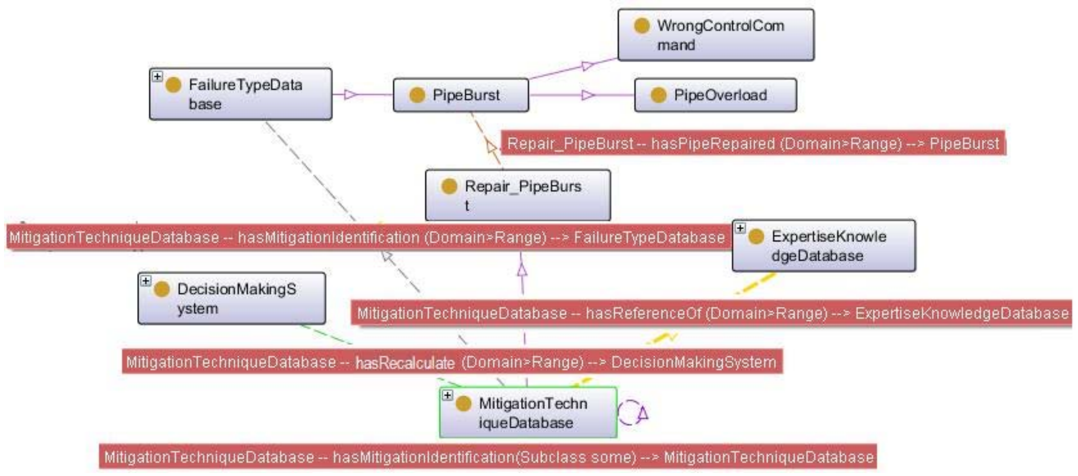

WrongControlCommand
FailureTypeDatabase
PipeBurst
PipeOverload
Repair_PipeBurst -- hasPipeRepaired (Domain>Range) --> PipeBurst
Repair_PipeBurst
MitigationTechniqueDatabase -- hasMitigationIdentification (Domain>Range) --> FailureTypeDatabase
ExpertiseKnowledgeDatabase
DecisionMakingSystem
MitigationTechniqueDatabase -- hasReferenceOf (Domain>Range) --> ExpertiseKnowledgeDatabase
MitigationTechniqueDatabase -- hasRecalculate (Domain>Range) --> DecisionMakingSystem
MitigationTechniqueDatabase
MitigationTechniqueDatabase -- hasMitigationIdentification(Subclass some) --> MitigationTechniqueDatabase

Figure 4.15: Classes and object properties relevant to failure mitigation.

to describe the pressure value of the water flow, specify respective numeric ranges for “high” and “low” flow pressure in a particular water consumption area, and determine whether the water allocated is sufficient. This judgment capability can be used to improve the dependability of the WDN. For example, we can configure a threshold value for the flow pressure, and once this threshold is exceeded (reflecting potential failure), a mitigation technique can be identified and initiated.

We use “pressure” as an example to demonstrate automated reasoning for values. Initially, we define a data property as “has-flow-pressure-value”, shown as Fig. 4.16. We then create four instances within the “Pressure” class and define the properties of each instance.

Figure 4.17 shows the data property definition for “Commercial-Area-Water-Pressure”, which is “has-flow-pressure-value 400”. This specifies that the average water pressure in a commercial area is approximately 400 pounds per square inch (psi). Similarly, the average pressure values in industrial, residential, and suburban areas are set to 700, 260 and 200 psi, respectively.

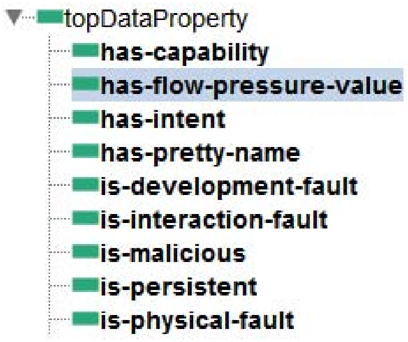


Figure 4.16: Definition of data property.

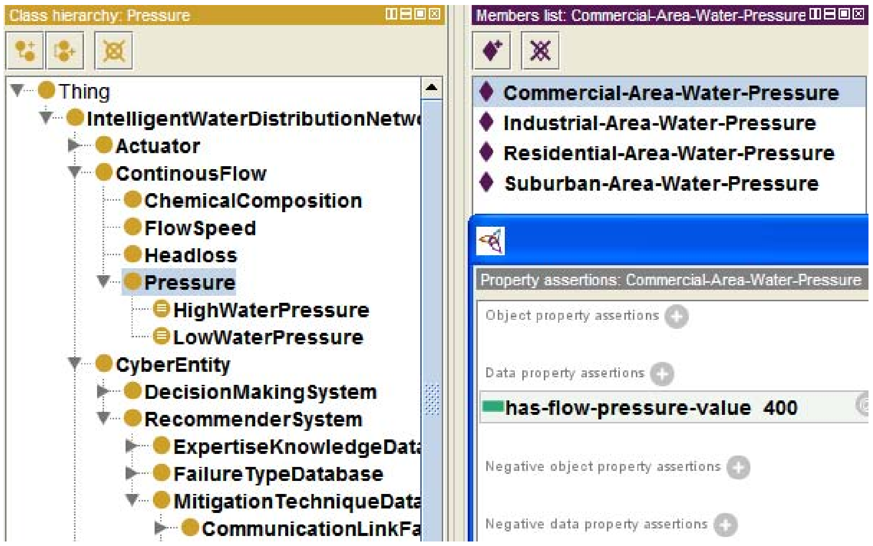
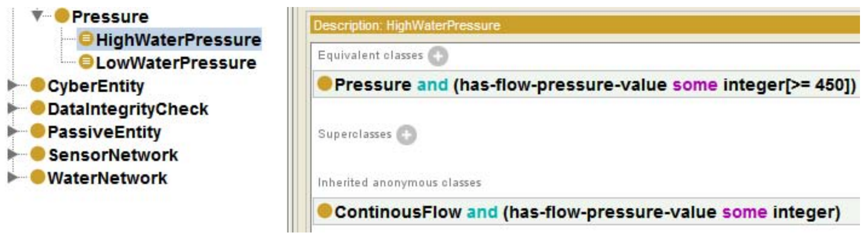

Class hierarchy: Pressure
Thing
IntelligentWaterDistributionNetw
Actuator
ContinousFlow
ChemicalComposition
FlowSpeed
Headloss
Pressure
HighWaterPressure
LowWaterPressure
CyberEntity
DecisionMakingSystem
RecommenderSystem
ExpertiseKnowledgeData
FailureTypeDatabase
MitigationTechniqueData
CommunicationLinkFa
Members list: Commercial-Area-Water-Pressure
Commercial-Area-Water-Pressure
Industrial-Area-Water-Pressure
Residential-Area-Water-Pressure
Suburban-Area-Water-Pressure
Property assertions: Commercial-Area-Water-Pressure
Object property assertions
Data property assertions
has-flow-pressure-value 400
Negative object property assertions
Negative data property assertions

Figure 4.17: Data property assertion for an individual.

We further refine the use of data types by adding restrictions on possible values. We define classes that specify a range of values in which we are interested; for instance, particularly high pressure values that may indicate a failure. For example, in Fig. 4.18, we specify that when the flow pressure is greater than 450 psi, then the pressure is identified as a “HighWaterPressure” class. Similarly, “LowWaterPressure” is specified as “has-flow-pressure-value equal to or lower than 200”.

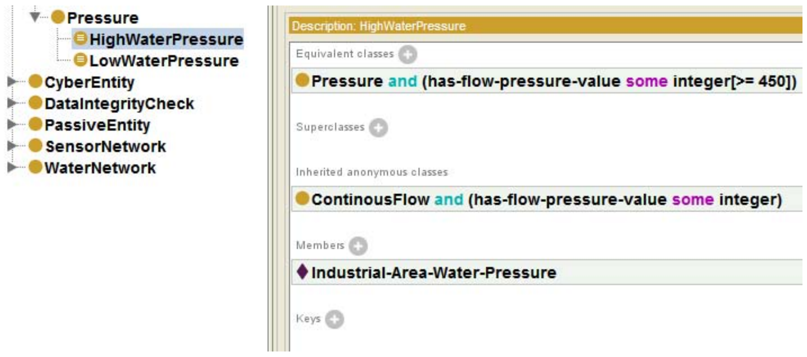


Pressure and (has-flow-pressure-value some integer[>= 450])

ContinousFlow and (has-flow-pressure-value some integer)

Figure 4.18: Definition of "HighWaterPressure."

By automatically reasoning based on the data properties defined for each individual and the range specified for each property, the members (instances) of each subclass can be intelligently added. In Fig. 4.19, the individual "industrial-area-water-pressure" (700) is automatically added as a member of "HighWaterPressure" that is more than 450.



Pressure and (has-flow-pressure-value some integer[>= 450])

ContinousFlow and (has-flow-pressure-value some integer)

Industrial-Area-Water-Pressure

Figure 4.19: Automated classification of an individual.

4.4 VALIDATION OF AUTOMATED FAILURE MITIGATION

In this section, we present and analyze an empirical test case used to validate the automated failure mitigation technique of Section 4. The integrated cyber-physical WDN simulator described in Section 3.5 was utilized. The sensor data was generated by EPANET, which is used to simulate the physical infrastructure. Identification of the failure type is carried out using the ontology model, as is determination of the corresponding failure mitigation technique. The automated reasoning procedure is represented by an OWL-DL script; the reasoned result is converted into a readable .txt format that can be parsed by MATLAB, which simulates the intelligent decision support and determines appropriate settings for physical components. These settings are fed back to EPANET, completing the control cycle.

4.4.1 Initial Configuration and Normal Operation. The topology assumed in EPANET for the physical infrastructure is shown in Fig. 4.20.

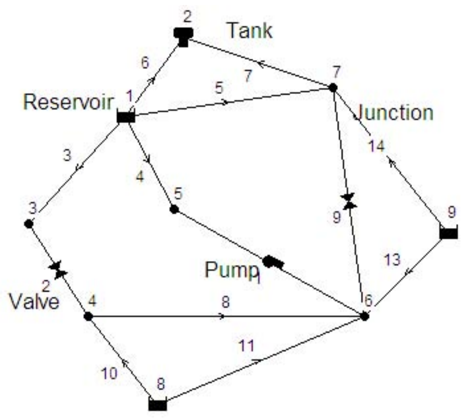


Figure 4.20: Topology assumed for physical infrastructure.

Based on this topology (and the laws of hydraulics), EPANET determined the initial demand (node labels) and flow (link labels) to be as depicted in Fig. 4.21.

The time span and time step of simulation are configured as 24 hours and 1 hour, respectively. Throughout the time span, it is possible to change the settings configured for any component in the physical infrastructure, or for the system as a whole. As an example, it is possible to set the value of "total head," which is the hydraulic head (sum of elevation and pressure head) of water in the reservoir

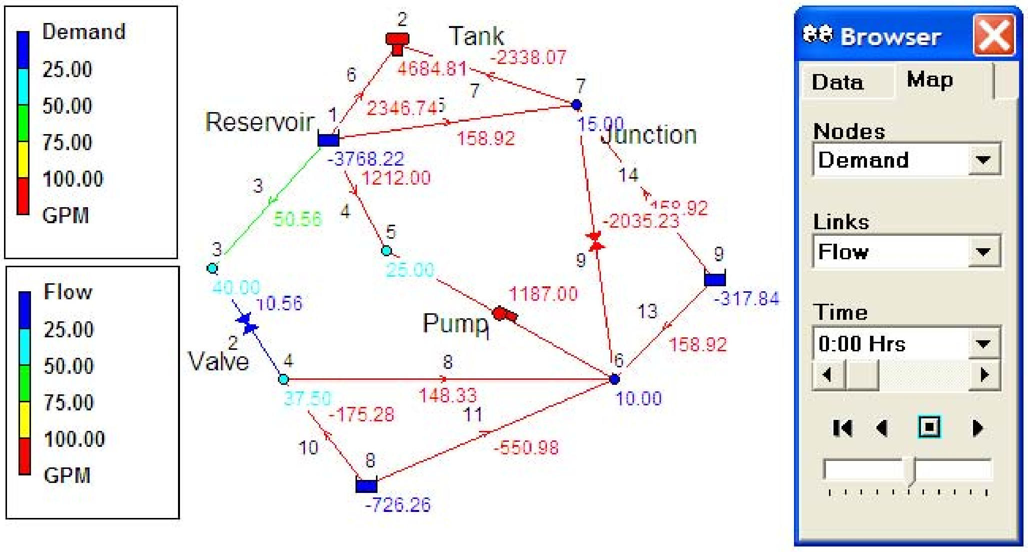
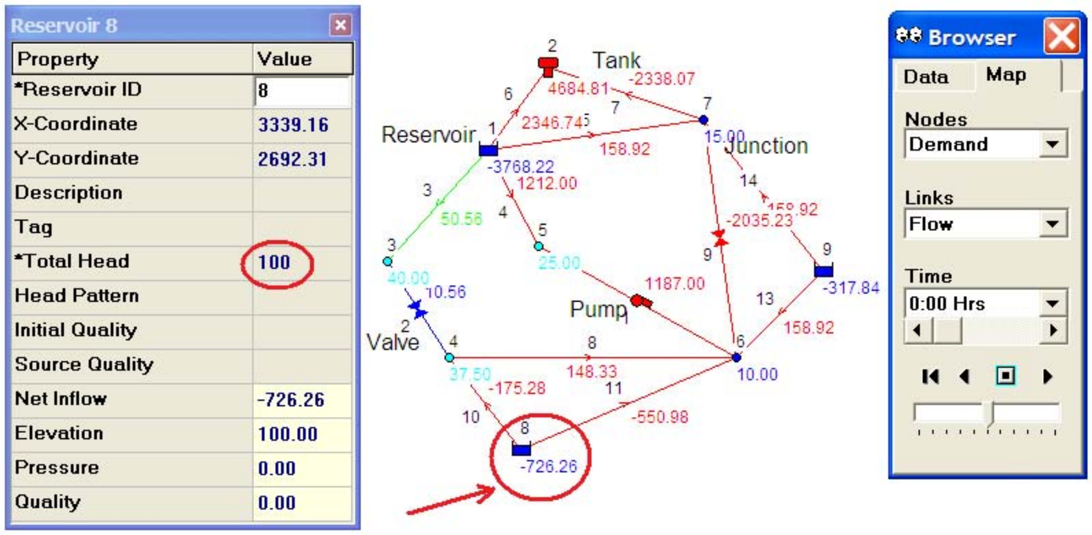


Figure 4.21: Values of demand and flow at time 0.

and a required property for simulation. Fluids possess energy and the total energy associated with a fluid per unit weight of the fluid is denoted as the fluid's "head," which is expressed in units of height. On many occasions, energy needs to be added to a hydraulic system to overcome elevation differences, friction losses, and other minor losses. A pump is a device to which mechanical energy is applied and transferred to the water as total head, therefore it can add more energy to the fluid. When no error occurs in the simulation, the status of the nodes and links in each time span can be displayed in EPANET. From the simulation results, shown in Fig. 4.22, it can be concluded that when the total head is configured as 100 ft, reservoir 8 (node at the bottom of the map) is operating normally at time 0.

The status of an actuator (a pump or valve) in different time slots can also be observed. The pressure values at 0 and 10 hours, respectively, are as shown as node labels in Figs. 4.23 and 4.24.

4.4.2 Failure Scenario and Automated Mitigation. Fault injection was carried out to validate the automated failure mitigation technique. The specific


Reservoir 8

Property	Value
*Reservoir ID	8
X-Coordinate	3339.16
Y-Coordinate	2692.31
Description	
Tag	
*Total Head	100
Head Pattern	
Initial Quality	
Source Quality	
Net Inflow	-726.26
Elevation	100.00
Pressure	0.00
Quality	0.00

Browser
Nodes: Demand
Links: Flow
Time: 0:00 Hrs

Figure 4.22: Status of reservoir 8 at time 0, when total head = 100 ft.

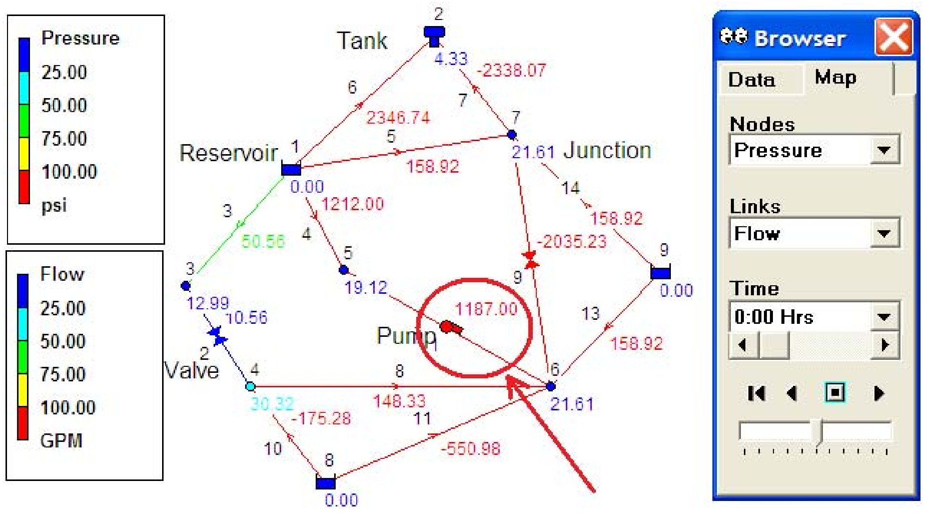

Browser
Nodes: Pressure
Links: Flow
Time: 0:00 Hrs

Figure 4.23: Pressure values (node labels) at time 0.

fault injected was decreasing the total head from 100 to 50 ft, which corresponds to a failure at reservoir 8. This is because the total head is a energy parameter associated with elevation. If the elevation of the reservoir can not sustain the updated total head value, the excessive energy will be distributed to its neighbors. This energy release can lead to excessive flow in neighboring links of the physical infrastructure, including

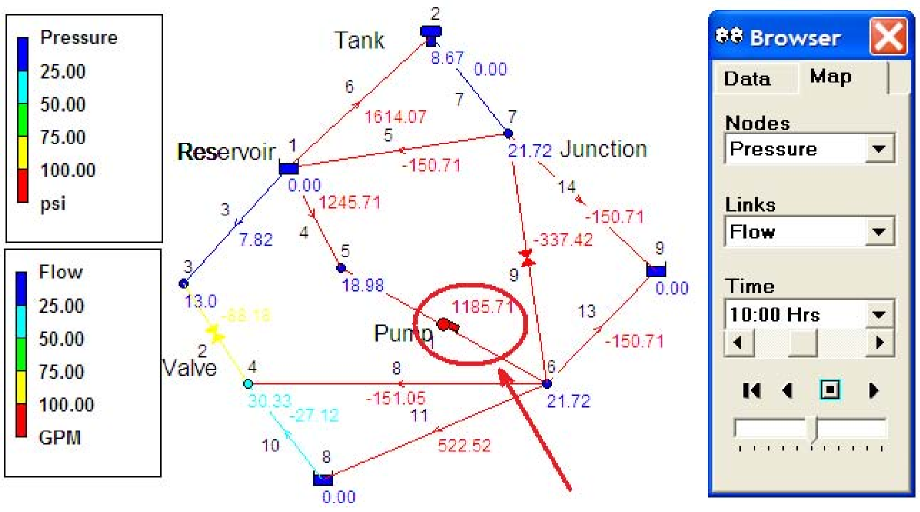
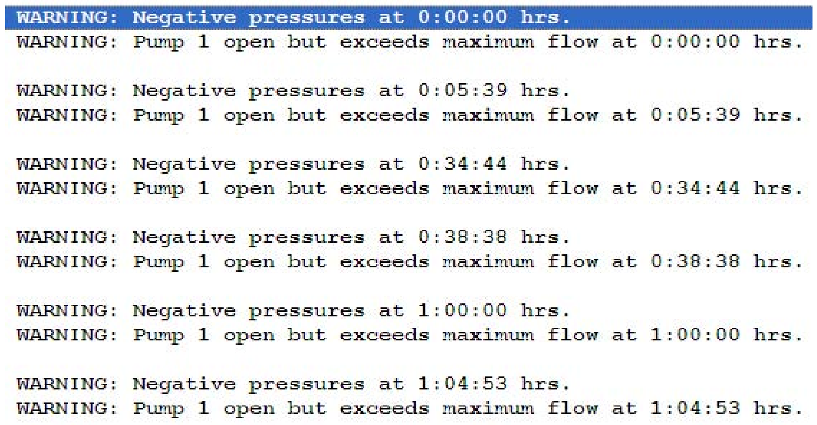


Figure 4.24: Pressure values (node labels) at 10 hours.

both pipes and pumps. EPANET reflects this failure by displaying a warning message with information about overloaded links, as shown in Fig. 4.25. The complete warning message has been omitted in the interest of brevity.



```
WARNING: Negative pressures at 0:00:00 hrs.
WARNING: Pump 1 open but exceeds maximum flow at 0:00:00 hrs.

WARNING: Negative pressures at 0:05:39 hrs.
WARNING: Pump 1 open but exceeds maximum flow at 0:05:39 hrs.

WARNING: Negative pressures at 0:34:44 hrs.
WARNING: Pump 1 open but exceeds maximum flow at 0:34:44 hrs.

WARNING: Negative pressures at 0:38:38 hrs.
WARNING: Pump 1 open but exceeds maximum flow at 0:38:38 hrs.

WARNING: Negative pressures at 1:00:00 hrs.
WARNING: Pump 1 open but exceeds maximum flow at 1:00:00 hrs.

WARNING: Negative pressures at 1:04:53 hrs.
WARNING: Pump 1 open but exceeds maximum flow at 1:04:53 hrs.
```

Figure 4.25: EPANET warning message.

