Ontology-Based Approach to Context Representation and Reasoning for Managing Context-Sensitive Construction Information

Han-Hsiang Wang[1]; Frank Boukamp, A.M.ASCE[2]; and Tarek Elghamrawy[3]

Abstract: The construction industry is an information-intensive industry and heavily relies on documents, including physical and virtual documentation and models, to exchange context-sensitive information among different project participants. Many research efforts have been made to help manage construction information; however, few of them considered the context-sensitive nature of the information. In this paper, the writers propose a new approach to facilitate the management of context-sensitive construction information that is stored in different textual documents. The approach addresses the context-sensitive nature of construction information by representing contexts in ontologies and by using contexts as indices of the information. The approach also presents a reasoning mechanism that leverages the semantically rich features of ontologies to reason about contexts to evaluate their applicabilities. Two case studies were conducted, and the results showed the proposed approach can effectively retrieve, classify, and manage construction information. Finally, the writers discuss the limitations of the proposed approach and future research directions. **DOI: 10.1061/(ASCE)CP.1943-5487.0000094.**

CE Database subject headings: Construction management; Knowledge-based systems; Information management.

Author keywords: Construction management; Knowledge management; Context; Ontology.

Introduction

The construction industry is an information-intensive industry with a relatively low level of information technology (IT) integration. A high percentage of the information is exchanged by using documents, which include physical and virtual documentation, such as design drawings, specifications, contracts, change orders, field reports, and requests for information, as well as virtual models. The information-intensive nature of the construction industry requires construction staff to have easy access to different project documents. However, information stored in textual documents, independent of whether they are paper-based or electronic, cannot be easily accessed and navigated without appropriate document classification mechanisms (Haimes 1994). In addition, the dynamic and unique nature of the construction industry requires the ability to integrate information from different sources and in different data formats for the improvement of the construction processes (Caldas et al. 2002). This, as well as the challenges in coordinating the information of on-site tasks and the information of office work necessitate the implementation of intelligent ways to support knowledge management in construction organizations (Aziz et al. 2006).

Much of the information being generated in architecture, engineering, construction, and facility management (AEC/FM) systems is context-specific, i.e., it applies or relates to a specific situation or a specific category (Caldas et al. 2002). Contexts, in this research, are defined as situations in which a construction entity acts or exists; whereby an entity can be a person involved in the construction, an action taken to complete a job, a component built in a project, or a resource utilized to perform the action or build the component. Despite the semistructured nature of the information contained in many types of construction-related documents, the data or information always contain references to construction contexts that are associated with it (Zhu et al. 2007).

The correct identification of contexts is important for the integration and utilization of construction information contained in different data and information repositories. Indexing the information items, such as construction reports and lessons learned, by using context descriptors, i.e., indexes that describe the contexts to which the information pertains, can improve information storage and retrieval (Boukamp and Ergen 2008). A substantial part of the construction context identification, modeling, and reasoning is intuitively determined on the basis of the experience and wisdom of the construction personnel. Scherer and Reul (2000) argued that if the context knowledge from all project documents could be retrieved, a great amount of knowledge and construction experience would be available for everyone and could be used in planning and forecasting future projects.

Current technologies used for document management, such as project websites and document management systems, do not provide direct support for information integration owing to many limitations, among which are the large number of documents and the discrepancy of these technologies in handling contextual

[1]Ph.D. and Adjunct Assistant Professor, Dept. of Civil Engineering, National Chiao Tung Univ., 1001 University Rd., Hsinchu, 300 Taiwan. E-mail: hanhsiang.wang@gmail.com

[2]Ph.D. and Senior Lecturer, School of Property, Construction and Project Management, Royal Melbourne Institute of Technology, Building 8, Level 8, Room 56, 360 Swanston St., GPO Box2476, Melbourne, VIC 3001, Australia (corresponding author). E-mail: frank.boukamp@rmit.edu.au

[3]Ph.D. Candidate, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, Newmark Civil Engineering Laboratory 3147, 205 N. Mathews Avenue, Urbana, IL 61801. E-mail: telgham2@illinois.edu

Note. This manuscript was submitted on December 3, 2009; approved on October 8, 2010; published online on October 12, 2010. Discussion period open until February 1, 2012; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Computing in Civil Engineering***, Vol. 25, No. 5, September 1, 2011. ©ASCE, ISSN 0887-3801/2011/5-331–346/$25.00.

information (Soibelman et al. 2008). Most of the search mechanisms applied in these systems are primarily based on keywords. Some keyword-based systems have been developed and successfully deployed, such as web-based keyword tagging (i.e., content annotating) systems for social networking and digital image sharing (Sørensen et al. 2010). Although such systems can benefit from information discovery and retrieval, they lack a predefined structure for tagged keywords and, thereby, may result in noisy metadata (Furnas et al. 2006). In addition, keyword-based searches have some limitations. First, words alone cannot represent the semantically rich nature that they have in different situations. Moreover, keyword-based searches are often either too specific and exclude documents that would have been of interest, or they are too general and include too many documents, many of which are not of interest (Boukamp 2006; Soibelman and Caldas 2006).

