Ontologies and Decision Support for Failure Mitigation in Intelligent Water Distribution Networks

Jing Lin, Sahra Sedigh, and Ali R. Hurson
Missouri University of Science and Technology
{lpg2, sedighs, hurson}@mst.edu

Abstract

One of the greatest benefits offered by cyber-physical systems is the potential for automated decision support, which can increase the intelligence and efficacy of environmental management. Agent-based modeling can facilitate the application of cyber infrastructure to environmental decision support, by abstracting physically-distributed and communication and control into the operation of one or more agents. Ontologies can capture the semantics of the operation of both physical and cyber components of an environmental management system, respectively; while reflecting interactions between the two. As such, they can serve as a basis for automated reasoning by agents for intelligent decision support. In this paper, we illustrate and validate the use of ontologies in decision support for an intelligent water distribution network. The focus of the decision support is on identification and mitigation of failure; the aim is dependable distribution of potable water.

1. Introduction

In environmental management systems, intelligent decision support is utilized in hopes of facilitating more efficacious use of scarce environmental resources, particularly by adapting resource allocation and system operation to user demand, resource availability, and environmental conditions. Intelligent environmental decision support systems (EDSSs), which make use of cyber infrastructure for computing services such as automated reasoning, have been developed to this end. The physical infrastructure that supports environmental management is monitored, controlled, and guided by the cyber infrastructure; creating a cyber-physical system (CPS) where embedded computing and communication capabilities are used to streamline and fortify the operation of the physical system [1], [2]. Real-time data collected by sensors from the physical infrastructure and its environment creates situational-awareness and enables intelligent decision support and control. As an example, automated reasoning can be used to determine whether a power grid is operating within safety thresholds, and to determine appropriate settings for power electronics devices that control the flow on transmission lines.

Design and development of a reliable, effective, and efficient CPS is contingent upon accurate characterization of the operation of the physical and cyber infrastructures, respectively; as well as the interaction between the two. This is especially critical in the design of EDSSs, where decision support and actuation are needed for allocation of physical resources. Among existing techniques for modeling and representation of complex systems, agent-based modeling holds promise for accurate characterization of the operation of the cyber and physical infrastructures in one integrated model, with high fidelity; and can capture interdependencies among the two infrastructures. This results from the ability of an agent-based model to encapsulate diverse component attributes within a single agent, while accurately capturing the interaction among autonomous, heterogeneous agents that share a common goal achieved in a distributed fashion.

The agent is the key element in our CPS model, in terms of implementing intelligent decision making; which requires autonomous conversion of raw data sensor data to information with the semantics and syntax required by the decision support algorithms. Ontologies can facilitate knowledge extraction by capturing (and presenting) the terminology for and semantic relationships among entities in a given application domain, simplifying the task of an agent responsible for creating useful information from raw data. The semantic relationships that can be captured by an ontology far exceed the “is-a” links typical of taxonomies, whose use is limited to classification of an entity per a predetermined hierarchy.

In this paper, we propose and validate an EDSS where ontologies are utilized to classify failure events and determine appropriate countermeasures to mitigate their effects. An intelligent water distribution network
(WDN) serves as a case study; however, the methodology proposed is applicable to a broad range of environmental management systems.

In a cyber-physical WDN, physical components, e.g., valves, pipes, and reservoirs; are coupled with hardware and software that supports intelligent water allocation. Fig. 1 depicts an example. The primary goal of WDNs is to provide a dependable source of potable water to the public. Information such as demand patterns, water quantity (flow and pressure head), and water quality (contaminants and minerals) is critical in achieving this goal, and beneficial in guiding maintenance efforts and identifying vulnerable areas requiring fortification and/or monitoring. Sensors dispersed in the physical infrastructure collect data, which is processed and interpreted by the decision support entities of the cyber infrastructure. Distributed algorithms, e.g., game theory, are used to determine appropriate settings for hardware controllers that are used to manage the allocation (quantity) and chemical composition (quality) of the water. Automated reasoning and computation are implemented through software executing on multiple distributed computing devices. This software is represented by the agents in our model, each of which is capable of perceiving its environment, acting on that perception, and communicating with other agents.