Detection of this failure by EPANET should trigger automated failure mitigation, using the ontologies introduced earlier in this section for identification of the failure type and determination of the appropriate countermeasure. As our fault injection was limited to the physical infrastructure, the countermeasures applied are changes in physical device settings. In the overload scenario described above, the cyber infrastructure determines settings for actuators that regulate the water flow.

Identification of the failure type is the first step in failure mitigation, and takes place based on the warning generated by EPANET. The information embedded in the text file shown in Fig. 4.25 is interpreted to denote a failure caused by "exceeds maximum flow," which corresponds to the failure type of "Exceed Total Head"- a node failure in the failure ontology of Fig. 4.8. The reasoning procedure that leads to this determination is shown in Fig. 4.26.

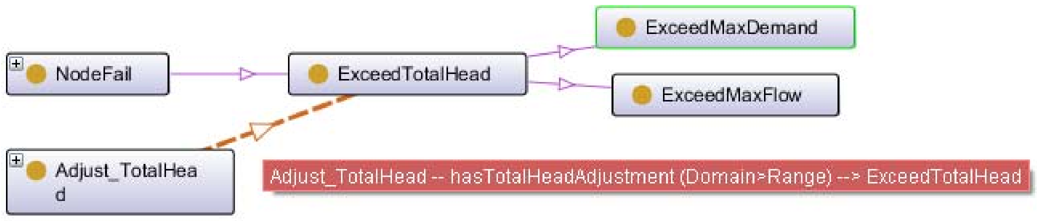
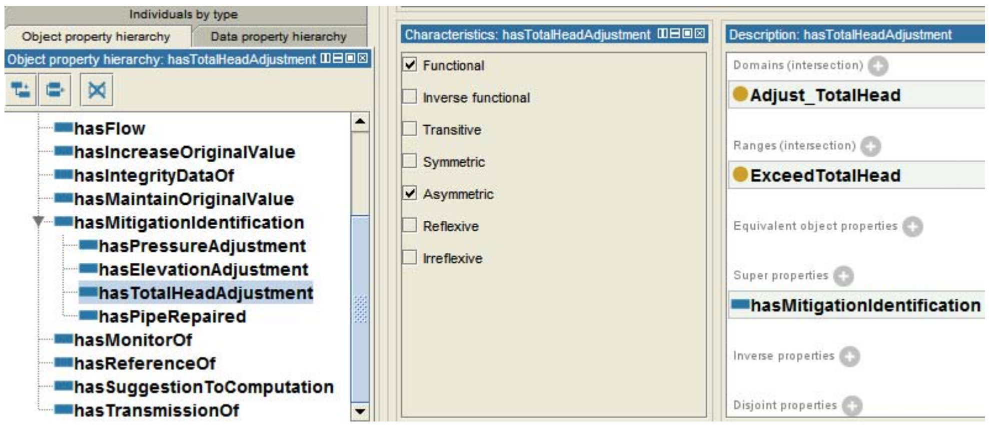

NodeFail
ExceedTotalHead
ExceedMaxDemand
ExceedMaxFlow
Adjust_TotalHead
Adjust_TotalHead -- hasTotalHeadAdjustment (Domain>Range) --> ExceedTotalHead

Figure 4.26: Ontology-based reasoning for failure classification.

Once the failure type is identified, the associated countermeasure is determined using the mitigation ontology of Fig. 4.9. A snapshot of the resulting mitigation object is shown in Fig. 4.27.

The automated reasoning procedure (an OWL-DL script), from "failure identification" to "mitigation technique identification" is depicted in Fig. 4.28.

In our simulator, the selected countermeasures are recorded in an OWL-DL text file, which can be parsed by MATLAB. For the failure scenario described, the "Adjust_TotalHead" countermeasure leads to configuration of the total head at reservoir 8 to 100 ft - the value under normal operating conditions. This value is calculated


Individuals by type
Object property hierarchy
Data property hierarchy
Object property hierarchy: hasTotalHeadAdjustment

hasFlow
hasIncreaseOriginalValue
hasIntegrityDataOf
hasMaintainOriginalValue
hasMitigationIdentification
hasPressureAdjustment
hasElevationAdjustment
hasTotalHeadAdjustment
hasPipeRepaired
hasMonitorOf
hasReferenceOf
hasSuggestionToComputation
hasTransmissionOf

Characteristics: hasTotalHeadAdjustment

- [✓] Functional
- [ ] Inverse functional
- [ ] Transitive
- [ ] Symmetric
- [✓] Asymmetric
- [ ] Reflexive
- [ ] Irreflexive

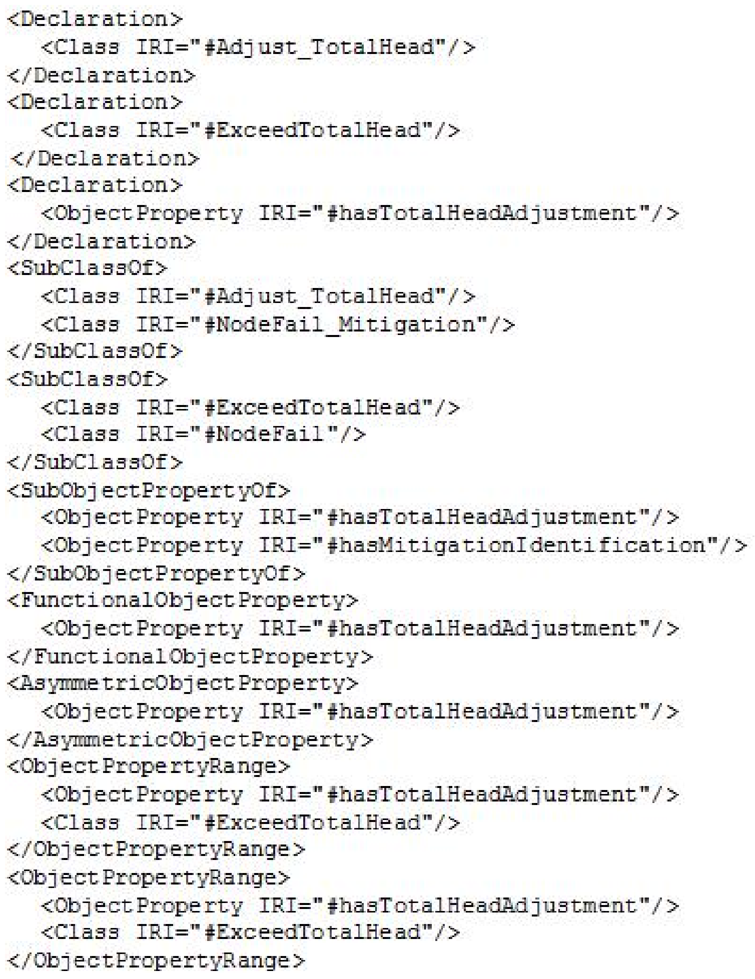
Description: hasTotalHeadAdjustment

Domains (intersection)
Adjust_TotalHead
Ranges (intersection)
ExceedTotalHead
Equivalent object properties
Super properties
hasMitigationIdentification
Inverse properties
Disjoint properties

Figure 4.27: Characteristics and description of mitigation technique.

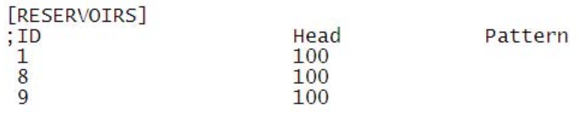
by MATLAB, and sent to EPANET as the input file shown in Fig. 4.29. The countermeasure does not affect the “Pattern” (an option relevant to time specification), and this value is left blank in the automatically-generated file.

Exertion of this countermeasure by EPANET, for a 24-hour simulation with 1-hour time steps (the same parameters as the normal operating case of Fig. 4.21) led to results identical to those depicted in Fig. 4.22, verifying the effectiveness of the countermeasure in restoring normal operation.



```
<Declaration>
   <Class IRI="#Adjust_TotalHead"/>
</Declaration>
<Declaration>
   <Class IRI="#ExceedTotalHead"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#hasTotalHeadAdjustment"/>
</Declaration>
<SubClassOf>
   <Class IRI="#Adjust_TotalHead"/>
   <Class IRI="#NodeFail_Mitigation"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#ExceedTotalHead"/>
   <Class IRI="#NodeFail"/>
</SubClassOf>
<SubObjectPropertyOf>
   <ObjectProperty IRI="#hasTotalHeadAdjustment"/>
   <ObjectProperty IRI="#hasMitigationIdentification"/>
</SubObjectPropertyOf>
<FunctionalObjectProperty>
   <ObjectProperty IRI="#hasTotalHeadAdjustment"/>
</FunctionalObjectProperty>
<AsymmetricObjectProperty>
   <ObjectProperty IRI="#hasTotalHeadAdjustment"/>
</AsymmetricObjectProperty>
<ObjectPropertyRange>
   <ObjectProperty IRI="#hasTotalHeadAdjustment"/>
   <Class IRI="#ExceedTotalHead"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
   <ObjectProperty IRI="#hasTotalHeadAdjustment"/>
   <Class IRI="#ExceedTotalHead"/>
</ObjectPropertyRange>
```

Figure 4.28: Automated reasoning procedure for failure mitigation.



```
[RESERVOIRS]
;ID                 Head           Pattern
 1                  100
 8                  100
 9                  100
```

Figure 4.29: Countermeasures provided by MATLAB to EPANET.

5 GAME-THEORY FOR DECISION SUPPORT IN WDNS

As an illustration of the utility of the tools and techniques proposed in earlier sections of this dissertation, this section describes an agent-based environmental decision support system that utilizes game theory to guide water allocation in a WDN. Interacting agents can elect or decline to serve (provide water to) other agents. Rewards are associated with provision of service, to encourage cooperation by the agents. A model is presented for this service game, and its Nash equilibrium is analyzed. The work presented in this section was published in the Proceedings of the 44th Hawaii International Conference on System Sciences in January 2011 [14].

5.1 MODEL OF THE SERVICE GAME

We define *service* in the model as provision of water to other agents. For simplicity, we assume that an agent can submit only one service request and can accommodate only one service request during a time slot. An agent's status for a given time slot is labeled as $\{Srv\}$ if it fulfills any of the requests received during the time slot. The status of all agents and requests is disseminated throughout the system. The cycle of service request and provision repeats indefinitely, which corresponds to an infinitely repeated game, G^{∞}. The basic game being repeated, G, is defined in terms of the following items:

- Players: all peer agents that participate in water allocation; for tractability, peer agents are assumed to be identical.
- Actions: each agent can decide for or against service provision, denoted as $\{Srv\}$ and $\{Dcln\}$, respectively.
- Preference of each player: represented by the expected value of a payoff function determined by the action taken. When service is received by an agent, the payoff

value of the agent denoted as utility, U; when the agent provides service, the payoff value is denoted as cost, C.

The reputation of a player, i, in a given time slot, t, is denoted by $R(t,i)$, and depends on whether or not it provides service, both in the current time period and in prior periods, as represented by Equation 5.1:

$$R(t,i) = R(t-1,i) * (1-a) + (w * a),\ 0 \le a \le 1, t \ge 2 \tag{5.1}$$

If service is provided by player i in time period t, w is set to 1, otherwise 0. The reputation of all players is initialized as 0 at time $t = 0$, and is defined as w at $t = 1$. Therefore, $0 \le R(t,i) \le 1$ is always maintained. In Equation 5.1, parameter a is a constant that captures the strength of the "memory of the system," i.e., the relative importance of current vs. past behavior of an agent in determining its reputation.

5.2 NASH EQUILIBRIUM OF THE GAME

In the game, each player wants to gain the maximum benefit, leading to a non-cooperative game. Nash equilibrium is reached when competition ends among the players. This occurs when the collective set of actions taken by the players with respect to service provision is locally optimum, i.e., no player can improve its utility by electing a different strategy. The two types of Nash equilibria are *pure* and *mixed*. Pure Nash equilibrium results when every player declines to serve, leading to a trivial scenario that is not a sustainable operational state for a WDN. The mixed Nash equilibrium, where players elect to serve in some time periods and decline service in others, is the focus of our investigation.

In the mixed-strategy symmetric Nash equilibrium action profile, each player, i, elects to serve with probability p and declines service with probability $1-p$, with

$p > 0$, meaning that either action is possible. We assume that each player can provide service prior to requesting it.

The expected payoff value of electing to serve during period t is defined as:

$$\text{Payoff}(Srv) = p * (-C + R(t, Srv) * U) \tag{5.2}$$

In Equation 5.2, the term $(-C+R(t, Srv)*U)$ illustrates the tradeoff inherent to service provision, namely, that cost of providing service as compared to the benefit of receiving service. The term $R(t, Srv)*U$ reiterates that the probability of obtaining service in the current time period depends on a player's reputation. This payoff value of a player not only reflects its current payoff after providing service, but also captures the potential to obtain service in the next period, through the inclusion of $R(t, Srv)$, which can be used as a health indicator that reflects the capability of the player to gain service in the near future. When service is provided, $w = 1$; per Equation 5.1:

$$R(t, i) = R(t-1, i) * (1-a) + a \tag{5.3}$$

Similarly, the payoff value of selecting the action $\{Dcln\}$ is:

$$\text{Payoff}(Dcln) = (1-p) * (R(t, Dcln) * U) \tag{5.4}$$

The equation reflects the "no contribution, no cost" case. When service is declined, $w = 0$, and per Equation 5.1:

$$R(t, i) = R(t-1, i) * (1-a) \tag{5.5}$$

In a mixed-strategy Nash equilibrium of finite games, each player's expected payoff should be the same for all actions. In other words, the respective payoff values for $\{Srv\}$ and {Dcln} are equal:

$$\text{Payoff}(Srv) = \text{Payoff}(Dcln) \tag{5.6}$$

Substituting from Equations 5.2 and 5.4 yields:

$$p * (-C + R(t, Srv) * U) = (1 - p) * (R(t, Dcln) * U) \tag{5.7}$$

Incorporating the iterative definition of reputation, from Equations 5.3 and 5.5, the probability of service provision, p, is determined as:

$$p = \frac{R(t-1) * U(1-a)}{-C + 2R(t-1) * U(1-a) + Ua} \tag{5.8}$$

5.3 C++ IMPLEMENTATION

We implemented the agent-based game-theoretic decision support system in C++. The class diagram of the prototype is depicted in Fig. 5.1.

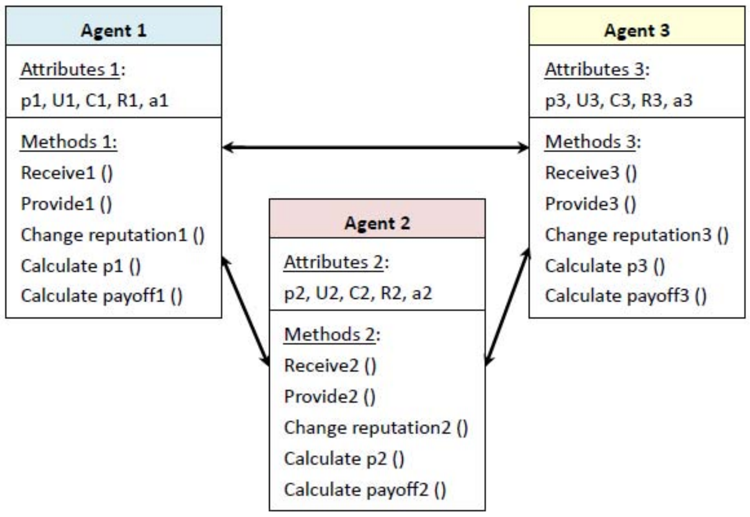


Figure 5.1: Class diagram of C++ implementation of agent-based decision support.

The software code for Agent 1, as an example, is shown in Fig. 5.2.

```
Agent 1:
Attributes 1:
{
    string actions ({serve}, {decline});
    int time interval t = 0;
    int utility value U;                    //U can be initialized as 800
    int cost value C;                       //C can be initialized as 10
    double reputation of the node R;        //R is constrained to (0,1)
    double parameter a = 0.2;               //a is constrained to (0,1)
    double payoff{serve};
    double payoff{decline};
}

Methods1:
{
    (for t = 1; t<=10; t++;)
    Receive request from agent 2 ({serve2} =1); //store the {serve} request from
    Receive request from agent 3 ({serve3} =1);  //different agents for further service provision
    Provide service to agent 2 ();
    Provide service to agent 3 ();
    Increment own reputation (double R)
    {
        R (t) = R(t-1)*(1-a)+w*a;                  //store the current reputation value for further computing
    };
    Calculate service probability p for next time interval (double p)
    {
        p = (R*(t-1)*U*(1-a))/(-C+2*R*(t-1)*U*(1-a)+U*a)
    };
    Calculate payoff value (double payoff)
    {
        if {serve} = 1 then                     // case that agent 1 serves other agents
        payoff {serve} = p*(-C+R(t)*U);
        return (payoff{serve});
        else then                               // case that agent 1 declines to serve
        payoff{decline} = (1-p)*(R(t)*U);
        return (payoff{decline});
    }
}
```

Figure 5.2: C++ code for Agent 1.

5.4 VALIDATION OF AGENT OPERATION

Experimental validation of the game-theoretic approach to water allocation was carried out through MATLAB simulation of the three interacting peer agents shown in Fig. 5.3. In this section, we are validating the operation of the cyber infrastructure (agents) only. Cyber-physical validation using our integrated simulator is presented in Section 5.5.

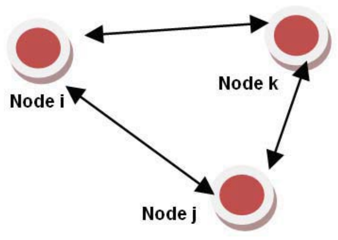


Figure 5.3: Interaction among three peer agents.

The agents are labeled Node i, Node j, and Node k, respectively. For each agent, the service strategy is as shown in Table 5.1. The strategy shown in Table 5.1 does not exhaustively capture all actions that could be taken by the three agents, but it provides a representative set of actions over a non-trivial duration of ten time slots.

Table 5.1: Strategy for service game.

Time t	Node i	Node j	Node k
1	Serve j	Serve k	Decline
2	Decline	Serve i	Decline
3	Serve k	Decline	Decline
4	Decline	Decline	Serve i
5	Serve k	Decline	Serve i
6	Serve j	Decline	Serve i
7	Serve j	Serve i	Decline
8	Decline	Decline	Decline
9	Decline	Decline	Serve j
10	Serve k	Serve i	Decline

According to Table 5.1, we can summarize the strategy of each player, i, as W_i below:

- $W_i = [1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1]$
- $W_j = [1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1]$
- $W_k = [0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0]$

5.4.1 Agent Reputation over Time. Firstly, based on the service strategy, we investigate how an agent's reputation varies in the 10 time slots by following the equilibrium strategy described in Section 5.2. Figure 5.4 depicts the changes in $R(t)$ for each of the three agents.

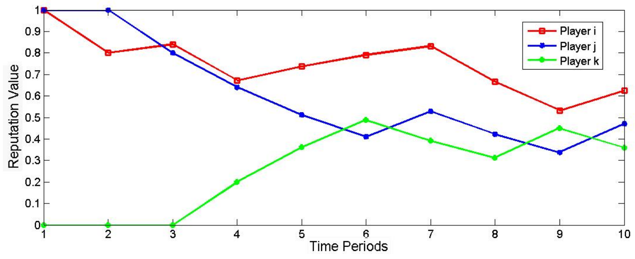


Figure 5.4: Change in agent reputation over time.

According to the strategies W_i, W_j, and W_k, respectively, the results show that the reputation value increases when the service is provided by a particular agent and decreases when the agent provides no service at all (or accepts service from its peers). The equilibrium strategy maintains stability - over the 10 time intervals, the three peer agents' action, i.e., providing service or accepting service, will be similar to each other. No one agent can be constantly acquiring or contributing service.

5.4.2 Probability of Service Provision over Time. Secondly, we investigate how the probability of service provision varies over time. Simulation results are shown in Fig. 5.5.

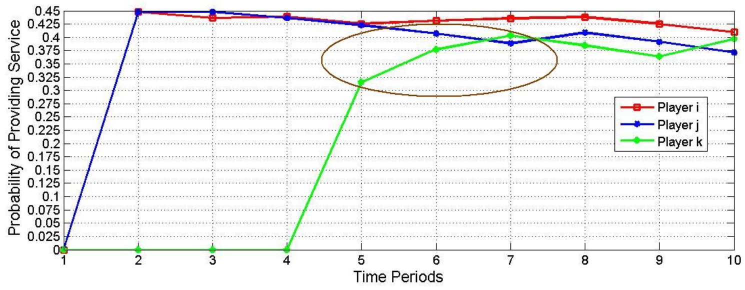


Figure 5.5: Probability of service provision.