Instead of relying on keyword-based search mechanisms to retrieve documents, Zhu et al. (2007) argued that information retrieval processes can be improved by using construction context metadata models to index information. This paper builds on this argument and proposes to index construction information by using construction context descriptors to define the applicability of the information. Whether a piece of information should be retrieved for a given situation then depends on whether the applicability conditions of the context descriptors indexing the information are satisfied. However, a major challenge in such a context-based information retrieval mechanism is that currently there is no reasoning mechanism that can leverage the semantically rich nature of contexts to reason about the contexts' applicabilities while also reasoning about the applicabilities of implicit contexts. This research aims to address this challenge by developing an ontology-based framework that enables context representation and reasoning that supports indexing, retrieving, and accessing information through leveraging a context ontology containing context descriptors and their semantic relationships.

In this paper, the writers focus on the management of context-sensitive construction information that is stored in textual documents. In addition, developing a comprehensive construction context ontology is a cumbersome task (Wang and Boukamp 2008) that is out of the scope of this research. However, to illustrate the value of the proposed context ontology representation and reasoning approach, prototypical ontologies by using Web Ontology Language (OWL) [World Wide Web Consortium (W3C) 2004] were developed in this research. These ontologies are then used in Java-based prototype systems that implement the developed reasoning mechanism to support the indexing and management of construction information. Test cases to validate the reasoning mechanism and to assess the usefulness of the proposed approach for indexing construction documents were implemented.

Research Vision

The vision in this research is to retrieve construction information on the basis of the applicable context descriptors and related descriptors that can be inferred, rather than on the basis of mere keywords. Fig. 1 illustrates the overview of the research vision. First, the context descriptors describing the contexts/situations to which information pertains are identified and acquired. These context descriptors are formalized by being represented as classes in context ontologies (details are given in a later section). Then, these context descriptors are used to index the information (as shown in the upper-left part of Fig. 1). For example, a request for information (RFI) related to missing information of reinforcement in a concrete column can be indexed through context descriptors such as *Rebar*, *Concrete*, and *Column*, which are also represented as classes in a context ontology. Every context descriptor in the ontologies can carry an applicability value, which represents whether the related context applies to the situation in the current project for which information is to be retrieved. All the context descriptors carry the applicability value of *possibly applicable* as their default value.

Following the process of representing context descriptors in a context ontology, the reasoning mechanism can then be deployed to deduce implicit knowledge from explicit facts according to the semantic relationships defined between different context descriptors in the context ontology, thus further refining a context description provided by a user [as shown in the "(2) Reasoning

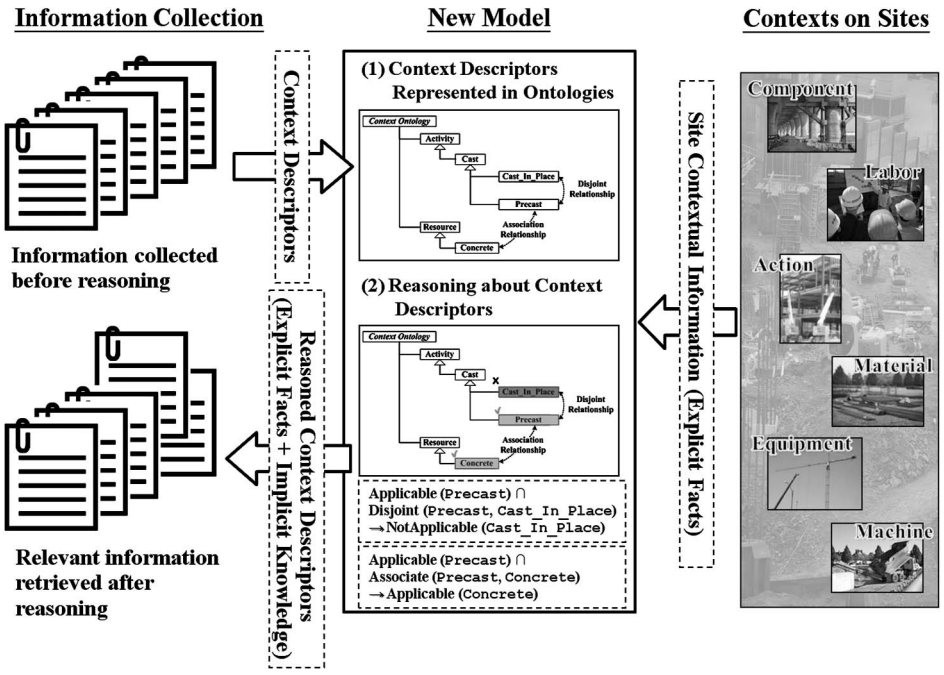

Fig. 1. Illustration of the research vision (photos courtesy of the writers)