![Figure 1. Intelligent water distribution network](image)

In this paper, we present ontologies that guide the perception and reasoning of agents in the cyber infrastructure of a WDN. We further illustrate and validate the use of these ontologies in classification and mitigation of failure events. In terms of the design science research methodology proposed by Peffers, et al. [3], we commenced our research efforts towards the design and development of an EDSS for water management by problem-centered initiation. The motivation for and introduction to our approach was described earlier in this section, where we also inferred and defined the broad and specific objectives of the EDSS; i.e., efficient and dependable water distribution, and failure classification and mitigation, respectively. Design and development of the EDSS is presented in Sections 3, 4, and 5; as is demonstration of its application to a WDN. Section 6 describes validation of the proposed approach through simulation. Section 7 concludes the paper.

2. Related work

Challenges in the design of EDSSs for management of critical infrastructure have been investigated in several studies; a representative subset is presented in this section. Model formulation of support systems for interactive decision making based on environmental data is discussed in [4]. Strategic decision support for the services industry is described in [5]. Application case studies on the utilization of EDSSs include, but not are limited in contextual role-based access control authorization [6], recognition of driving events [7], and guidance on design of transmission lines [8].

Agent-based modeling and simulation (ABMS) for characterizing systems composed of autonomous, interacting agents is discussed in [9], which points out that ABMS is particularly beneficial to decision-making. The utility of agent-based models for representation of distributed complex systems has been investigated in [10]. The work in [11] adopts a distributed multi-agent architecture to analyze observed information in real-time, facilitating adaptation of a multi-agent system to the evolution of its environment. A multi-agent approach to system assurance is articulated in [12], which demonstrates how to determine the relative importance of any agent.

Ontologies can be considered a method for modeling and presenting knowledge; they are frequently used to organize and represent knowledge in artificial intelligence. Experts differ on their definitions of “ontology,” but every definition encountered by the authors concurs that an ontology is a representation of entities and the relationships among them [13]. A definition given in [14] 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 [15, 16]. Two well-known examples are Friend of a Friend (FOAF) [17] and Flink [18], which have been used to analyze social networks, discover communities of practice [19], and explore “hot” topics [20]. Recent applications of ontologies in automated reasoning include the use of ontologies in improving situational awareness [21].
3. Agent-based model for WDN operation

The very first step in constructing a CPS model is to identify the major functional components of the system. Fig. 2 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 (CCM) 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 2. Functional model of transport CPS](image)

Fig. 2. Functional model of transport CPS

Fig. 3 depicts the instantiation of the functional model of Fig. 2 for a WDN.

![Figure 3. Functional model of a WDN](image)

The functional framework of Fig. 3 serves as the basis for qualitative modeling of the WDN, which is carried out with the goal of capturing the interaction between the cyber and physical infrastructures. We use the Uniform Modeling Language (UML) to represent the model, due to the precise semantics offered by this formal specification language. A detailed demonstration of the use of UML for the specification of an agent-based system is presented in [22].

A use case diagram specifies the major functionalities for qualitative system analysis. Each use case captures the interaction of a number of external actors with the system towards the accomplishment of a specific goal. Capturing this functionality facilitates the tracing of information flow through the CPS. Fig. 4 depicts the actors and use cases involved in a typical WDN. The blue circles emphasize the sources of cyber or physical heterogeneity, a significant challenge that has been investigated in [23]. This use case diagram can be readily generalized to other CPSs whose main goal is management of a physical commodity.

![Figure 4. Use case diagram for a WDN](image)

The agent is the actor in the use case diagram, and associated with the decision support algorithm. The agent queries the various data sources available to the EDSS, e.g., sensor networks or databases with information on past events. For simplicity, only use cases associated with one agent are shown in Fig. 4; all other agents are associated with similar use cases.

As shown in Fig. 4, 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. This heterogeneous data 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 database 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 Summary Schema Model (SSM) is an advanced semantic processing model that supports the semantic service by extracting similar semantics of access terms from the underlying local database and forming a hierarchical structure to reduce redundant information. It can provide 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) metadata 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. Utilization of SSM to reconcile data heterogeneity in WDN decision support has been described in [23].

4. Classes in WDN ontology framework

On the basis of the functional components of Fig. 3 and the use case diagram of Fig. 4, in this section we present the building blocks of the ontology framework - the classes. We use Protégé 4 [24] as the platform for creation of the WDN ontology.

4.1. WDN ontology class

The topmost classes of the WDN are shown in Fig. 5, according to the functionalities identified in Figs. 3 and 4. The class “Thing” is the set containing all the subclasses, under which all other classes are defined.