According to Equation 5.8, p varies in each time interval depending on the agent's reputation at the end of the previous time interval. As U, C and a remain constant in this scenario, the agent's reputation at the end of the previous time interval is determined by its strategy, W. If during the previous time interval, the agent provided service, then in the next time interval, the probability of service provision by the agent increases. If the agent did not provide service in the previous time interval, then the probability of providing service decreases.

Another observation is that if an agent continuously provides service to its peers, then the increase in probability of service provision within each interval, compared with the previous interval, will actually decrease (indicated as the circle in the figure for player k). This again demonstrates the role of the equilibrium strategy in the resource allocation (service provision or acquisition), i.e., to restrain an agent that is constantly providing or constantly obtaining service.

5.4.3 Steady-State Behavior. An interesting question is whether steady-state behavior of the water allocation game will settle on the service provision probability, p, of 0.5. Figures 5.6 through 5.11 provide valuable insight.

Figure 5.6 illustrates the case for player i, where $U/C = 80$, $a = 0.2$, and the strategy for 100 time slots consists of repeating [1 0 1 0 1 1 1 0 0 1] [1 1 0 0 0 0 1 0 0 1] [0 0 0 1 1 1 0 0 1 0]. The maximum value reached by p is 0.45.

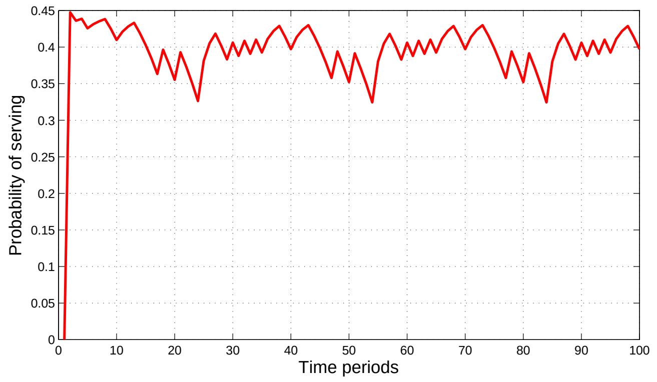


Figure 5.6: Probability of service provision, diverse strategy.

Keeping $U/C = 80$, $a = 0.2$, but changing the strategy of agent i to 100 time slots of repeating [1 0 1 0 1 1 1 0 0 1], the simulation result is as depicted in Fig. 5.7. Similar to the case above, p barely reaches 0.45.

In the third test case, we still have $U/C = 80$ and $a = 0.2$, but the strategy of agent i remains a constant 1 over 80% of the simulation time, which means that agent i is serving most of the time. The simulation result shown in Fig. 5.8 illustrates that p still does not reach 0.5.

In yet another experiment, we maintained $U/C = 80$, but changed a to 0.01, corresponding to a system with a good memory (past actions of a player have strong bearing over its reputation). With near-continuous service provision, the simulation

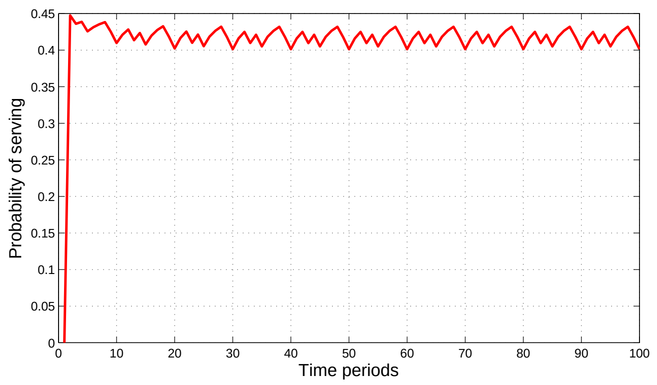


Figure 5.7: Probability of service provision, monotonous strategy.

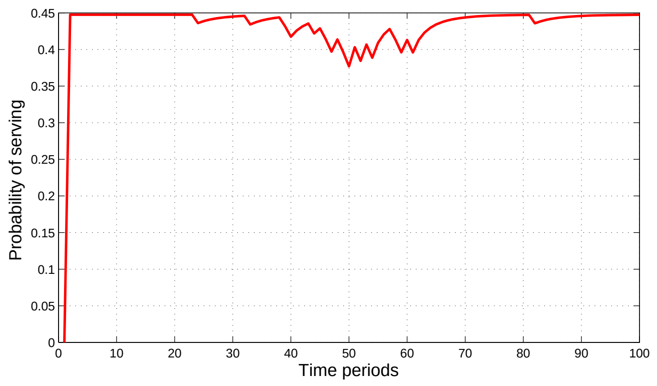


Figure 5.8: Probability of service provision, near-continual service.

results were as depicted in Fig. 5.9, where unlike the aforementioned cases, p reaches 0.5. Considering the factors that can affect p in equation 5.8, we investigated the situation where the U/C ratio is changed to 8 and a remains the same, the simulation result in Fig. 5.10 shows that p can reach 0.5.

Finally, we investigate the case that the agent is initially requesting service, i.e., W is 1 for the first third of the simulation time. The simulation results are depicted in Fig. 5.11. Simulation results show that regardless of variations in U/C

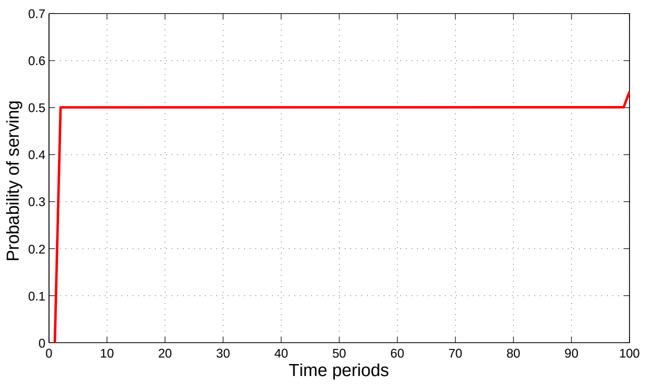


Figure 5.9: Probability of service provision, near-continual service, strong memory, and $U/C = 8$.

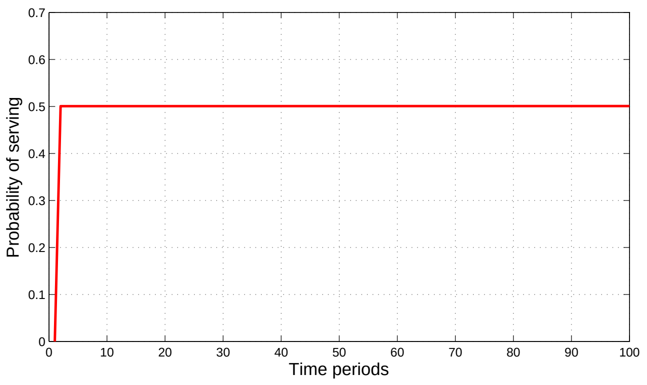


Figure 5.10: Probability of service provision, near-continual service, strong memory, and $U/C = 80$.

and service strategy, when time goes to infinity, the value of p is mainly determined by the constant a, which reflects the importance of serving during the current period.

5.4.4 Reputation vs. Service Behavior. In addition to the probability of service provision, p, we also investigate how the agent reputation, $R(t)$, changes with variations in the strategy, W. Simulation results are shown in Fig. 5.12, which illustrates that the collaborative decision making strategy discourages continual provision of service by an agent.

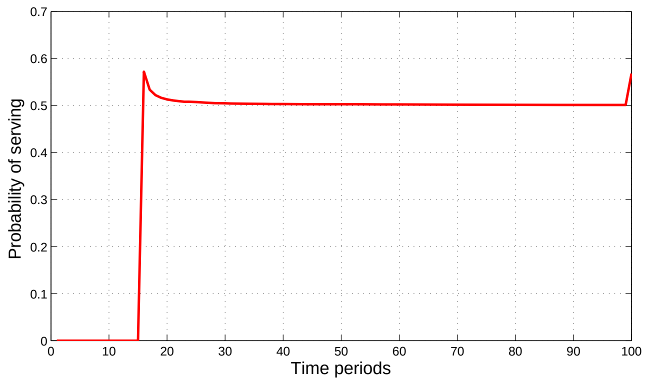


Figure 5.11: Probability of service provision, initial continual service request.

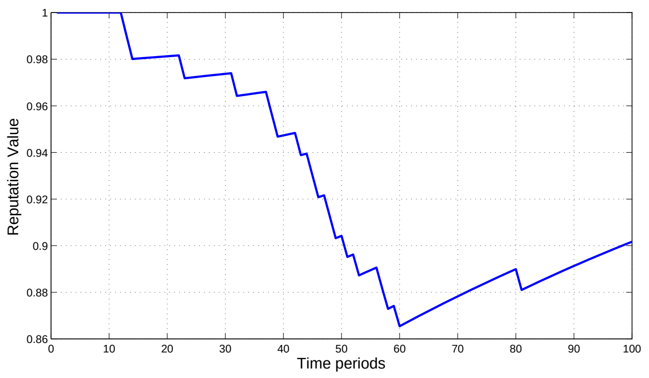


Figure 5.12: Reputation of agent with near-continual service.

5.4.5 Reputation vs. Memory. For additional insight into the effect of various parameters on the system operation, we vary a to be 0.2, 0.4, 0.6 and 0.8, respectively to see how the reputation changes as a changes. Figure 5.13 shows the variations in the reputation of agent i for different values of a. The same strategy, $W = [1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1]$, is used for the four groups of data.

As seen in Fig. 5.13, a larger a will cause a more drastic change in the reputation during each time interval. This is because the constant a plays a major role in

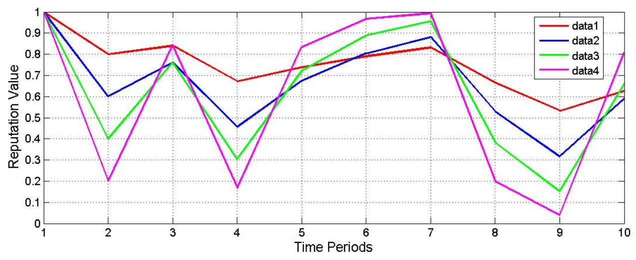


Figure 5.13: Effect of a on agent reputation.

terms of deciding the importance of service provision in determining the reputation, $R(t, i)$, of an agent (see Equation 5.1).

5.4.6 Accumulated Payoff vs. Service Behavior. The simulation results depicted in Figs. 5.14 to 5.17 show that the accumulated payoff values for $\{Srv\}$ and $\{Dcln\}$, respectively, are quite close, regardless of changes in a. The service ratio, defined as ratio of the accumulated payoff value for $\{Srv\}$ to the accumulated payoff value for $\{Dcln\}$, is 1.07, 0.95, 0.84 and 0.74, respectively, for $a = 0.2$, 0.4, 0.6, and 0.8.

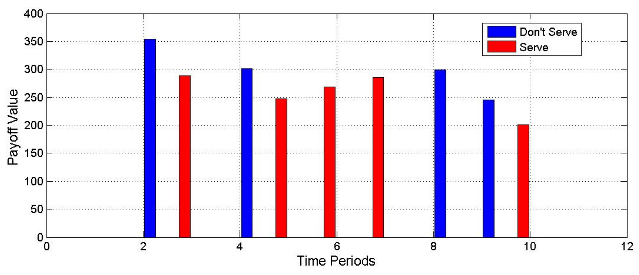


Figure 5.14: Payoff value, $a = 0.2$.

As depicted in Fig. 5.18, the variation in service ratio decreases as a increases.

As seen in Equation 5.1, as a increases, an agent's reputation in the previous time interval plays a less important role in determining the agent's current reputation,

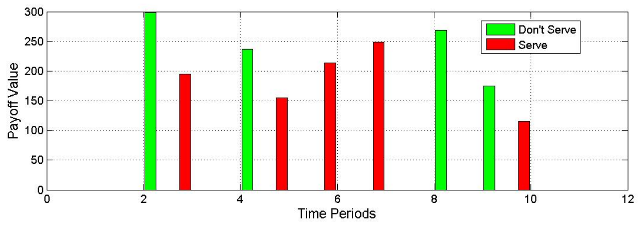


Figure 5.15: Payoff value, $a = 0.4$.

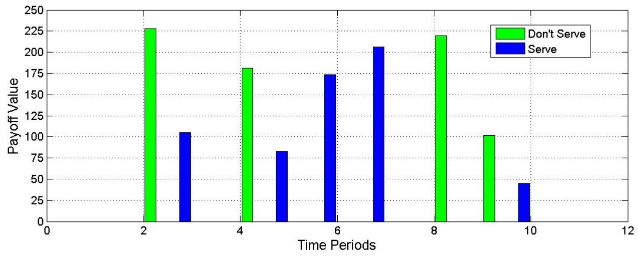


Figure 5.16: Payoff value, $a = 0.6$.

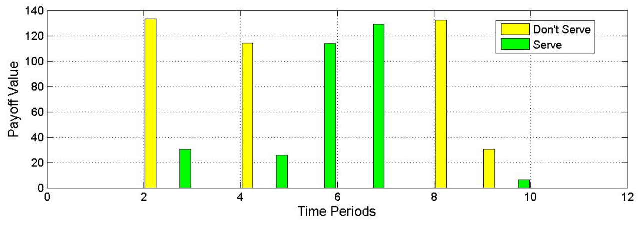


Figure 5.17: Payoff value, $a = 0.8$.

i.e., the system's memory becomes weaker. As an agent's reputation increases only when it provides service to its peers, an increased a can discourage the agent from contributing service to the system, and hence lead to a decrease in the $\{Srv\}/\{Dcln\}$ ratio.

5.4.7 Accumulated Payoff vs. U/C. We also investigate how the payoff value varies as U/C varies. Figures 5.19 through 5.23 illustrate the payoff value for each time period, for different U/C ratios.

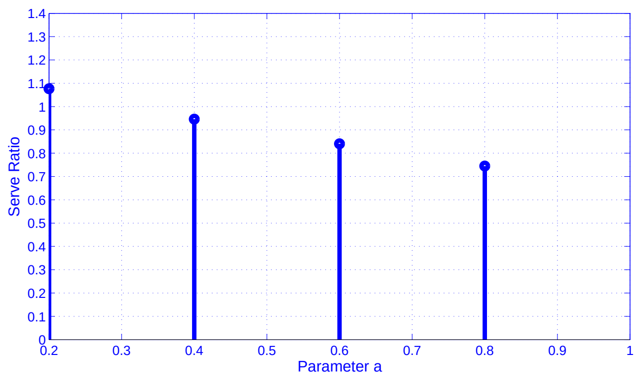


Figure 5.18: Service ratio $\{Srv\}/\{Dcln\}$ vs. a.

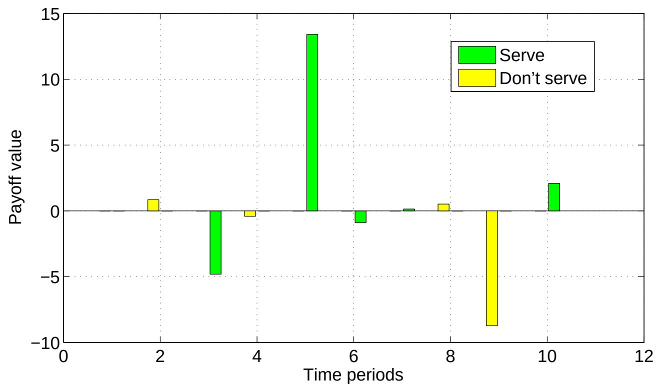


Figure 5.19: Payoff value, $U/C = 1.1$.

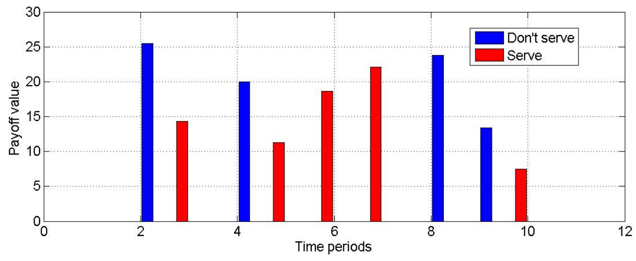


Figure 5.20: Payoff value, $U/C = 8$.

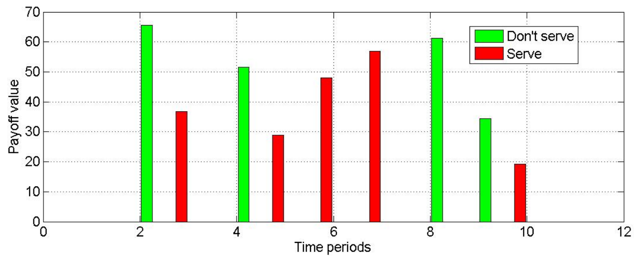


Figure 5.21: Payoff value, $U/C = 20$.

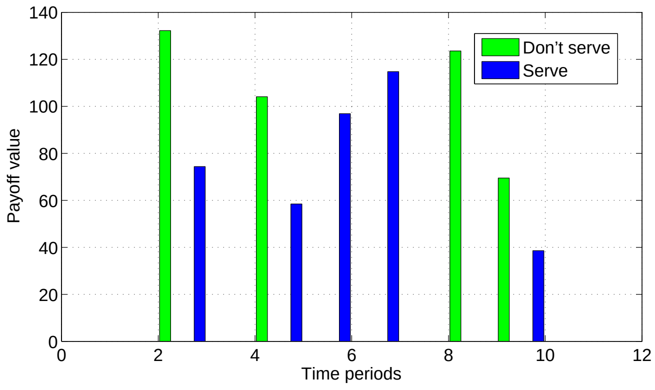


Figure 5.22: Payoff value, $U/C = 40$.

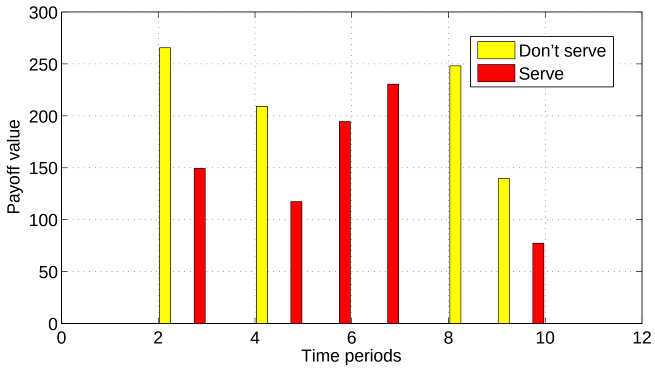


Figure 5.23: Payoff value, $U/C = 80$.

The simulation results are summarized in Table. 5.2. The variation of U/C from 8 to 80 leads to very little change in the service ratio, $\{Srv\}/\{Dcln\}$; however, when U and C are nearly equal, as in they are in Fig. 5.19, the payoff value, and hence the service ratio may become negative. If $U/C = 1$, the payoff value explodes to ∞.

Table 5.2: Service ratio for different U/C values.

U/C	Service ratio
1.1	-1.27
8	0.89
20	0.89
40	0.89
80	0.89

5.4.8 Service Probability and Payoff vs. *U/C*. Finally, we investigate the effect of U/C on the probability of service provision and the payoff. Figure 5.24 depicts the probability of service provision for values of U/C ranging from 40 to 120. Figure 5.25 provides a more detailed view of the same simulation results.

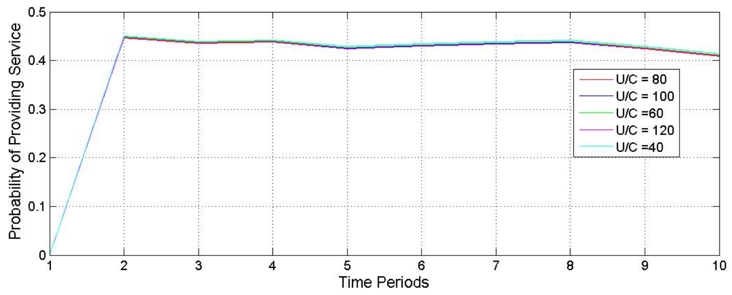


Figure 5.24: Effect of U/C on p.

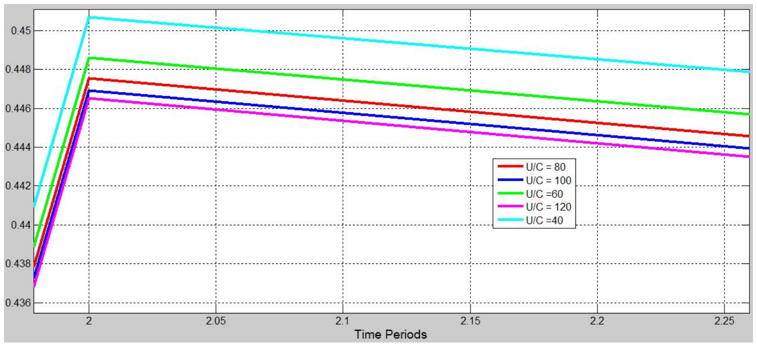


Figure 5.25: Detailed view of Figure 5.24

Dividing the numerator and denominator of Equation 5.8 by C yields:

$$p = \frac{R(t-1)*(U/C)(1-a)}{(-1+2R(t-1)*(U/C)(1-a)+(U/C)a} \tag{5.9}$$

Figure 5.25 illustrates that the lower the U/C ratio, the higher the probability to of service provision. With a low U/C ratio, an agent will achieve a lower payoff value if it obtains service, as compared to the alternative action of providing service. Therefore, a lower U/C ratio can encourage agents to provide service rather than obtain or decline service. However, the impact of U/C on the probability of service provision appears to be quite minor.