![Figure 5. Topmost classes of the WDN](image)

Classes can be organized into a superclass-subclass hierarchy, which is quite similar to a taxonomy. We adopt OWL-DL [25] 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 failure type and mitigation technique classes we have defined in OWL-DL for the WDN are presented in Figs. 6 and 7, where the superclass-subclass relationships are clearly depicted.

![Figure 6. Failure type class](image)
The failure type class is created based on FMEA [26] and fault tree analysis [27]. The Computer-System-Vulnerability class is adopted from an existing ontology developed by the Resilience for Survivability group [28]. The mitigation technique database is designed to address a broad range of failure types, and makes reference to [29].

4.2. 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. 8. 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. 8.

As an example, shown in Fig. 9, a “PipeOverload” can be automatically identified as a type of failure of the pipe. 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 [30].
5. Automated failure mitigation

In OWL, properties describe relationships between classes or instances of classes and the subclass can also be viewed as individuals of the superclass. The two main types of properties in OWL-DL are “object” and “datatype.” Our focus is on reasoning about the behavior of the system, and as such, we focus on the first category.

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 [24]; and to facilitate automated reasoning.

The object properties we have defined for the WDN ontology are shown in Fig. 10.

![Figure 10. Top object properties](image)

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 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, which have been discussed earlier in this paper. Fig. 11 shows the characteristics and the description of the domain and range of property “hasCommandOver”.

![Figure 11. Characteristics and description of a property](image)

Two types of arcs appear in Fig. 12 – 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 hasAdvancedComputation of the “Continuous Flow”. Similarly, “WaterNetwork” hasFlow of the “ContinuousFlow”, “SensorNetwork” hasMonitorOf the “ContinuousFlow”, “CyberEntity” hasIntegrityDataOf “DataIntegrityCheck”, “Computer” hasCommandOver “MitigationTechniqueDatabase”, “Computer” hasSuggestionToComputation for “DecisionMakingSystem”, and the “Computer” can send three different types of control command to the valve and the pump (subclasses of the actuator), including “hasIncreaseOriginalValue”, “hasDecreaseOriginalValue” and “hasMaintainOriginalValue”.

Fig. 13 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 “hasMitigationIndentification” 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.

Figure 12. Map of object properties

Figure 13. Classes and object properties relevant to failure mitigation

6. Validation of failure mitigation

In this section, we present and analyze an empirical test case used to validate the automated failure mitigation technique of Section 5. The integrated cyber-physical WDN simulator developed in [31] 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.

6.1. Normal operation

The topology assumed in EPANET for the physical infrastructure is shown in Fig. 14. 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. 15.

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 (elevation + pressure head) of water in the reservoir 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. From the simulation results, shown in Fig. 16, when no error occurs in the simulation, the status of the nodes and links in each time span can be displayed in EPANET. 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.

6.2. Failure scenario

Fault injection was carried out to validate the automated failure mitigation technique. The specific 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 an energy parameter associated with elevation. If the elevation of the reservoir cannot 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 both pipes and pumps. EPANET reflects this failure by displaying a warning.
message with information about overloaded links, as shown in Fig. 19. The complete warning message has been omitted in the interest of brevity.

Figure 19. EPANET warning message

Detection of this failure by EPANET should trigger automated failure mitigation, using the ontologies of Section 4 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. 19 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. 6. The reasoning procedure that leads to this determination is shown in Fig. 20.

Figure 20. Failure classification

Once the failure type is identified, the associated countermeasure is determined using the mitigation ontology of Fig. 7. A snapshot of the resulting mitigation object is shown in Fig. 21.

The automated reasoning procedure (an OWL-DL script), from “failure identification” to “mitigation technique identification” is depicted in Fig. 22.

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 by MATLAB, and sent to EPANET as the input file shown in Fig. 23. The countermeasure does not affect the “Pattern” (an option relevant to time specification), and this value is left blank in the automatically-generated file.

Figure 21. Characteristics and description of mitigation technique

Figure 22. Automated reasoning procedure for failure mitigation

Figure 23. Countermeasures provided by MATLAB to EPANET

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. 15 led to results identical to those depicted in Fig. 16, verifying the effectiveness of the countermeasure in restoring normal operation.
7. Conclusion

Facilitation of intelligent decision support is the primary factor motivating the use of CPSs in environmental management. We proposed and illustrated the use of ontologies in decision support for an intelligent WDN. An agent-based model representing the major functionalities of the WDN served as the basis for constructing ontologies for identification and mitigation of failures. Empirical validation with an integrated cyber-physical simulator has successfully validated the approach.

8. References