Figure 5.26 depicts the effect of U/C on the payoff value. As seen in the figure, the higher the U/C ratio, the higher the payoff value will be within each time interval. According to the expected payoff function for an agent in time period t, if an agent provides service, the payoff value will be negative. The strategy represented is $W = [1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1]$. As depicted in the center of the figure, service provision can help to increase the payoff value.

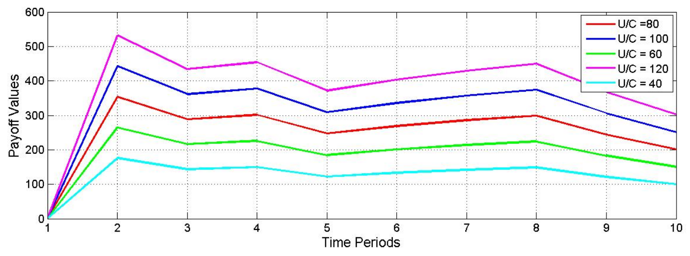


Figure 5.26: Effect of U/C on payoff value.

5.5 CYBER-PHYSICAL VALIDATION OF DECISION SUPPORT

In this section, we present results of using the integrated cyber-physical simulator to validate the environmental decision support system of Section 5. This is an effort to reflect dynamic behavior of the WDN and reveal interdependencies across the cyber-physical boundary. The work presented in this section will appear as a book chapter in late 2011 [16].

5.5.1 Topology for Integrated Simulation. The topology that we utilize for validating the game-theoretical decision support is identical to that of Fig. 4.20. Three agents were assumed to control the WDN, as shown in Fig. 5.27; where reservoir 1, tank 2, junction 5 and 7, pump 1 are in the same group; reservoir 8, valve 2, junction 3 and 4 are in the same group; and reservoir 9, junction 6 and valve 9 are in the same group. A single actuator controls the physical components within the scope of each agent.

5.5.2 Initial Configuration. Simulation results of node demand and link flow, respectively; at the first hour (with a simulation time step of one hour) are summarized in Fig. 5.28 and Fig. 5.29.

From Fig 5.28, we can infer that at 1:00 hour, reservoir 1 is providing water (indicated by its negative demand value) and reservoirs 8 and 9 are retrieving water

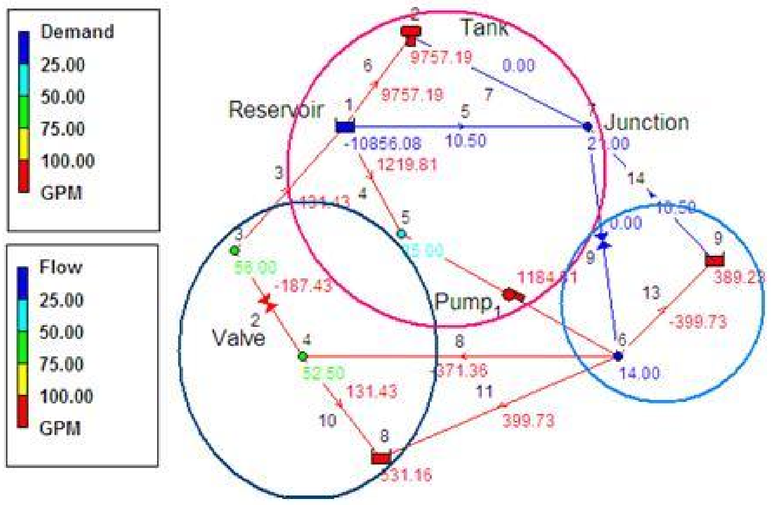
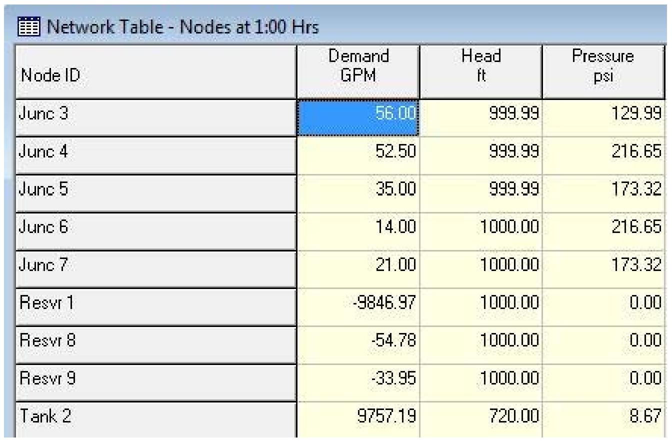


Figure 5.27: Scope of each of three agents.

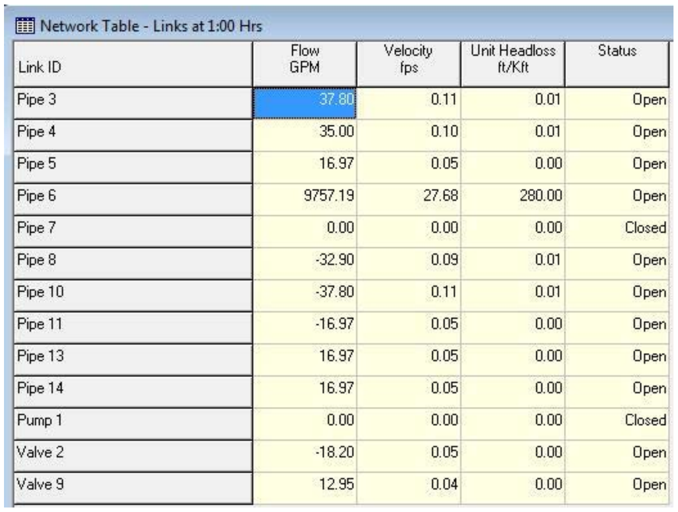


Network Table - Nodes at 1:00 Hrs

Node ID	Demand GPM	Head ft	Pressure psi
Junc 3	56.00	999.99	129.99
Junc 4	52.50	999.99	216.65
Junc 5	35.00	999.99	173.32
Junc 6	14.00	1000.00	216.65
Junc 7	21.00	1000.00	173.32
Resvr 1	-9846.97	1000.00	0.00
Resvr 8	-54.78	1000.00	0.00
Resvr 9	-33.95	1000.00	0.00
Tank 2	9757.19	720.00	8.67

Figure 5.28: Node demand (in gpm) at 1:00 hour.

(indicated by their respective positive demand values). Similar to the (purely cyber) validation of agent operation in Section 5.4, we use 1 and 0 to denote an agent that is serving and declining to serve water, respectively. Accordingly, in the first simulation



Network Table - Links at 1:00 Hrs

Link ID	Flow GPM	Velocity fps	Unit Headloss ft/Kft	Status
Pipe 3	37.80	0.11	0.01	Open
Pipe 4	35.00	0.10	0.01	Open
Pipe 5	16.97	0.05	0.00	Open
Pipe 6	9757.19	27.68	280.00	Open
Pipe 7	0.00	0.00	0.00	Closed
Pipe 8	-32.90	0.09	0.01	Open
Pipe 10	-37.80	0.11	0.01	Open
Pipe 11	-16.97	0.05	0.00	Open
Pipe 13	16.97	0.05	0.00	Open
Pipe 14	16.97	0.05	0.00	Open
Pump 1	0.00	0.00	0.00	Closed
Valve 2	-18.20	0.05	0.00	Open
Valve 9	12.95	0.04	0.00	Open

Figure 5.29: Link flow (in gpm) at 1:00 hour.

period, the collective strategy of the three agents is (1, 0, 0). In the topology of Fig. 5.3, we assume that reservoirs 1, 8, and 9 are nodes i, j and k, respectively.

Water attributes; e.g., demand, pressure, head, and flow, are controlled in EPANET by actuators (pumps and valves). By sending a control command to the actuator from the cyber infrastructure (implemented in MATLAB), we can configure the operation of the node (reservoir) as “serve” or “decline”. As each of the three actuators in Fig. 5.27 can be either open or closed, the eight combinations of Table 5.3 result.

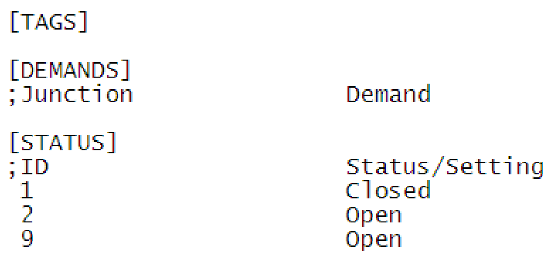
Table 5.3 shows that two of the eight (actuator) configurations can lead to failure. In other words, EPANET cannot continue simulation if pump 1, valve 2 and valve 9 are either (open, open, open) or (open, closed, open), respectively. This is an instance of failure propagation from the cyber to the physical infrastructure.

In Table 5.3, three water provision strategies are repeated: (1, 0, 0), (1, 1, 1) and (1, 0, 1). In other words, if the initial strategy is (1, 0 ,0), we configure the subsequent strategy to be (1, 1, 1). Multiple actuator settings can be used to achieve this strategy; for this case, we select the combination of (closed, open, open)

Table 5.3: Result at time 0 with different configurations of actuators.

Reservoir	1	8	9	Result at 0:00 hr
Actuator	Pump1	Valve 2	Valve 9	
Status of Actuator	Open	Open	Open	Error. Pump 1 opens but exceeds max flow at 0:00 hr.
	Open	Open	Closed	Reservoir 1 is serving, others are not. [1,0,0].
	Open	Closed	Open	Error. Pump 1 opens but exceeds max flow at 0:00 hr.
	Closed	Open	Open	All three reservoirs are serving at 0:00 hr [1,1,1].
	Closed	Closed	Open	All three reservoirs are serving at 0:00 hr [1,1,1].
	Closed	Open	Closed	All three reservoirs are serving at 0:00 hr [1,1,1].
	Open	Closed	Closed	Reservoirs 1 and 9 are serving, reservoir 8 is not [1,0,1].
	Closed	Closed	Closed	All three reservoirs are serving at 0:00 hr [1,1,1].

for pump 1, valve 2 and valve 9, respectively. The remainder of the settings remain unchanged from the initial configuration. The control command file generated by MATLAB (input .INP file to EPANET) is shown in Fig. 5.30, which reflects the actuator configurations. As shown in the .INP file, the three actuators are configured as (closed, open, open).



```
[TAGS]

[DEMANDS]
;Junction                Demand

[STATUS]
;ID                      Status/Setting
 1                       Closed
 2                       Open
 9                       Open
```

Figure 5.30: Actuator settings recommended to EPANET.

5.5.3 Results and Analysis. Figure 5.31 shows simulation results for when the actuators are configured as (closed, open, open) - the scenario that all reservoirs are serving. In Fig. 5.31, the serving reservoirs are represented in blue and have a negative demand value (in gpm).

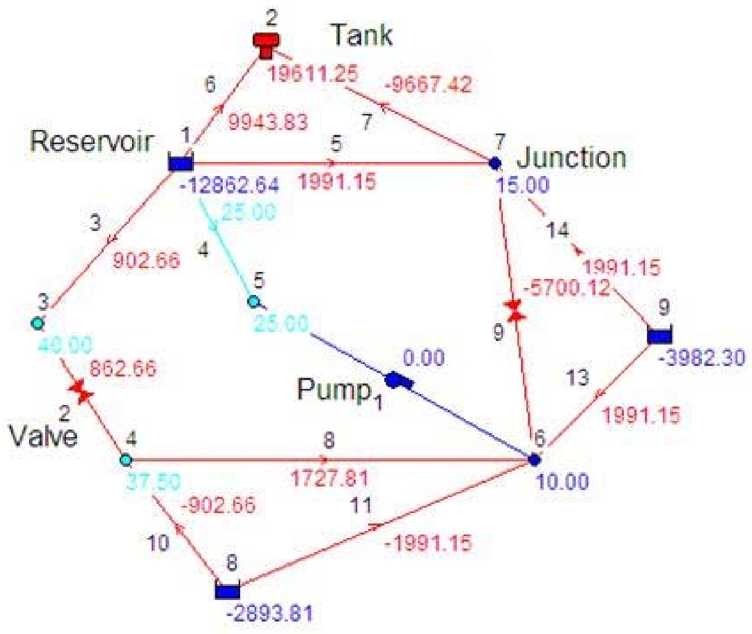
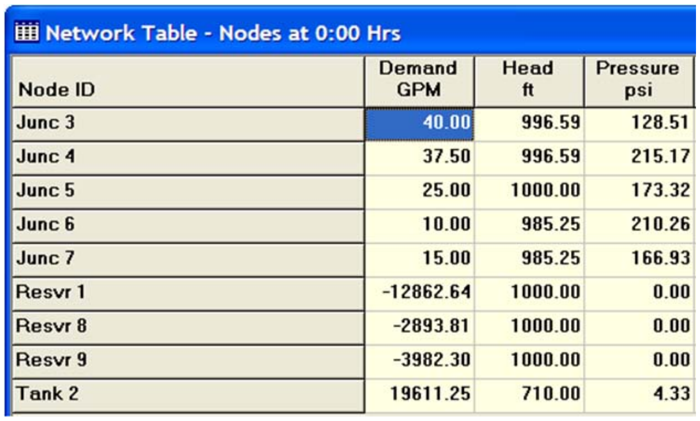


Figure 5.31: Results of applying the recommended actuator settings.

The node demand and link flow at time 0, respectively; resulting from application of the recommended actuator settings are shown in Figs. 5.32 and 5.33.

We further investigate the case that three reservoirs are consistently (throughout the 10 simulation periods) providing water. Figure 5.34 depicts the demand value of each reservoir for each simulation period.

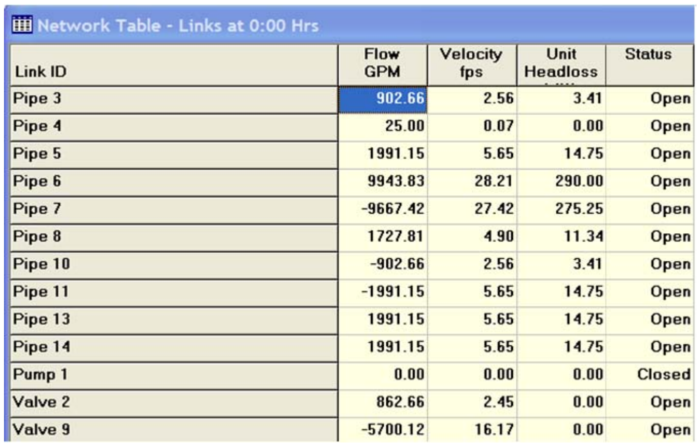
Given the initial configuration, EPANET (which reflects operation of the physical infrastructure) can operate successfully. At time 0, all reservoirs are providing water, but the water quantity provided by reservoir 1 is much higher than that of



Network Table - Nodes at 0:00 Hrs

Node ID	Demand GPM	Head ft	Pressure psi
Junc 3	40.00	996.59	128.51
Junc 4	37.50	996.59	215.17
Junc 5	25.00	1000.00	173.32
Junc 6	10.00	985.25	210.26
Junc 7	15.00	985.25	166.93
Resvr 1	-12862.64	1000.00	0.00
Resvr 8	-2893.81	1000.00	0.00
Resvr 9	-3982.30	1000.00	0.00
Tank 2	19611.25	710.00	4.33

Figure 5.32: Node demand at time 0 when all reservoirs are serving.



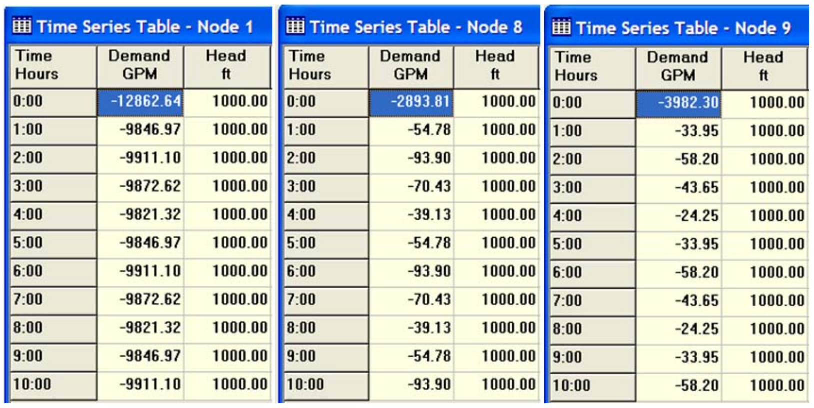
Network Table - Links at 0:00 Hrs

Link ID	Flow GPM	Velocity fps	Unit Headloss	Status
Pipe 3	902.66	2.56	3.41	Open
Pipe 4	25.00	0.07	0.00	Open
Pipe 5	1991.15	5.65	14.75	Open
Pipe 6	9943.83	28.21	290.00	Open
Pipe 7	-9667.42	27.42	275.25	Open
Pipe 8	1727.81	4.90	11.34	Open
Pipe 10	-902.66	2.56	3.41	Open
Pipe 11	-1991.15	5.65	14.75	Open
Pipe 13	1991.15	5.65	14.75	Open
Pipe 14	1991.15	5.65	14.75	Open
Pump 1	0.00	0.00	0.00	Closed
Valve 2	862.66	2.45	0.00	Open
Valve 9	-5700.12	16.17	0.00	Open

Figure 5.33: Link flow at time 0 when all reservoirs are serving.

reservoirs 8 or 9. From 1:00 hour onwards, the water quantity provided by each of reservoirs 8 and 9 has dramatically decreased, as the bulk of the water is provided by reservoir 1.

Cyber-physical simulation of the decision support system demonstrates interdependencies that can lead to fault propagation between the cyber and physical infrastructures. Incorrect configuration of the actuators may cause failures, as shown in Table 5.3; due to constraints on the operation of physical components. The simulation


Time Series Table - Node 1

Time Hours	Demand GPM	Head ft
0:00	-12862.64	1000.00
1:00	-9846.97	1000.00
2:00	-9911.10	1000.00
3:00	-9872.62	1000.00
4:00	-9821.32	1000.00
5:00	-9846.97	1000.00
6:00	-9911.10	1000.00
7:00	-9872.62	1000.00
8:00	-9821.32	1000.00
9:00	-9846.97	1000.00
10:00	-9911.10	1000.00

Time Series Table - Node 8

Time Hours	Demand GPM	Head ft
0:00	-2893.81	1000.00
1:00	-54.78	1000.00
2:00	-93.90	1000.00
3:00	-70.43	1000.00
4:00	-39.13	1000.00
5:00	-54.78	1000.00
6:00	-93.90	1000.00
7:00	-70.43	1000.00
8:00	-39.13	1000.00
9:00	-54.78	1000.00
10:00	-93.90	1000.00

Time Series Table - Node 9

Time Hours	Demand GPM	Head ft
0:00	-3982.30	1000.00
1:00	-33.95	1000.00
2:00	-58.20	1000.00
3:00	-43.65	1000.00
4:00	-24.25	1000.00
5:00	-33.95	1000.00
6:00	-58.20	1000.00
7:00	-43.65	1000.00
8:00	-24.25	1000.00
9:00	-33.95	1000.00
10:00	-58.20	1000.00

Figure 5.34: Changes in demand values of reservoirs 1, 8 and 9.

results reiterate the need for caution in deploying CPSs for critical applications. Decision support algorithms have to be designed with knowledge of physical constraints, to avoid causing failure in an otherwise operational physical system - defying the purpose of using intelligent decision support.

6 CONCLUSION AND FUTURE WORK

The goal of the research presented in this dissertation is to model, simulate, and analyze the operation of a WDN, as a representative CPS. After careful study of the literature, which articulates challenges to these tasks; we developed a qualitative multi-agent model capable of reflecting both cyber and physical aspects of WDN operation, and more importantly, the interplay between the cyber and physical infrastructures. After extending the multi-agent WDN model to represent semantic interpretation of raw data, we enhanced the semantic model by utilizing SSM to reconcile semantic heterogeneity among the data sources - enabling support for imprecise queries. Qualitative modeling served as a precursor to quantitative analysis of the WDN, with focus on system reliability and using a Markov chain model.

The insights gained from modeling were employed in developing ontologies that represent and/or classify relationships among various aspects of the WDN, with focus on classification of failure types and identification of corresponding countermeasures for failure mitigation. These ontologies underpin automated reasoning that realizes one of the main benefits of utilizing intelligent decision support for physical system operation: increasing the robustness of the system against failure, by facilitating self-healing measures that do not rely on human operators for mitigating the effects of failure. To validate our methods, we developed and utilized a simulator capable of representing, with high fidelity, the operation of cyber and physical components and the interplay between them.

As an illustration of the practical utility of our techniques, we described and validated a decision support system that used game theory as an algorithmic tool for water allocation in a WDN. Correct operation of the decision support system, and the resulting cyber-physical WDN was validated using simulation.

A future extension planned for this research is to generalize the models and methods developed in this dissertation to CPS domains other than water distribution, in particular smart grids. This requires extensive study and analysis of the attributes and functionalities of these CPS domains, and identification of their commonalities with and differences from WDNs. The context of any CPS can be decomposed into the physical-level context, i.e., the concrete device and data level; and conceptual-level context, i.e., the abstract decision support and algorithm level. We plan to specify the context of a generic CPS in terms of these two levels. Generic context specification will facilitate the extension of our model to these domains, and is planned as a research task. Ontologies are expected to be very helpful in achieving this goal, which requires understanding of the semantics of each CPS domain.

Another extension is validation of the models and methods of this dissertation with field data. The validation carried out thus far relies on simulation alone. Field data will also be used to refine and validate our quantitative analysis. Currently, the state transition probabilities in our Markov chain model for both intelligent and purely physical WDNs have been estimated based on understanding of WDN operation and study of hydraulics literature. We have identified an industry collaborator and will be acquiring field data that will enable more accurate estimation of these values. This will facilitate more meaningful assessment of the effect of the cyber infrastructure on the reliability of water allocation. The penultimate goal is to develop a quantitative model for reliability and security of CPSs, which would guide the deployment of these systems and alleviate concerns about their dependability.

APPENDIX A

FURTHER QUALITATIVE ANALYSIS OF A WDN

This appendix articulates further detail of the qualitative WDN model presented in Section 3.1.

In qualitative modeling of agent-based WDNs, we utilize UML to capture static and dynamic aspects of the system.

1. Class Diagram Based on the use cases and interconnections defined in Fig. 3.1, Fig. A.1 provides an overview of different classes in the WDN, along with the specified attributes and the corresponding methods for each class. Figure A.1 also depicts how the classes interrelate. Other information provided in Fig. A.1 includes the data types of the attributes and the main constraints used in the decision making algorithm. The attributes of the water facility classes have been chosen to be most representative of both static (elevation) and dynamic aspects (head loss, a description of water energy) of water.

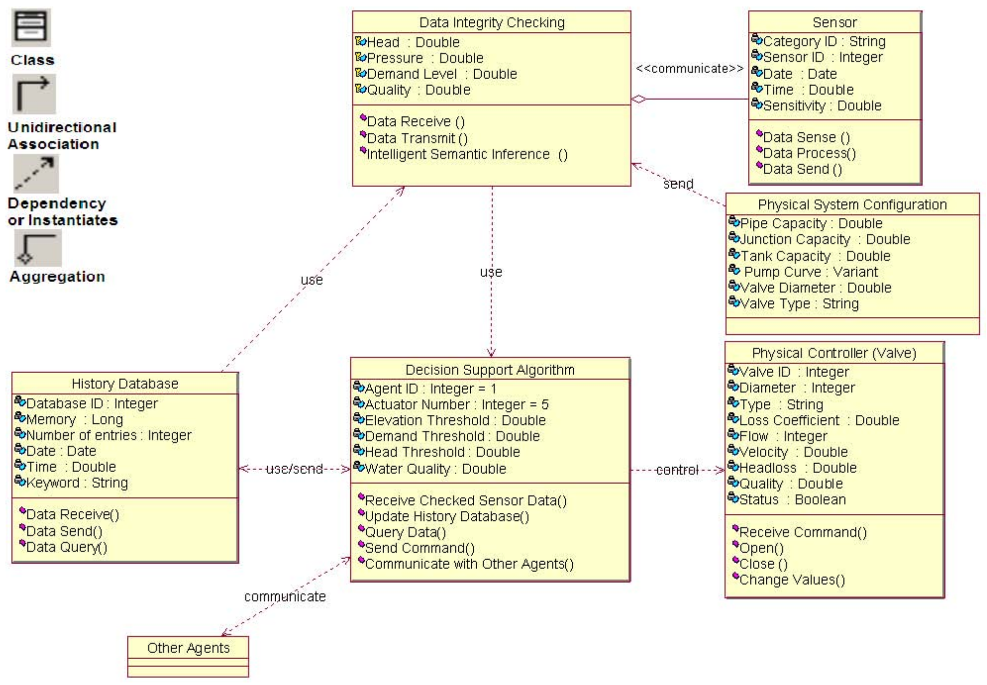


Figure A.1: Class diagram of a WDN.

The *Data Integrity Checking* class takes three data streams, from *Sensor, Physical System Configuration* and *History Database*, respectively. Data collected by the sensors is aggregated by the multiplexor (representing by the small diamond) and sent for data integrity checking. The *Physical System Configuration* block specifies the basic configuration and topology physical water infrastructure. This configuration data is sent to *Data Integrity Checking* to assist in evaluating physical constraints, e.g., judging whether a newly requested water value (such as quantity) will exceed the capacity of a pipe. History data can be queried by the *Data Integrity Checking* for comparing abnormal real-time data with historical values. Various types of semantic analysis are carried out through *Intelligent Semantic Inference*, including the aforementioned evaluation of physical constraints and corroboration with historical data or data from nearby nodes.

The purpose of this semantic inference is to screen out illegitimate or corrupted data (based on the preliminary judging criteria), to ensure that only legitimate data is sent to the decision making algorithm. The agent has varied types of association with other classes: it receives the data after semantic processing, stores the data in the history database or queries data from the database to assist in decision making (bidirectional), negotiates resource allocation with other agents, and exerts control over actuators (valves and pumps). All these functionalities are implemented through running the advance *Decision Support Algorithm* in the agent.

2. Component Diagram In Fig. A.2, the main program that implements water allocation executes on the cyber infrastructure. The physical location of the main program is immaterial. The main program is directly dependent on the code specification, which is the head file of the agent class. It includes prototype

information for the class function. The remainder of the script is the package body, which exhibits functionality similar to that of the main program and executes in distributed fashion within its autonomous management scope. For instance, if the script is written in C++, the package body is a .cpp file. An independent database is attached to each script, meaning that the script can only retrieve data from or store data to the database for management purposes within its own scope. All the data sent to the script for advanced semantic analysis or advanced computation during the phase of decision making has been checked its integrity, as described earlier in this section.

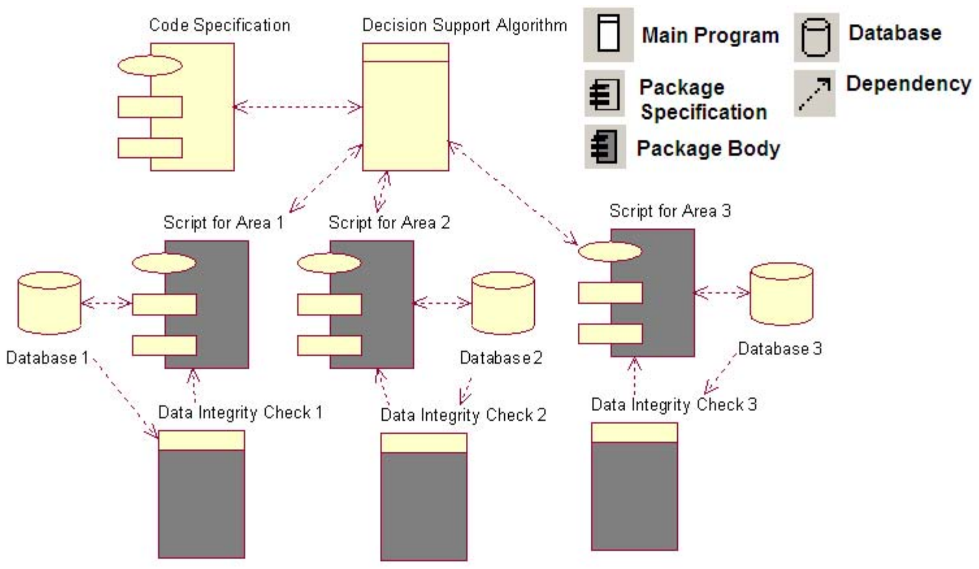


Figure A.2: Component diagram of a WDN.

3. Activity Diagram

In Fig. A.3, which depicts the activity diagram for a WDN, three entities are involved, including the physical networks; agent 1, acting as the main agent; and agent 2 as the agent interacting with agent 1.

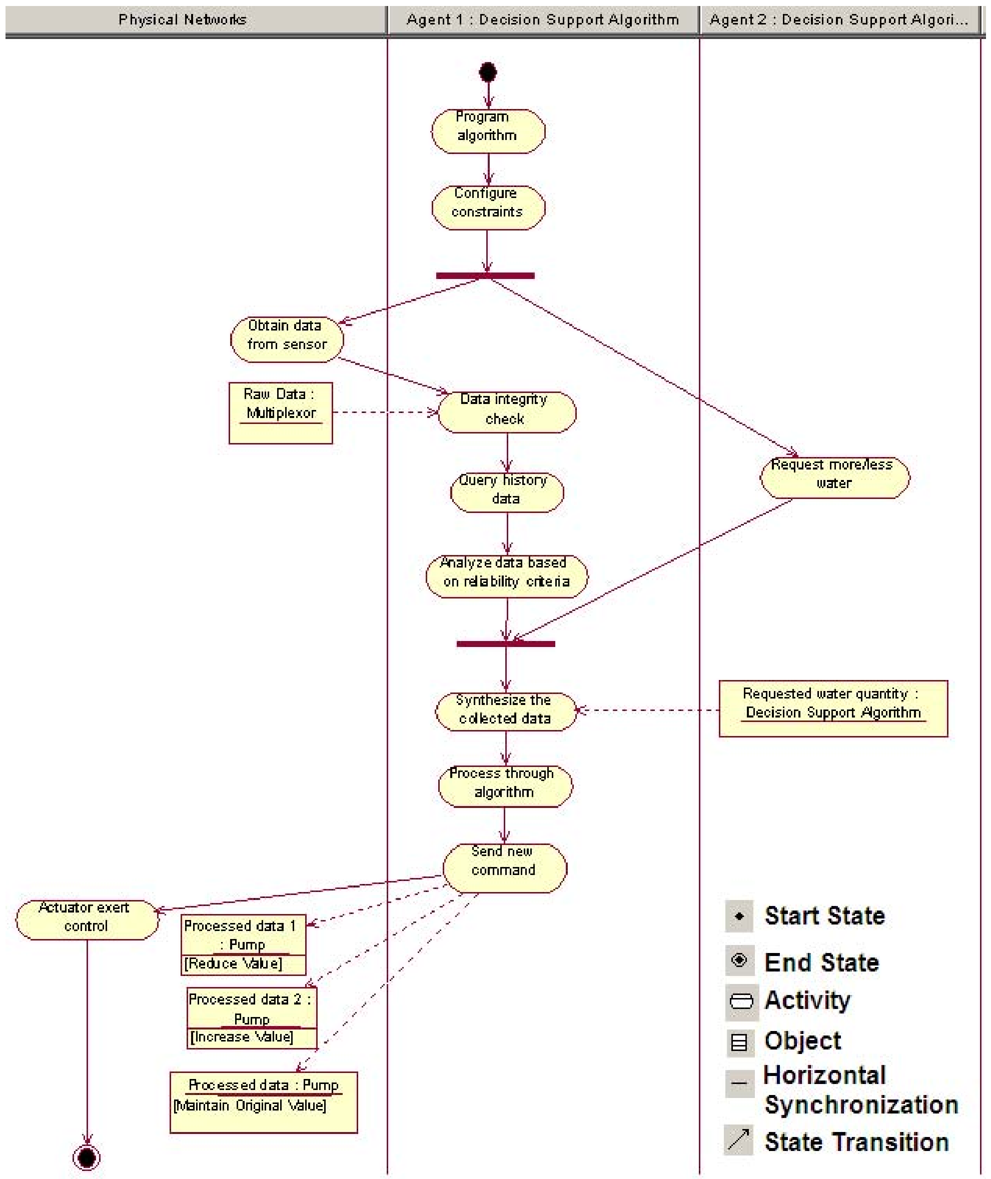

Physical Networks
Agent 1 : Decision Support Algorithm
Agent 2 : Decision Support Algori...
Program algorithm
Configure constraints
Obtain data from sensor
Raw Data : Multiplexor
Data integrity check
Request more/less water
Query history data
Analyze data based on reliability criteria
Synthesize the collected data
Requested water quantity : Decision Support Algorithm
Process through algorithm
Send new command
Actuator exert control
Processed data 1 : Pump [Reduce Value]
Processed data 2 : Pump [Increase Value]
Processed data : Pump [Maintain Original Value]
Start State
End State
Activity
Object
Horizontal Synchronization
State Transition

Figure A.3: Activity diagram of a WDN.

The activity diagram reflects how an agent interacts with the environment, and how the values in the associated object change after date integrity checking and data processing. For instance, the raw data is changed into semantically-processed data for control, and the requested water quantity of one agent may affect another agent's water consumption quantity.

4. Sequence Diagram

Figure A.4 depicts the sequence of messages exchanged among different entities in the WDN. The message on the line shows the method adopted by the receiver (class defined in the class diagram) upon receiving the message.

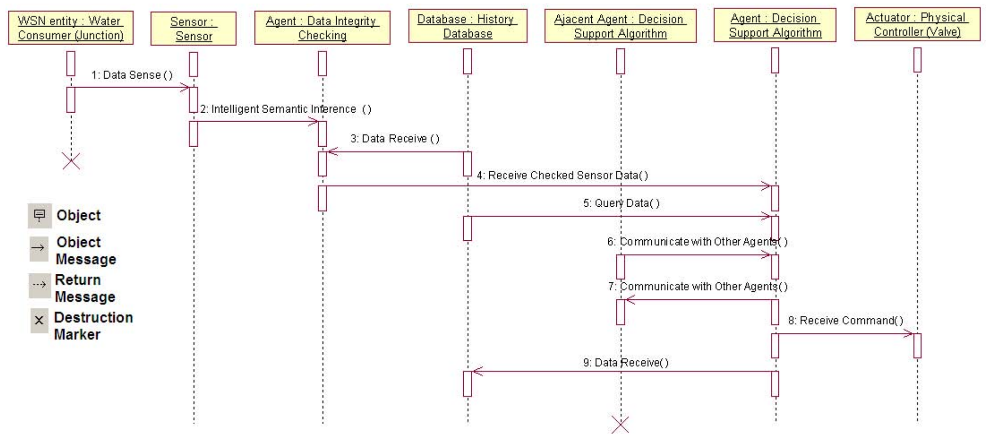

WSN entity : Water Consumer (Junction)
Sensor : Sensor
Agent : Data Integrity Checking
Database : History Database
Ajacent Agent : Decision Support Algorithm
Agent : Decision Support Algorithm
Actuator : Physical Controller (Valve)
1: Data Sense ()
2: Intelligent Semantic Inference ()
3: Data Receive ()
4: Receive Checked Sensor Data()
5: Query Data()
6: Communicate with Other Agents()
7: Communicate with Other Agents()
8: Receive Command()
9: Data Receive()
Object
Object Message
Return Message
Destruction Marker

Figure A.4: Sequence diagram of a WDN.

The figure shows the sequence of data received by the data integrity checking object and the decision support algorithm object of agent. For the former object, it directly receives and checks the raw data from the sensors (collected by multiplexor) and then if it needs to compare the real-time data with previous history data, it will receive data from its local database to make sure the result of checking is based on a reliable history record. The water consumer object and the adjacent agent object are eliminated after they send the return message, which means that no message from these two objects will be accepted outside of particular periods. The decision support algorithm of the agent first receives checked sensor data first, queries data from the history database, and finally communicates with the adjacent agent. Such a sequence is from the physical infrastructure to the cyber infrastructure (bottom-up). After the decision has been made, the calculated result will be sent to the community agent first, then a command will be sent to actuator to exert real-time control over the physical commodity, and finally the calculated data is recorded as history data in the database. Such a sequence is from the cyber network to the physical infrastructure (top-down), culminating in data recording.

APPENDIX B

FURTHER QUANTITATIVE ANALYSIS OF A WDN

This appendix elaborates on the quantitative analysis presented in Section 3.2.

We used the analysis editor provided by SHARPE to configure the scenarios simulated for the Markov model. These scenarios are given in Table B.1. The Reward rate *REW* denotes the reward, based on the contribution to the intelligence of the system. Simulation results are presented in Table B.2.

Table B.1: Scenarios simulated for Markov model of an intelligent WDN.

Label	Configuration Group
Down State (Down1) Down State (Down2) Down State (Down3) Down State (Down4) Down State (Down5)	decision_fail pipe_bursts and actuator_fail pipe_bursts sensor_fail sensor_fail and check_fail
Up State (UP1)	Components operational but no actuator_repair
Up State (UP2)	Components operational but no pipe_repair
Up State (UP3)	Components operational but no actuator_repair or pipe_repair
Initial Probability (INI)	new_quantity 0.5, sensor_working 0.1 decision_making: 0.05, actuator_working:0.05 agent_community: 0.1, pipe_working: 0.2
Reward rate(REW)	new_quantity: 5, sensor_working: 2 decision_making: 20, DB_store: 7 actuator_working: 2, sensor_fail: -2 agent_community: 10, decision_fail: -15 actuator_fail: -2, actuator_repaired: 3 pipe_bursts: -4, pipe_working: 4 pipe_repaired: 5, data_check: 12 check_fail: -10

The main purpose of this quantitative assessment exercise is to determine the value of the cyber infrastructure, in terms of added reliability of the WDN, as compared to its purely physical counterpart. The numeric results confirm that the

Table B.2: Numerical simulation results of an intelligent WDN.

Parameter	Results	Configurations
Transient Unavailability	7.25e-003	Down 1 INI at 100 sec
	1.18e-002	Down 2 INI at 100 sec
	5.91e-003	Down 3 INI at 100 sec
	3.47e-003	Down 4 INI at 100 sec
	1.29e-002	Down 5 INI at 100 sec
Down Time	3.81e+003 sec	Down1
	6.18e+003 sec	Down 2
	3.11e+003 sec	Down 3
	1.83e+003 sec	Down 4
	6.81e+003 sec	Down 5
Meantime to system failure ($MTTSF$)	1.75e+002 sec	UP1
	1.75e+002 sec	UP2
	8.69e+001 sec	UP3
Expected reward at time *t(Exrt)*	8.66e+000	REW, INI at 100 sec

greater the number of failures, the higher the *Transient Unavailability.* For instance, the value in *Down 2*(1.18e-002) is much higher than the value in *Down 3*(5.91e-003). For the most critical failure, which is the "decision_making_fail", the *Transient Unavailability* is 7.25e-003. This value is better than the case in *Down 3*(5.91e-003) but not better than the case in *Down 4*(3.47e-003). The *Down Time* refers to the period of time when a system fails to perform its primary function. The results confirm that the greater the number of failure states, the longer the *Down Time.* The case in *Down 5* is the worst case, as both the sensor and the data integrity check have failed. The resulting Down Time is the longest (6.81e+003 sec). The second worst case is *Down 2* (6.18e+003 sec), where "pipe_bursts" and "actuator_fail" occur.

The shortest *MTTSF* occurs when both "pipe_repair" and "actuator_repair" fail, confirming that these repair mechanisms can extend the life of the system (by 50% in this case). We use the *expected reward rate at time t (Exrt)* to measure the reward rate of the overall system, based on the intelligence weight assigned to each state, which is in turn based on the contribution of the state to the intelligence of the system. For example, "decision_making" has the highest reward rate (20). Similarly, as software failure will consider-ably affect the intelligence of the system, the reward rate of "decision_fail" is high (-15). In a purely physical WDN, a local node can exchange water with its neighboring nodes, which can in turn interact with each other. However, this exchange is not controlled by agents, and therefore no intelligent decision support or failure prevention measures exist.

The configuration of parameters in SHARPE and the simulation results for the purely physical WDN are shown in Table B.3 and Table B.4, respectively. Like the intelligent WDN case, the *Exrt* denotes the expected reward rate at time t and the *Mean time to absorption* denotes the state failure.

Table B.3: Scenarios for purely physical WDN.

Label	Configuration Group
Initial Probability (INI_1)	new_water_quantity: 0.5, neighbor_1:0.2 neighbor_2:0.1, actuator_control:0.2
Initial Probability (INI_2)	new_water_quantity: 0.4, neighbor_1:0.2 neighbor_2:0.1, actuator_control:0.2 sensor_detection:0.1
Reward rate(REW_1)	new_water_quantity: 5, neighbor_1:2 neighbor_2: 2, pipe_bursts:-3 actuator_control:10, sensor_detection:7 actuator_fail:-9
Reward rate(REW_2)	new_water_quantity: 5, neighbor_1:2 neighbor_2: 2, pipe_bursts:-3 actuator_control:15, sensor_detection:7 actuator_fail:-12

The numerical results show that the system reliability decreases dramatically as time goes by, in both configurations, as expected of a system with no repair. As the "actuator_control" is the sole component that can exert intelligent control, it has been assigned the highest reward rate (10 in *REW_1* and 15 in *REW_2*). The

Table B.4: Numerical simulation results of purely physical WDN.

Parameter	Results	Configurations
Reliability	7.83e-003	INI_1 at 20 sec
	6.39e-006	INI_1 at 50 sec
	3.28e-008	INI_2 at 100 sec
	7.16e-003	INI_2 at 20 sec
	5.86e-006	INI_2 at 50 sec
	3.28e-008	INI_2 at 100 sec
Mean time to system failure *(MTTSF)*	4.07e+000	INI_1 pipe_bursts
	3.94e+000	INI_1 sensor_detection
	4.06e+000	INI_2 pipe_bursts
	4.02e+000	INI_2 sensor_detection
Expected reward rate at time t (*Exrt*)	4.48e+000	REW_1 INI_1 at 100 sec
	4.67e+000	REW_1 INI_2 at 100 sec
	5.08e+000	REW_2 INI_1 at 100 sec
	5.22e+000	REW_2 INI_2 at 100 sec

main difference between REW_1 and *REW_2* is the increase of the reward rate for "actuator_control" and the decrease of reward rate for "actuator_fail" in REW_2. The result shows that the configuration in *REW_2* can enhance the *Exrt* of the system, and if "sensor_detection" takes place at the beginning of the simulation, the *Exrt* further increases. This is due to the fact that a sensor is an intelligent device and can contribute to the intelligence of the physical system; the earlier the sensor can play this role, the more it can contribute to the *Exrt*.

APPENDIX C

INTEGRATED SIMULATION OF A COMPLEX WDN

Results of integrated cyber-physical simulation of a WDN with a simple physical topology were presented in Section 3.5. Simulation of a more complex physical infrastructure is presented in this appendix.

Figure C.1 shows a screen capture at hour 8:00 of a 24-hour simulation period of a WDN. This figure also depicts node groupings, circled in green, that can facilitate study of a subset of the nodes in the topology.

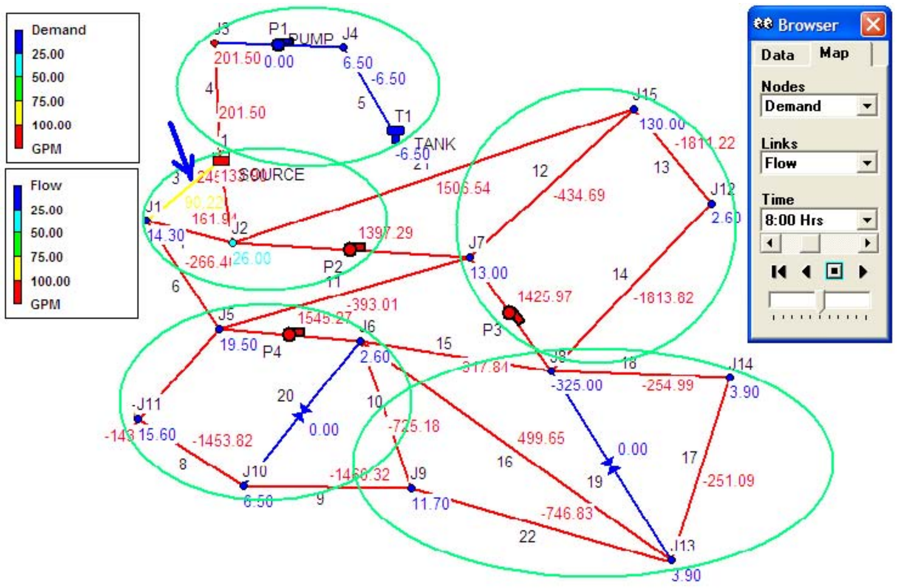
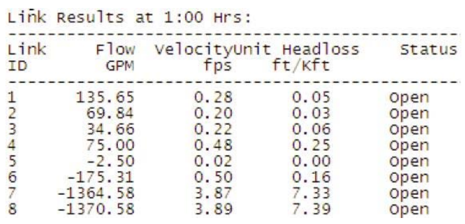


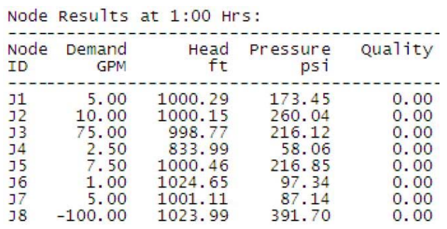
Figure C.1: A more complex topology and node groupings in EPANET.

After simulating the system for the specified duration, EPANET can provide a report in graph, table, or text form. Among the various reports available, the full report provides the most comprehensive data, including the initial and updated values of all properties of the nodes and links within each simulation time step (one hour by default). The water flow, pressure at each node, depth of water in tanks and reservoirs, and concentration of chemical substances can be tracked from the recorded data. In Fig. C.2, Figs C.2(a) and C.2(b) present snapshots of the link and node information, respectively, of the full report.



```
Link Results at 1:00 Hrs:
----------------------------------------------
Link       Flow  VelocityUnit Headloss    Status
ID          GPM         fps    ft/Kft
----------------------------------------------
1        135.65        0.28      0.05      Open
2         69.84        0.20      0.03      Open
3         34.66        0.22      0.06      Open
4         75.00        0.48      0.25      Open
5         -2.50        0.02      0.00      Open
6       -175.31        0.50      0.16      Open
7      -1364.58        3.87      7.33      Open
8      -1370.58        3.89      7.39      Open
```

(a) Link information from full report.



```
Node Results at 1:00 Hrs:
------------------------------------------------
Node  Demand       Head  Pressure   Quality
ID       GPM         ft       psi
------------------------------------------------
J1      5.00    1000.29    173.45      0.00
J2     10.00    1000.15    260.04      0.00
J3     75.00     998.77    216.12      0.00
J4      2.50     833.99     58.06      0.00
J5      7.50    1000.46    216.85      0.00
J6      1.00    1024.65     97.34      0.00
J7      5.00    1001.11     87.14      0.00
J8   -100.00    1023.99    391.70      0.00
```

(b) Node information from full report.

Figure C.2: Component information from full report.

The full report generated as the output file of EPANET is automatically saved as a plain-text .NET file. This information includes values required as input by the decision support algorithms of the cyber infrastructure, which in turn determine settings for physical control elements such as valves.

To simulate the provision of sensor readings and other information about the physical infrastructure to the cyber control system, the full report generated as output by EPANET needs to be provided as input to MATLAB. This necessitates pre-processing of the file, and parsing of the data into the matrix form required by MATLAB. A script using the *textscan* and *cell2mat* commands can be defined within MATLAB to carry out this pre-processing to generate a separate matrix from the EPANET data for each entity (node or link) for each simulation time step recorded in the full report, e.g., hour 1:00.

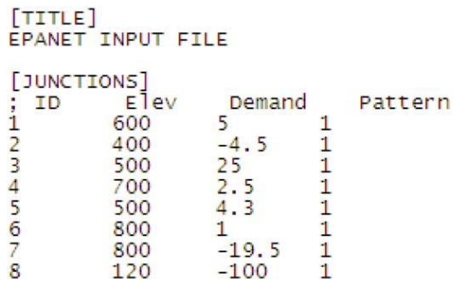
For simplicity, the simulation presented in this dissertation was focused on node flow. The controller (pump or valve) settings were determined by averaging the node demand within a node group, which is a subset of nodes defined in EPANET. Figure C.1 shows a number of groups. The same parsing approach can be used to extract additional data, e.g., water pressure or concentration of a given chemical, from the EPANET report, as required by more sophisticated decision support algorithms.

Each node group can reflect an associated group of consumers, such as residential nodes in the south of a city. The only requirement is that each node group include at least one controller (pump or valve), so controller settings determined by the cyber infrastructure can be utilized in water allocation. The focus of the research presented in this dissertation was integrated simulation of the CPS, and as such, a simplistic approach was taken to water allocation, with the goal of distributing the water as equitably as possible, subject to physical constraints on the nodes.

MATLAB generates a matrix of controller settings, which need to be provided to EPANET, as they would be to the physical control elements in an actual WDN. A .INP file is required, in a format identical to the original input provided to EPANET in the first step of the simulation, with controller values updated to reflect the settings determined by the decision support algorithm. A MATLAB script utilizing the *dlmwrite* and *fprintf* commands can be used to generate a .INP file with the format expected by EPANET.

In the final stage of the simulation, the .INP file generated by MATLAB, which specifies settings for various control elements, is used to initiate another execution of EPANET, closing the physical-cyber-physical loop. The process can be repeated as necessary to simulate operation of the WDN over multiple cycles of cyber control. Figure C.3 shows the file resulting from execution of the water allocation algorithm for the node groups of Fig. C.1.

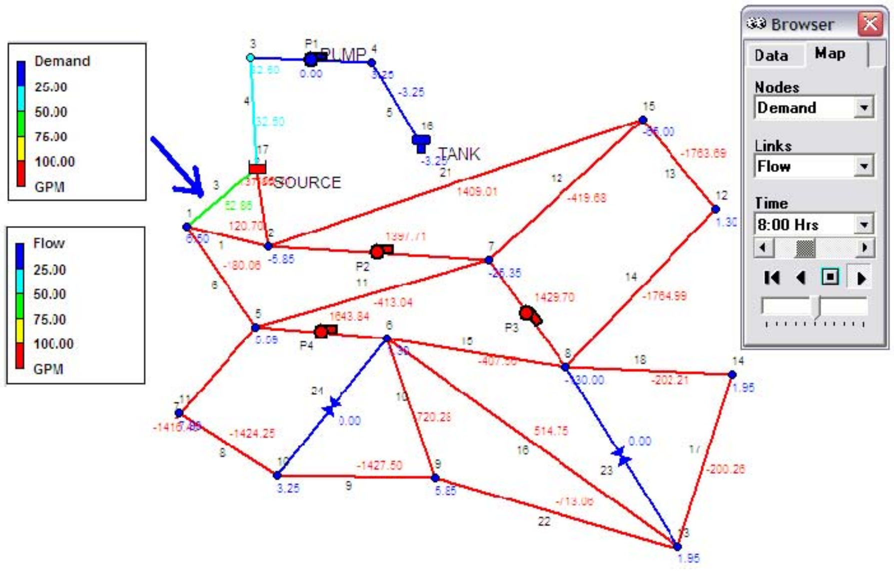
The result of executing EPANET with the .INP file generated by MATLAB is shown in Fig. C.4. As an example of the manifestation of cyber control, the flow in the link connecting Junction1 (J1) and SOURCE, marked with an arrow, has been reduced from 75-100 gpm (yellow) in Fig. C.1 to 50-75 gpm (green) in Fig. C.4.



```
[TITLE]
EPANET INPUT FILE

[JUNCTIONS]
; ID      Elev     Demand     Pattern
1         600      5          1
2         400      -4.5       1
3         500      25         1
4         700      2.5        1
5         500      4.3        1
6         800      1          1
7         800      -19.5      1
8         120      -100       1
```

Figure C.3: EPANET input file generated by MATLAB.



Nodes: Demand
Links: Flow
Time: 8:00 Hrs

Figure C.4: Complex topology after applying cyber control.

APPENDIX D

INVESTIGATION OF THE EFFICACY OF DATA RETRIEVAL FROM A FAULT MITIGATION DATABASE

The premise underlying the automated failure mitigation proposed in this dissertation is that a database is available that associates a set of countermeasures with a set of failure types. In this appendix, we utilize rank-weight-biased precision (RWBP) to measure the effectiveness of data retrieval from such a database, in the hope of determining practices most likely to lead to an efficacious search. The work described here will be submitted in August 2011 to the Track on Dependable and Adaptive Distributed Systems (DADS) of the ACM Symposium on Applied Computing [19].

D.1 MOTIVATION

What we have presented in [11] and [14] are the semantic agents and SSM model, which allow us to interpret the semantics in the data stream and reconcile heterogeneity in the semantics in the query. One of the greatest benefit lying in SSM model is the tolerance of imprecise query on the failures. When an imprecise query is interpreted, the embedded information of failure is classified on certain position on the taxonomy (a type of ontology) of failure type database, as shown in 4.8 and measured by SDM in 3.15. Once the failure type has been identified, the corresponding mitigation techniques can be found by the help of ontology reasoning [17]. The failure can have a large number of available mitigation techniques, if it locates at the higher level on the taxonomy, such as a broad category like a “NodeFail”. On the other hand, a specific failure type can have very limited number of mitigation techniques, if it locates at lower level on the taxonomy, such as “ExceedTotalHead” at the root of failure type database.

SSM model can greatly reduce the heterogeneity in the useful semantic information extracted from raw data collected by sensor. The performance evaluation of SSM has been extensively discussed in Section 3.4, but limited discussion is carried on the precision of the interpretation, i.e. how much uncertainty incurred when the failure is classified on the taxonomy.

Both of the failure type taxonomy and the mitigation technique taxonomy discussed in Section 4 are represented by set of data and stored in database. As a CPS is a purely automatic system without human intervention, we assume that system takes the role of human operator completely. The system can automatically interpret the query, classify the failure type on the predefined taxonomy, scan the mitigation technique in the mitigation database according to the identified failure type, stop at one mitigation technique and select it as the appropriate one for the failure type. As discussed above, based on the semantics in the query, when the classification of failure on the taxonomy has reached certain level on the taxonomy, a relatively narrow data pool with multiple mitigation techniques are associated with one failure type. Each mitigation technique can be abstracted as data in the database, with different usefulness to resolve that particular failure characterized in different weight. The system needs to select which data is a good solution and decides when to stop retrieving the data from the pool. The system does not necessarily look for all the data available in the data pool, as it can not feasibly seek all data in a large mitigation database (more than one million data) for a high-level failure.

The quantification of the lost reliability due to the acceptance of imprecise queries, which add more uncertainty in the complex CPSs, is not discussed in this section. What we focus on is the process beyond the point that one failure type has been identified on the taxonomy of failures, and we investigate at the effectiveness of data retrieval when the system selects the available option from data pool. Figure D.1 depicts the scenario about how system selects the available mitigation technique to address an identified failure in the failure database. It shows that SSM resolves the problem of imprecise query but introduces the uncertainty. After a failure is identified, the system starts to automatically select the available mitigation technique from the database. The distribution of the available mitigation techniques and the usefulness of each data are unknown by the system.

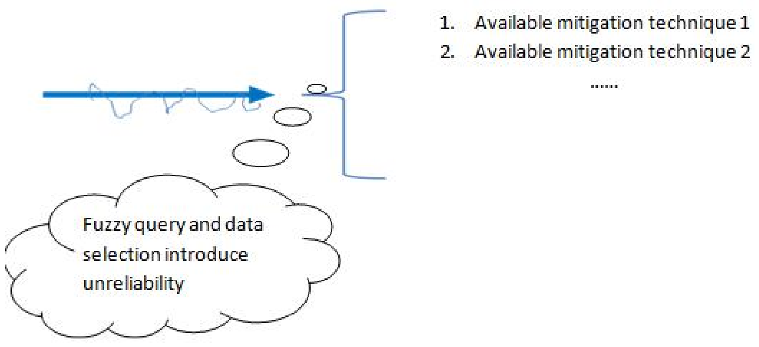

1. Available mitigation technique 1
2. Available mitigation technique 2
......
Fuzzy query and data selection introduce unreliability

Figure D.1: Unreliability introduced in the imprecise query and data selection.

In this section, we measured the effectiveness of data retrieval in the mitigation search-and-selection phase, given different data distribution cases and search ways. The simulation reveals how to configure the system can result in a cost-effective search, which is quite meaningful as the determination of the selected data is the final stage of the "query-retrieval" process. The empirical result can facilitate more effective configuration of the system for automatic operation. Because selection is a random process, a statistical model is suitable to capture the uncertainty in the data retrieval phase. We utilized the theory developed for the information retrieval system in [54] to capture the uncertainty in this process.

D.2 BACKGROUND ON INFORMATION RETRIEVAL SYSTEM

Information retrieval system needs to compute a *score* to represent the degree of satisfaction between the selected data and a query. Specifically, each score is a numeric estimation about the probability that how much the selected data matches with the information (semantics) embedded in the query. The semantics in the query have been interpreted by SSM, and on the basis of the selected failure, we can quantify the effectiveness of data selected from mitigation database. All the available data has an associated weight.

The two notations in the information retrieval theory are *total number of relevant documents* for the query (R) and *total number of retrieved documents* (d). Here, a document refers to a single data in the database, and we use it in the remaining part of this section to conform to the terminology in information-retrieval theory. It is often the case that R is larger than d, which implies that system can only feasibly check part of the relevant documents rather than all of the documents that match with system's query.

The *recall* and *precision* are the two well-known elementary notations for measurement. Their definitions are illustrated as Fig. D.2. *Recall* is the proportion of the relevant documents that have been retrieved, i.e. C/R, while *precision* is the proportion of retrieved documents that are relevant, i.e. C/d. These two concepts tend to be in tension.

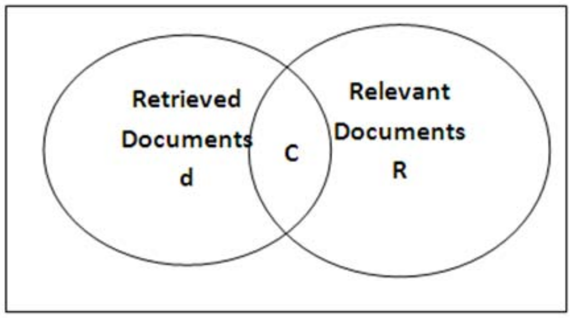


Figure D.2: Retrieved vs. relevant information in database.

Most of the existing studies towards the information retrieval effectiveness measure the *average precision* (AP), *precision at d documents retrieved (P@d)*, *R-precision (P@R)* and *reciprocal rank (RR)*. All these measurements require the knowledge of R, which is unknown in some cases. Besides, the existing measurement methods have shortcomings due to other two facts: i)complete relevance judgments are impractical in current system and the recall tends to be overestimated; ii)recall, as an overall evaluation of relevant documents, does not capture the notions that the system selects data randomly and the behavior of search can stop at any relevant document.

Given a document ranking, i.e. the sequence of documents, we use *rank-weight-biased precision (RWBP)* to measure the data retrieval effectiveness in the system. The benefits are summarized as follows:

1. It measures the rate at which data usefulness is gained by a system working at a given degree of persistence, measured by *advanced probability* p that the system continues to search for the next relevant data. The higher the degree of persistence, the more data that the system will look for before stops and selects the data. The probability of continuous searching can be previously configured in the system.

2. By adjusting the persistence p, a *parameter (RWBP)* that represents the efficiency of system searching behavior has the advantage of capturing the critical facets of other measurements, including *AP, RR* and *P@d*.

3. *RWBP* allows quantification of effectiveness when only partial relevance judgments are available, specifically, the system does not need to query all the relevant data residing in the database.

4. *RWBP* takes the weight of each data into consideration.

D.3 MODEL

For the intelligent WDN case, all the available mitigation techniques for all types of failures reside in the database in parallel, i.e. there is no priority to retrieve any one of them when the system performs searching automatically. We suppose the data retrieval is performed like a scan from left to right as shown in Fig. D.3. The system only has an identified failure type, and has no awareness of the distribution of the useful mitigation techniques, which are scattered randomly in a huge database system. The dashed line indicates the available mitigation technique is a weak one for resolving the failure, while the solid line indicates a strong one.

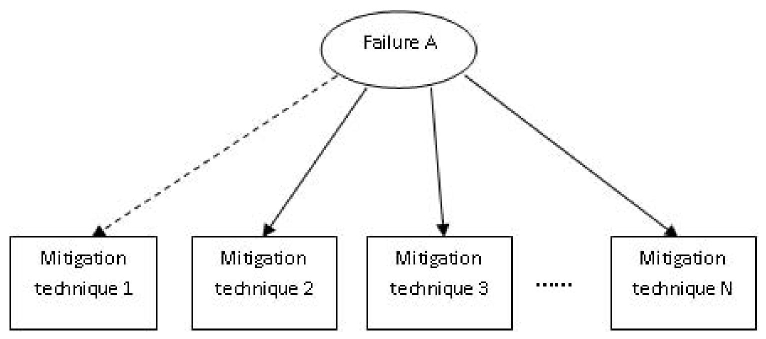


Figure D.3: Multiple mitigation techniques for one failure type.

We suppose for each of the mitigation technique, it has an associated efficiency value measured by *weight* (from 0 to 1). The efficiency value is evaluated and assigned by the experts in the design phase of the system beforehand, based on the previous accumulated experience on the usefulness of one particular mitigation technique to address a particular failure. Sometimes, certain mitigation technique can resolve various types of failures, for instance, "Adjust Pressure" can resolve the failures such as "ExceedPressure" and "PipeBurst", with different efficiency. For the strongest mitigation technique, its *utility* to the system is 1 (a unit to measure the effectiveness of the mitigation technique that addresses the failure). For the weakest or irrelevant one, its utility is 0. The deeper that the system searches through the ranked documents (the mitigation technique profile), the more utility that the system can gain. If the total number of researched documents is high, the total utility will be increased as well.

Introduction of probability p captures the notion of "expected search length per desired relevant documents" or "rate at which relevant documents are found". We assume that p is a constant for each movement forward. We also assume that the stop of searching is determined independently of the position in the ranking, independently of the previous decision, and irrelevant of the weight of the document just searched. In the intelligent WDN case, the system has no cache for storing the weight of previous searched results, therefore, it has no sorting process. Given the advanced probability p of seeking appropriate document in the database, the behavior of the system is described by the algorithm in Fig. D.4.

```
{int retrieved_doc = d;   //d is the number of retrieved documents
int relevant_doc = R;     //R is the number of relevant documents
if R < d
d-- until d <=R; //We suppose that the number of
                  //retrieved documents in query should be less than the
                  //available relevant documents in the system

                 //Start seeking the relevant document
int i;          //counter of the retrieved documents
double w;       //weight of mitigation technique to address failure
double p;       //proceeding probability
double U = 0;   //total utilities of the retrieved documents

for (i = 1;;i <= d;)
{
if i++,              // counter increases to proceed retrieval
     then proceed to view next doc in the ranked list with p;
     if  w > 0,    //indicate the mitigation technique is useful
        add the utility of mitigation technique into
        total utility U with associated probability p;
  if i stops,
     finish searching, calculate utilities;
}
}              //end searching
```

Figure D.4: System search behavior given advanced probability p.

According to the algorithm in Fig. D.4, the probability that the counter i will reach to the i^{th} document is p^{i-1}. On average, the number of documents examined during each search (N) can be calculated as Equation D.1:

$$\sum_{i=1}^{\infty} i \cdot p^{i-1} \cdot (1-p) = \frac{1}{1-p} \tag{D.1}$$

The total expected utility (U) is calculated as Equation D.2:

$$U = \sum_{i=1}^{d} w_i \cdot p^{i-1} \tag{D.2}$$

The expected rate at which utility is transferred from the automatic searching service provider is the total utility divided by the average number of searched documents, i.e. the *RWBP* in Equation D.3. *RWBP* should be within the range from 0 to 1.

$$RWBP = U/N = \frac{\sum_{i=1}^{d} w_i \cdot p^{i-1}}{\sum_{i=1}^{d} i \cdot p^{i-1} \cdot (1-p)} \tag{D.3}$$

D.4 SIMULATION RESULTS

We utilize MATLAB to simulate the searching behavior of the system given different ranks and advanced probabilities.

D.4.1 N, U, RWBP for Four Probabilities of Continuous Search. We design three different ranks and associated weights to observe how the system behaved.

Initially, we suppose the rank and associated weight as below, with relevant documents $R = 7$ and total retrieved documents $d = 14$. In real application case, the relevant documents and total retrieved documents can be very large, such as thousands or millions available documents in the database [54]. However, for a failure type that has been identified in the water distribution case, the number of matched mitigation techniques are limited, and $R = 7$ is a reasonable assumption.

Ranking1: **-♣-♣♣-♣♣♣-♣–** (♣ represents relevant document)

Weight1 of each: [0, 0.1, 0, 0.3, 0.5, 0, 0, 0.2, 0.4, 0.8, 0 ,0.7, 0, 0]

The rank shows that the distribution of relevant documents is quite even. Given four different probabilities, $p1 = 0.25$, $p2 = 0.5$, $p3 = 0.8$, $p4 = 0.95$, we can calculate corresponding average number of documents(N), total utilities (U), and $RWBP$ values.

The average number of documents (N) is straightforward to be obtained. As the factor of d goes to infinity, the higher probability, the higher N will be, indicating the more persistently the system searches, the more documents the system retrieves. The calculated (N)s are $N1 = 1.33$, $N2 = 2$, $N3 = 5$, $N4 = 20$.

The results of total utilities for four probabilities are shown in Fig. D.5. As the number of retrieved documents increases, the higher the probability to continue searching, the higher the total utilities will be gained. Higher weight has higher contribution in the total utilities, even though the relevant documents can appear late.

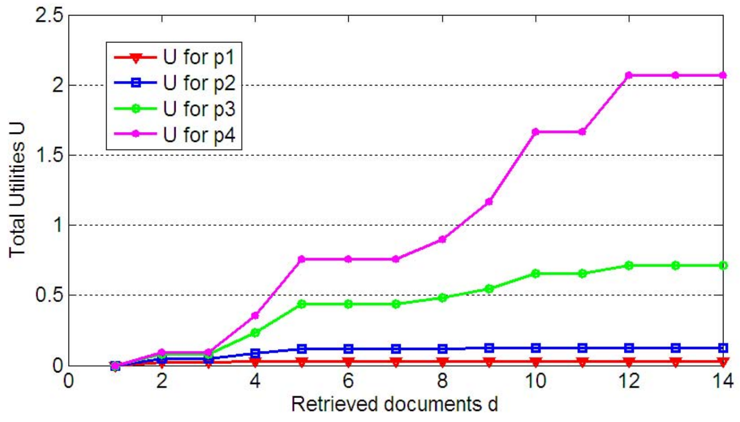


Figure D.5: Total utilities for weight1.

*RWBP*s for four probabilities during simulation are shown as Fig. D.6. When $p = 0.25$ and $p = 0.5$, the variation of *RWBP* is similar. *RWBP* in $p = 0.5$ is always higher than the one in $p = 0.25$, almost twice after $d = 3$. When p reaches 0.8, *RWBP* begins to surpass the one in $p = 0.5$ when $d = 4$, but even less the one in $p = 0.25$ before d reaches 4. As p approaches to 0.95, *RWBP* is the lowest among all the four cases, but starts to surpass the one in $p = 0.5$ when $d = 9$, and it is anticipated that it has a tendency to surpass *RWBP* in $p = 0.8$ at certain d. This tells us that, for an evenly-distributed rank, a low advanced probability might be more cost-effective for the case that system searches at shallow depth, such as the first 5 documents. As system continues seeking for more documents, a relatively high advanced probability contributes higher effectiveness. In brief, this advanced probability should be configured at a moderate value, given the length of the rank. If the rank is not long, a high advanced probability is not cost-effective.

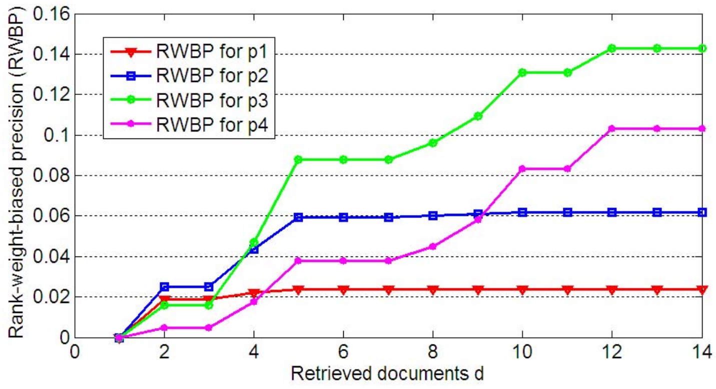


Figure D.6: RWBP values for weight1.

Then, we design *rank2* that relevant documents appear late. In this case, $R = 7$, $d = 14$, and the weight for each relevant one remain the same.

Ranking2: ——♣♣♣-♣-♣♣♣ (♣ represents relevant document)

Weight2 of each: [0, 0, 0, 0, 0, 0.1, 0.3, 0.5, 0, 0.2, 0, 0.4, 0.8, 0.7]

The calculation of N remains the same. Simulation results for total utilities U are shown in Fig. D.7.

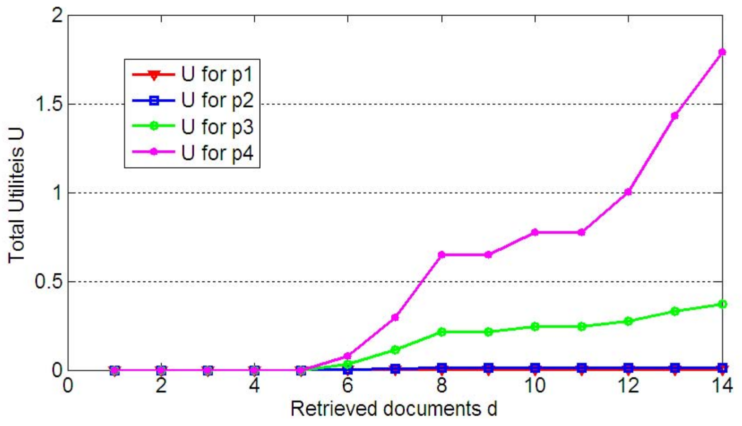


Figure D.7: Total utilities for weight2.

The total utilities start to grow once the relevant document is detected. Same as case in *rank1*, as d increases, the total utilities start to grow along with the detection of relevant documents. The final U for each probability is $U1$=0.00020233, $U2$=0.0126, $U3$=0.3709 and $U4$=1.7923, respectively. Compared with the U in *rank1*, they are lower. It shows that the distribution of relevant documents can affect total utilities, i.e. the later the relevant documents appear, the lower total utilities will be. Therefore, to increase the efficiency of data retrieval by increasing total utilities, the relevant documents should appear at earlier positions in the rank.

Simulation result for *RWBP* given *rank2* is as Fig. D.8. The *RWBP* for each probability is *RWBP1* = 0.00015175, *RWBP2* =0.0063, *RWBP3* =0.0891 and *RWBP4*=0.5242, respectively.

Generally speaking, compared with the *RWBP*s in *rank1*, they are lower. This is because N remains the same, but the total utilities decrease. As the relevant

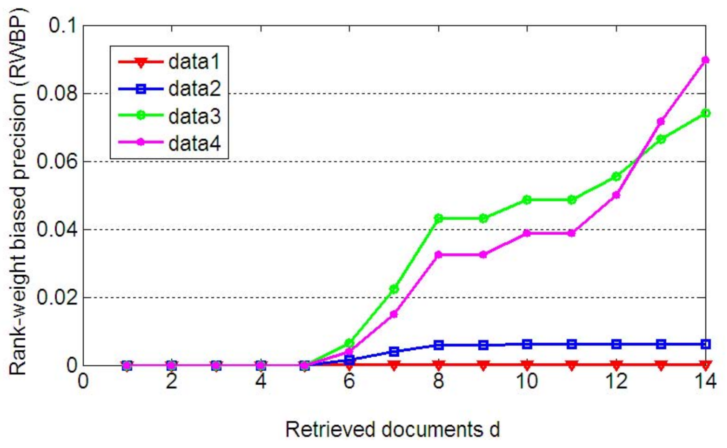


Figure D.8: RWBP values for weight2.

documents appear late, the *RWBP* that captures the rate at which relevant documents are found slows down. Note that the interpretation of precision scores needs to be tempered by knowledge of R, the number of relevant documents. Similar as case of *rank1*, *RWBP* in the most-persistent system (p=0.95) has not started to lead the value until d is larger than 12. Since a persistent system ($p = 0.95$) is guaranteed to obtain a low expected utility from a search with only a few relevant documents. Non-persistent system will also obtain low *RWBP* scores if none of the initial few documents are relevant. Different from the case in *rank1*, *RWBP* for non-persistent system (for p is less than 0.5) is lower than the one in persistent system, from the moment when the first relevant document is detected.

Finally, we change the rank and weight as below, and name it as *rank3*. It represents the case that the relevant documents appear early. In this case, $R = 7$, $d = 14$, and the weight for each relevant one remains the same.

Ranking3: ♣♣♣♣—-♣♣-♣– (♣ represents relevant document)

Weight3 of each: [0.1, 0.3, 0.5, 0.2, 0, 0, 0, 0, 0.4, 0.8, 0, 0.7, 0, 0]

N is the same as *rank1* and *rank2*. Simulation result for total utilities U is in Fig. D.9. The U for each probability is $U1 = 0.2094$, $U2 = 0.4035$, $U3 = 0.9970$

and *U4* $= 2.1755$, respectively. Compared with *RWBP*s in *rank1*, they are higher. It justifies the analysis for the vice-versa case that the relevant documents appear late.

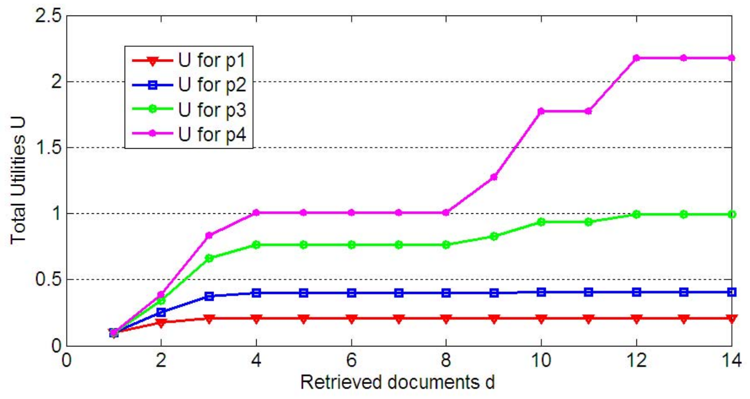


Figure D.9: Total Utilities for weight3.

Simulation result for *RWBP* in weight 3 is as Fig. D.10.

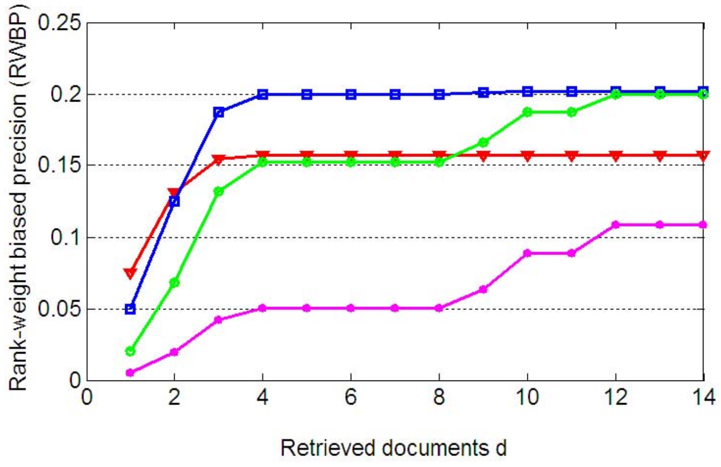


Figure D.10: RWBP values for weight3.

Compared with *RWBP* in *rank1* and *rank1*, the difference is distinct. Generally speaking, the value of *RWBP*s in this case are much higher, and the non-persistent system has the highest effectiveness. For persistent system, even the advance probability $p = 0.8$ is slightly higher than $p = 0.95$, the *RWBP* for the previous probability is more than twice of the *RWBP* for the latter. The simulation result reveals that, for a rank that relevant documents appear early, the system can just select the initial relevant documents and then gain relatively high effectiveness.

As relevant documents appear less frequently in later positions, the total utilities slow down to increase, and the amount of increased *RWBP* decreases. For non-persistent system, *RWBP* reaches to stable value since $d = 4$, whereas the persistent system has potential to drastically improve *RWBP* even after $d = 8$. *RWBP* for $p = 0.8$ surpasses the one for $p = 0.5$ after $d = 8$.

D.4.2 Simulation Results for Conditional Search. We add the conditional probability of advancement, subjected to whether the previous examined document is relevant. If the previous document is relevant, the probability to proceed to next one is *p1* and we assume it as *p1* $= 0.4$; if not relevant, the advanced probability is *p2* and we assume it as *p2* $= 0.9$. In comparison, for the unconditional search, we assume $p = 0.5$.

Given *weight1*, we can simulate the number of searched documents for at each d, based on the expected number of retrieved documents as following Equation D.4:

$$N(d) = \sum_{i=1}^{d} i \cdot p^{i-1} \cdot (1-p) \tag{D.4}$$

The simulation result for N is as Fig. D.11.

For the unconditional case, the finalized number of examined documents is higher, whereas number of examined documents in the conditional case grows steadily.

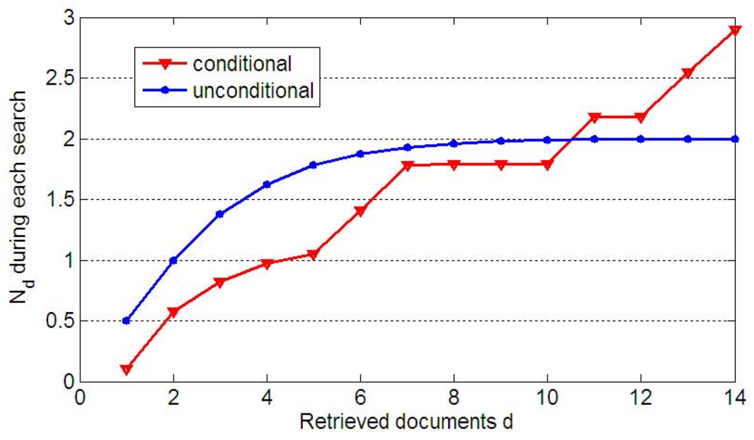


Figure D.11: N for conditional vs. unconditional search in weight1.

It indicates that, even the conditional probability is higher when the previous document is irrelevant, the persistent system without condition ($p = 0.5$) still has searched more documents on average.

Simulation result for U according to Equation D.2 is as Fig. D.12.

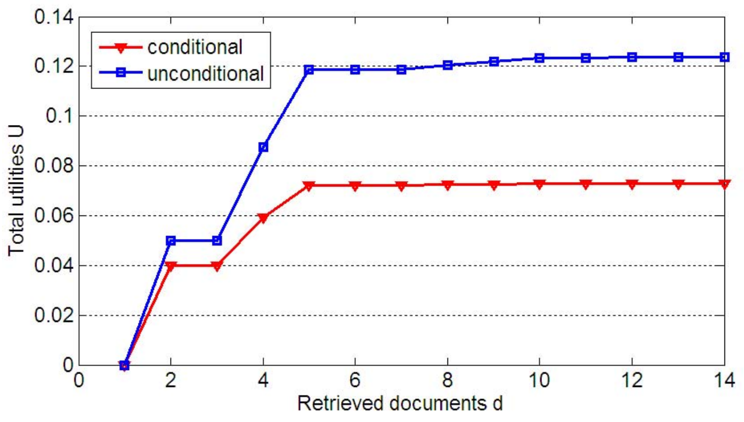


Figure D.12: U for conditional vs. unconditional search in weight1.

For the unconditional case, the total utility is higher than the one in conditional case. Again, on average, the system that searches documents without constraints on the previous document still gain higher total utilities.

Simulation result for $RWBP$ according to Equation D.3 is as Fig. D.13.

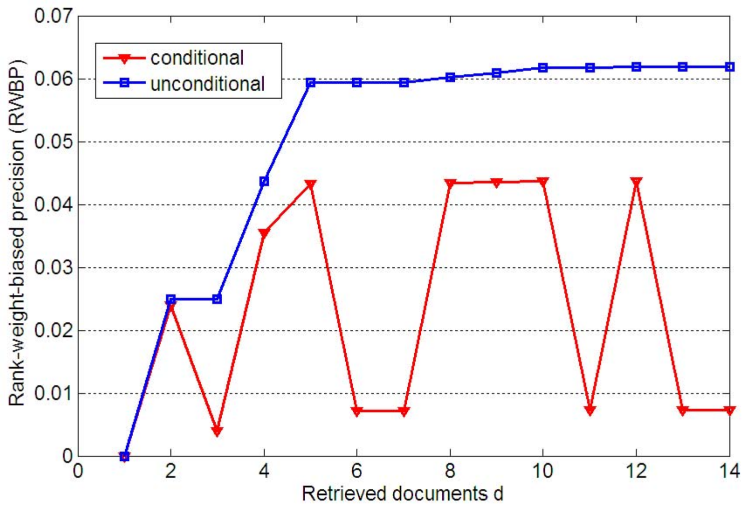


Figure D.13: RWBP for conditional vs. unconditional search in weight1.

$RWBP$ of unconditional search is much more steadily-growing and higher than that in conditional search. $RWBP$ is changing dramatically according to the positions of relevant documents. This is mainly because, the average number of examined documents N are quite different given the previous document is relevant or not. If the previous document is relevant, then N is *1/(1-p1)* = 1.67, whereas if it is not relevant, $N = 10$. Because of Equation D.3, $RWBP$ has much greater transition whenever the previous document is not relevant. The result proves that a high advanced probability indeed can improve the data retrieval effectiveness given the previous document is not relevant. The weight of each document does not provide much contribution in the discussion here.

Secondly, we investigate the case in *rank2*. Given *weight2*, the simulation result for N at each d is as Fig. D.14.

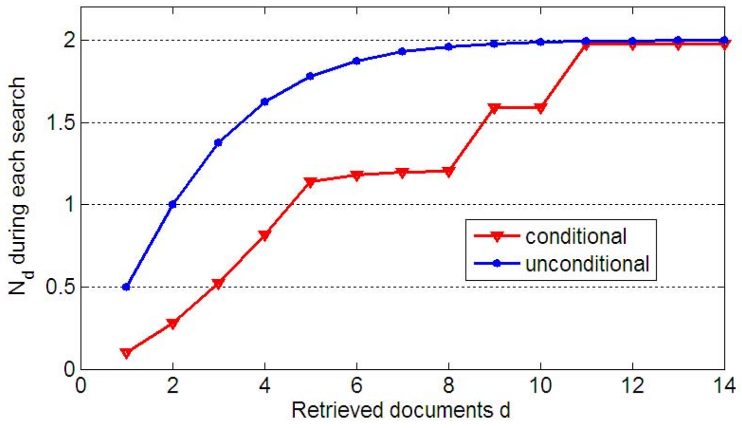


Figure D.14: N for conditional vs. unconditional search in weight2.

Similar to *rank1*, the average number of examined documents is lower in the conditional case than in the unconditional case. As the relevant documents appear late, N changes late in conditional case, whereas the change of N becomes minor and steady in unconditional case.

Simulation result for U is as Fig. D.15. Total utilizes start to increase in both cases after relevant documents have been detected. However, the total utility in unconditional case is 6 times higher than the one in conditional case, whereas in *rank1*, the difference is smaller than double times. And U in *rank2* is much smaller than the one in *rank1*, particularly, U for unconditional case in *rank1* is 10 times larger than the one in *rank2*. It shows that, even in the same weights, if the relevant documents appear late, the total utilities can be greatly affected.

Simulation result for *RWBP* is as Fig. D.16.

Similar to the situation in *rank1*, the effectiveness in unconditional case is steadily increasing, while the one in conditional case varies subjected to the position

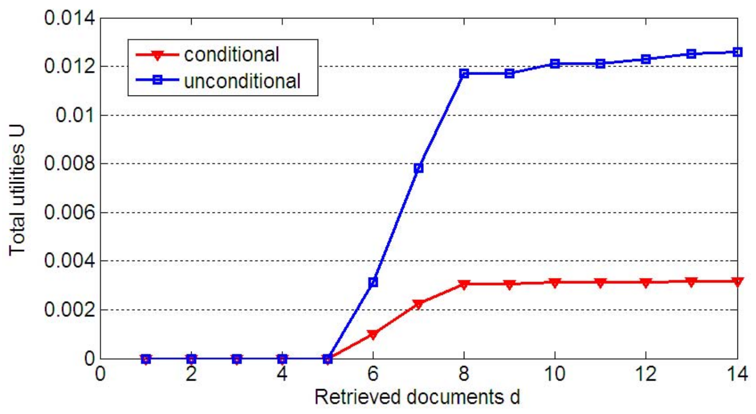


Figure D.15: U for conditional vs. unconditional search in weight2.

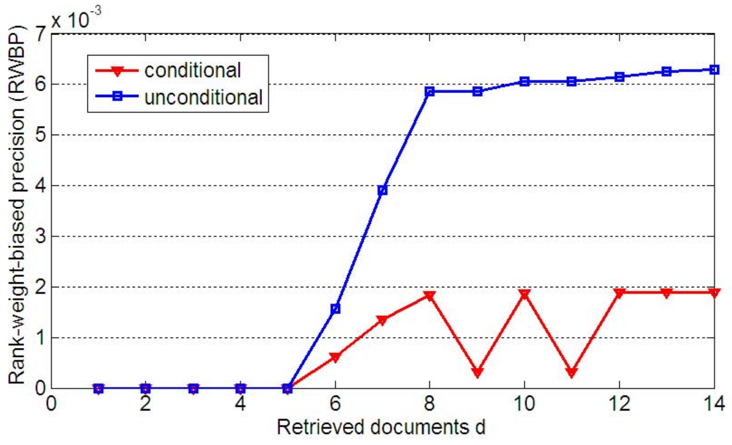


Figure D.16: RWBP for conditional vs. unconditional search in weight2.

of irrelevant documents. Compared with the values of RWBP in *rank1*, the one in *rank2* is much smaller, specifically, the *RWBP* in *rank1* is about 1000 times of the one in *rank2*. This again shows that the effectiveness of data retrieval can greatly deteriorate in the rank where relevant documents appear late.

Finally, we investigate the case in *rank3*. Subjected to *weight3*, the simulation result for N is as Fig. D.17.

As the relevant documents appear early, N is higher in conditional case than the one in the unconditional case, but the value in unconditional case still grows more

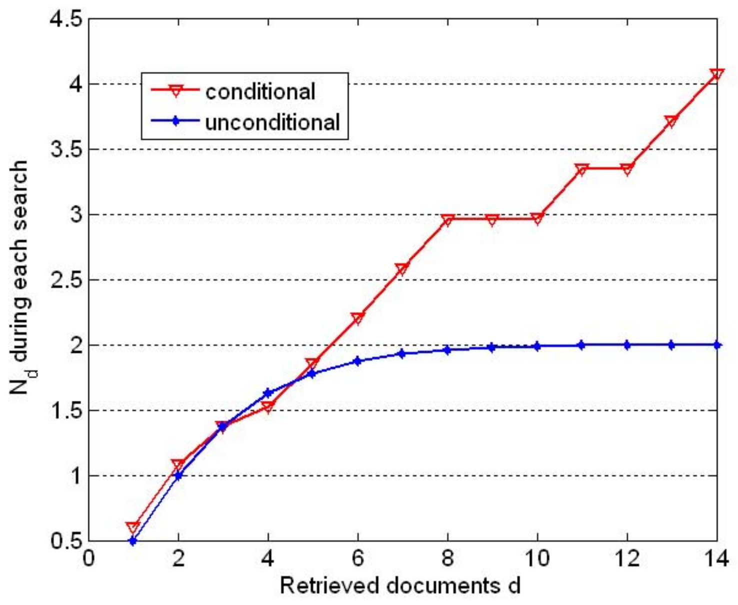


Figure D.17: N for conditional vs. unconditional search in weight3.

steadily. The average number of searched documents in *rank3* are higher than both the ones in *rank1* and *rank2*.

Simulation result for U is as Fig. D.18.

The values of total utilities in conditional case are still lower than the one in unconditional case. But in *rank3*, both of them grow steadily and are much higher the corresponding values in *rank1* and *rank2*.

Simulation result for $RWBP$ is as Fig. D.19.

In *rank3*, as the relevant documents appear early, $RWBP$ in both cases starts to grow at the beginning. This is due to the fact that the U grows more faster than N grows at first 4 relevant documents. $RWBP$ in unconditional case reaches to stable value around 0.2 after $d = 4$, whereas $RWBP$ in conditional case drops sharply when irrelevant documents appear. In *rank3*, the amount of variation in $RWBP$ is the largest one among the three ranks, and the highest value can be almost 40 times larger than the lowest one. The effectiveness in *rank3* is better than *rank1* and *rank2*, in both conditional and unconditional cases.

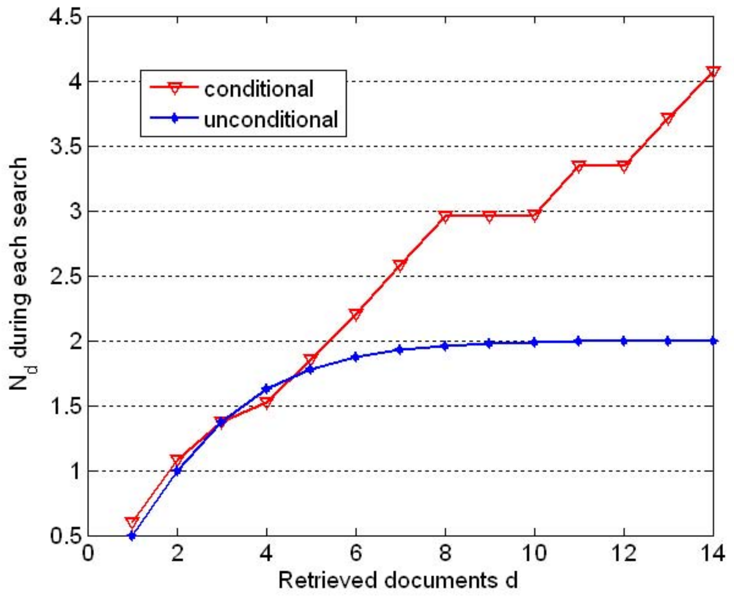


Figure D.18: U for conditional vs. unconditional search in weight3.

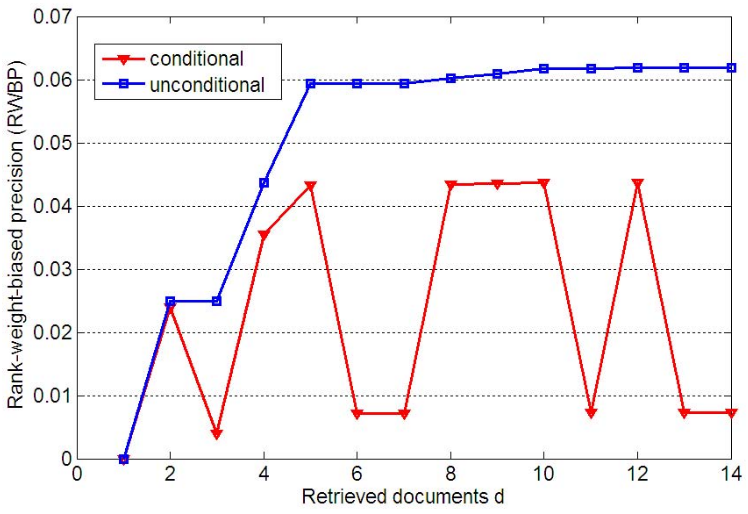


Figure D.19: RWBP for conditional vs. unconditional search in weight3.

In summary, the performance of unconditional case is better than the one of conditional case, measured by total utilities and effectiveness. The simulation results build the basis for system configuration to carry on the search task. When the system is configured to seek for the relevant mitigation technique, it is more cost-effective to configure the advanced probability to be unconditional in general.

D.4.3 Minimum Depth of Query. We can calculate the minimum depth of query (d) based on the requirement of precision.

A useful consequence of the proposed *RWBP* metric is that it is possible to compute upper and lower bounds on effectiveness, even when the knowledge of ranking and relevance judgments is partial rather than comprehensive in a large system. With the help of *RWBP*, it is straightforward to accumulate an uncertainty value that captures the unknown component of the effectiveness metric.

The simplest case is when the ranking is calculated to a depth of d answers per query, and the contributions from depth d + *1* on are not available. Then the uncertainty in the *RWBP* score is given as following Equation D.5:

Equation D.5:

$$(1-p)\cdot\sum_{i=d+1}^{\infty}p^{i-1}=(1-p)\cdot p^{d}\cdot\sum_{i=1}^{\infty}p^{i-1}=p^{d} \tag{D.5}$$

The calculation of uncertainty value can be done in advance of any experimentation. For example, with $p = 0.25$ and a pooling depth of $d = 15$, the uncertainty value from all remaining terms in the geometric series is $0.25^{15} = 9.3132\times10^{9}$, which implies the precision that calculated *RWBP* should be quoted to 10 decimal digits. Conversely, when four decimal digits of accuracy are required, i.e. the residual should be less than 0.001, and the required depth to attain this is a function of the value of p used as Equation D.6:

$$p^{d} < 0.001 \tag{D.6}$$

$$\Rightarrow d > \frac{ln0.001}{ln(p)} \tag{D.7}$$

When $p = 0.5$, $p = 0.8$, and $p = 0.95$, this expression suggests minimum evaluation depth of $d = 10$, $d = 31$, and $d = 135$, respectively. By extending the depth d of the ranking, we can increase the amount of information taken into account, which enhance precision in the estimations of effectiveness values.

BIBLIOGRAPHY

[1] E. Lee, “Cyber physical systems: Design challenges,” *Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC ’08)*, pp. 363–369, May 2008.

[2] J. Sztipanovits, “Composition of cyber-physical systems,” in *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS ’07)*, (Washington, DC, USA), pp. 3–6, IEEE Computer Society, Mar. 2007.

[3] P. Pederson, “Critical infrastructure interdependency modeling: The survey of U.S. and international research,” tech. rep., Idaho National Laboratory, August 2006.

[4] S. M. Rinaldi, “Modeling and simulating critical infrastructures and their interdependencies,” in *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS ’04)*, Jan. 2004.

[5] N. K. Svendsen and S. D. Wolthusen, “Analysis and statistical properties of critical infrastructure interdependency multiflow models,” in *Proceedings of the IEEE Information Assurance and Security Workshop (IAW ’07)*, pp. 247–254, June 2007.

[6] C. M. Macal and M. J. North, “Tutorial on agent-based modeling and simulation,” in *Proceedings of the 37th Winter Simulation Conference (WSC ’05)*, pp. 2–15, Dec. 2005.

[7] C. M. Macal and M. J. North, “Tutorial on agent-based modeling and simulation part 2: How to model with agents,” in *Proceedings of the 38th Winter Simulation Conference (WSC ’06)*, pp. 73–83, Dec. 2006.

[8] United States Environmental Protection Agency, “EPANET2 User’s manual.” `http://www.epa.gov/nrmrl/wswrd/dw/epanet/EN2manual.PDF`, retrieved Jul. 2011.

[9] J. Lin, S. Sedigh, and A. Miller, “Towards integrated simulation of cyber-physical systems: a case study on intelligent water distribution,” in *Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC ’09)*, (Chengdu, China), pp. 690 –695, Dec. 2009.

[10] J. Lin, S. Sedigh, and A. Miller, “A general framework for quantitative modeling of dependability in cyber-physical systems: A proposal for doctoral research,” in *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC ’09)*, vol. 1, (Seattle, USA), pp. 668 –671, Jul. 2009.

[11] J. Lin, S. Sedigh, and A. Miller, "Modeling cyber-physical systems with semantic agents," in *Proceedings of the 5th IEEE Workshop on Engineering Semantic Agent Systems (ESAS), in conjunction with the 34th IEEE International Computer Software and Applications Conference (COMPSAC '10)*, (Seoul, South Korea), Jul. 2010.

[12] J. Lin, S. Sedigh, and A. Miller, "A semantic agent framework for cyber-physical systems," in *Semantic Agent Systems: Foundations and Applications* (A. Elci, M. T. Kone, and M. A. Orgun, eds.), Springer-Verlag, 2010.

[13] J. Lin and S. Sedigh, "Qualitative and quantitative modeling of reliability for intelligent water distribution networks," *Special Issue on Performance and Dependability Modeling of Dynamic Systems of the International Journal of Performability Engiineering*, vol. 358, no. 10, 2011.

[14] J. Lin, S. Sedigh, and A. Hurson, "An agent-based approach to reconciling data heterogeneity in cyber-physical systems," in *Proceedings of the 20th International Heterogeneity in Computing Workshop (HCW), in conjunction with the 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS '11)*, (Anchorage, AK), May 2011.

[15] J. Lin, S. Sedigh, and A. Miller, "Investigating the application of game theory to resource allocation in cyber physical systems," in *Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS-44)*, (Kauai, HI), Jan. 2011.

[16] J. Lin, S. Sedigh, and A. Miller, "Integrated cyber-physical simulation of intelligent water distribution networks," in *Matlab/Book2* (E. P. Leite, ed.), Intech, 2011, to appear.

[17] J. Lin, S. Sedigh, and A. Hurson, "Ontologies and decision support for failure mitigation in intelligent water distribution networks," in *Proceedings of the 45th Hawaii International Conference on System Sciences (HICSS-45)*, (Maui, HI), Jan. 2012, under review.

[18] J. Lin and S. Sedigh, "An ontology-based framework for decision support in intelligent water distribution networks," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, to be submitted in Aug. 2011.

[19] J. Lin and S. Sedigh, "Quantifying the effectiveness of automated failure mitigation in intelligent water distribution networks," in *7th Track on Dependable and Adaptive Distributed Systems (DADS) of the 27th ACM Symposium on Applied Computing (SAC '12)*, (Riva del Garda, Trento, Italy), to be submitted in Aug. 2011.

[20] T. Rigole and G. Deconinck, "A survey on modeling and simulation of interdependent critical infrastructures," in *Proceedings of the 3rd IEEE Benelux Young Researchers Symposium on Electrical Power Engineering*, (Ghent, Belgium), pp. 27–28, Apr. 2006.

[21] M. de C. Gatti, C. de Lucena, and J. Briot, "On fault tolerance in law governed multi-agent systems," in *Proceedings of the 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS '06)*, May 2006.

[22] B. Bauer and J. Odell, "UML 2.0 and agents: How to build agent based systems with the new UML standard," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 2, 2005.

[23] A. Faza, S. Sedigh, and B. McMillin, "Reliability analysis for the advanced electric power grid: from cyber control and communication to physical manifestations of failure," in *Proceedings of the 28th International Conference on Computer Safety, Reliability and Security (SAFECOMP '09)*, Sep. 2009.

[24] Y. Y. Haimes, B. M. Horowitz, J. H. Lambert, J. R. Santos, C. Lian, and K. G. Crowther, "Inoperability input-output model (IIM) for interdependent infrastructure sectors: theory and methodology," *Journal of Infrastructure Systems*, vol. 11, pp. 67–79, June 2005.

[25] G. Jiang, W. Chung, and G. Cybenko, "Semantic agent technologies for tactical sensor networks," in *Proceedings of the SPIE*, pp. 311–320, Sep. 2003.

[26] J. Liu and F. Zhao, "Towards semantic services for sensor-rich information systems," in *2nd International Conference on Broadband Networks*, pp. 44–51, Oct. 2005.

[27] A. Elci and B. Rahnama, "Consideration on a new software architecture for distributed environments using autonomous semantic agents," in *Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC '05)*, Jul. 2005.

[28] J. E. Kim and D. Mosse, "Generic framework for design, modeling, and simulation of cyber physical systems," *ACM SIGBED Review*, vol. 5, no. 1, pp. 1–2, 2008.

[29] A. Al-Hammouri, V. Liberatore, H. Al-Omari, Z. Al-Qudah, M. S. Branicky, and D. Agrawal, "A co-simulation platform for actuator networks," in *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys '07)*, pp. 383–384, Nov. 2007.

[30] M. W. Bright, A. R. Hurson, and S. Pakzad, "Automated resolution of semantic heterogeneity in multidatabases," *ACM Transactions on Databases Systems*, vol. 19, Jun. 1994.

[31] Y. Jiao and A. R. Hurson, "Mobile agents in mobile data access systems," in *Proceedings of the Confederated International Conferences DOA, CoopIS and ODBASE 2002 On the Move to Meaningful Internet Systems, DOA/CoopIS/ODBASE '02*, (London, UK), pp. 144–162, Springer-Verlag, 2002.

[32] Y. Jiao and A. R. Hurson, "Mobile agents and energy-efficient multidatabase design," in *Proceedings of the 18th International Conference on Advanced Information Networking and Applications (AINA '04)*, vol. 1, pp. 255 – 260, 2004.

[33] S. Blackburn, *The Oxford Dictionary of Philosophy.* Oxford University Press, 1996.

[34] T. R. Gruber, "Towards principles for the design of ontologies used for knowledge sharing," *International Journal Human-Computer Studies*, vol. 43, Nov. 1995.

[35] J. J. Jung and J. Euzenat, "Towards semantic social networks," in *Proceedings of the 4th European Semantic Web Conference*, pp. 267–280, 2007.

[36] I. Cantador and P. Castells, "Multilayered semantic social network modeling by ontology-based user profiles clustering: Application to collaborative filtering," in *Proceedings of the 15th International Conference on Managing Knowledge in a World of Networks*, Springer Verlag Lectures Notes in Artificial Intelligence, 2006.

[37] J. Golbeck and M. Rothstein, "Linking social networks on the web with FOAF: A semantic web case study," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, Jul. 2008.

[38] P. Mika, "Flink: Semantic web technology for the extraction and analysis of social networks," *Journal of Web Semantics*, vol. 3, 2005.

[39] H. Alani, S. Dasmahapatra, K. O'Hara, and N. Shadbolt, "Identifying communities of practice through ontology network analysis," *IEEE Intelligent Systems*, vol. 18, no. 2, 2003.

[40] V. Paola, N. Roberto, and C. Alessandro, "A new content-based model for social network analysis," in *Proceedings of the 2th IEEE International Conference on Semantic Computing*, Aug. 2008.

[41] E. Miguela, P. Patro, K. E. Brown, Y. R. Petillot, and D. M. Lane, "Semantic knowledge-based framework to improve the situation awareness of autonomous underwater vehicles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, May 2011.

[42] C. M.Macal and M. J. North, "Tutorial on agent-based modeling and simulation Part 2: How to model with agents," in *Proceedings of the 38th Winter Simulation Conference (WSC '06)*, pp. 73–83, 2006.

[43] L. R. Phillips, H. Link, R. Smith, and L. Weiland, "Agent-based control of distributed infrastructure resources," tech. rep., Sandia National Laboratories, Jan. 2006.

[44] Argonne National Laboratory, "Recursive porous agent simulation toolkit manual." `http://repast.sourceforge.net/docs/tutorial/SIM/index.html`, retrieved Jul. 2011.

[45] K. S. Trivedi and R. Sahner, "SHARPE at the age of twenty two," *SIGMETRICS Performance Evaluation Review*, vol. 36, no. 4, pp. 52–57, 2009.

[46] FMEA-FMECA.com, "Failure mode and effects analysis." `http://www.fmeainfocentre.com/`, retrieved Jul. 2011.

[47] Cambridge Information Group, "Cambridge scientific abstracts." `http://www.csa.com/`, retrieved Jul. 2011.

[48] C. Byrne and S. A. McCracken, "An adaptive thesaurus employing semantic distance, relational inheritance and post-coordination for linguistic support of information search and retrieval.," *Journal of Information Science*, vol. 25, no. 2, pp. 113–131, 1999.

[49] Stanford University, "Protégé." `http://protege.stanford.edu/`, retrieved Jul. 2011.

[50] W. Recommendation, "Owl web ontology language reference." `http://www.w3.org/TR/owl-ref/`, retrieved Feb. 2011.

[51] Isograph, "Fault tree analysis." `http://www.faulttree.org/`, retrieved Jul. 2011.

[52] ReSIST Project, "Resist resilience for survivability in IST." `http://www.rkbexplorer.com/explorer/`, retrieved Jul. 2011.

[53] K. S. Trivedi, D. S. Kim, A. Roy, and D. Medhi, "Dependability and security models(keynote paper)," in *7th International Workshop on Design of Reliable Communication Networks (DRCN '09)*, pp. 11–20, Oct. 2009.

[54] A. Moffat and J. Zobel, "Rank-biased precision for measurement of retrieval effectiveness," *ACM Transactions on Information Systems*, vol. 27, Dec. 2008.

VITA

Jing Lin was born in Chengdu, China on November 5, 1982. In June 2005, she received her BSc. in Communication Engineering from the University of Electrical Science and Technology of China (UESTC), in Chengdu, China. In December 2007, she received her M.S. in Software Engineering from the UESTC. She started working toward her Ph.D. in Computer Engineering at the Missouri University of Science and Technology (formerly the University of Missouri-Rolla) in January 2008, and received the degree by December 2011.