J. Comput. Civ. Eng. 2011.25:331-346.

about Context Descriptors" box of Fig. 1]. For example, the reasoning mechanism can deduce that the machine *Crane* (implicit knowledge) is being used if a user specifies that the construction action *Pour* is being conducted on the material *Concrete* using the equipment *Bucket* (explicit facts). Then, an information retrieval query can be updated by incorporating the new context descriptor *Crane* in addition to the user-specified context descriptors *Pour*, *Concrete*, and *Bucket*. The explicit facts represent the real-time contextual information collected on sites through either users' observation or automated reality capture technologies, such as radio-frequency identification (RFID) (Boukamp and Ergen 2008). They can also be based on contextual information provided by the user, no matter whether it is a description of real-time, as-planned, or as-built construction contexts.

The applicability values of the context descriptors in the ontologies are updated as *applicable* or *not applicable* if the explicit facts and implicit knowledge are found and inferred as *applicable* or *not applicable*, respectively. In the previous example, the context descriptor *Crane* in the context ontology would now carry the applicability value of *applicable* rather than the default value of *possibly applicable*. Finally, the updated context descriptors can be used to retrieve context-relevant information by checking for information items that are indexed by using context descriptors or conjunctions of context descriptors that are found *applicable* or *possibly applicable* in the ontologies (as shown in the lower-left part of Fig. 1).

For instance, if the user explicitly searches for applicable specification information related to *Pour Concrete Column*, the information retrieval system should return additional information related to *Place Concrete* because the context descriptors *Pour* and *Place* should be defined as equivalent classes in the context ontology when related to concrete. Moreover, documents that are indexed with the context descriptors *Concrete Slab* and *Concrete Foundation* should be ignored because the context descriptors *Column*, *Slab*, and *Foundation* should be defined as disjointed classes in the context ontology. Finally, if the user broadens the search to include *Pour Concrete Building Component*, the reasoning mechanism should return documents that are indexed with the context descriptors *Column*, *Slab*, and *Foundation* because these context descriptors should be defined as subclasses for the class *Building Component*. The representation and reasoning mechanism, including the propagation of context descriptors' applicabilities, are explained in detail in later sections of this paper.

Research Background

The research background targets four topics related to the proposed framework. The first topic reviews information systems developed in the construction industry and discusses the hurdles of managing project information. The second topic discusses context awareness, modeling, and reasoning and lists context models that have been developed in the construction industry. The third topic discusses ontological modeling and elaborates on existing context ontologies. The fourth and last topic discusses the Web Ontology Language (OWL) (W3C 2004), which the writers used for modeling the sample ontologies and built the context-reasoning mechanism upon.

Construction Information Systems

Construction information systems, which are computational systems for storing, managing, processing, analyzing, and outputting construction information, have been studied and used for indexing and retrieving construction information in the construction industry. Russell (1993) developed a computerized program to collect and process site information on the basis of traditional superintendents' daily site reports. Kartam (1996) presented an interactive information retrieval system for constructability improvement, where construction lessons were analyzed, classified, and retrieved by using the 16 divisions of the Master Format System defined by the Construction Specification Institute (CSI). The system included alternative means to information access and multiple views of the knowledge base to improve on-site construction processes. Caldas et al. (2005) adopted text mining techniques to develop a text information integration approach to improve the management of project documents and to provide a semiautomatic support for the integration of these documents into a model-based information system. El-Diraby and Zhang (2005) developed a web-based knowledge system to support the representation and utilization of corporate memory in the construction domain. The system used an ontology for building construction to create and retrieve semantically related construction reports and meeting agendas. A prototype ontology was used as the interoperability platform for a multiagent architecture to support semiautomated generation of various corporate construction reports. Zhu et al. (2007) demonstrated the usability of Industry Foundation Classes (IFCs) (buildingSMART 2010) for constructing a metadata model for RFIs that enhances the retrieval of RFI-related information.

This previous research showed that retrieving and reusing construction information relied on classification systems to categorize the information, such as the classification with 13 divisions adopted in Caldas et al. (2002) and the BCTaxo classification with five major root divisions developed in El-Diraby and Zhang (2005). Classification systems not only enable construction information to be systematically organized to facilitate reuse of the information and retrieval of the documents containing the information, but also allow information or documents to be rearranged in different dimensions in which users are interested. Moreover, classification systems help cluster information of same or similar nature together so that when a piece of information is found useful, another in the same classification division may be identified as useful as well. This feature played an important role in information and document retrieval in the aforementioned research efforts.

Classification systems, however, cannot represent other semantic relationships between information. For example, two pieces of information that have the same meaning but are described differently at best are represented as sibling classes in the same level of a hierarchy, but such a representation cannot reflect the fact that their meanings are identical. In addition, another semantic issue is that classification systems ignore the notion of ambiguity of information objects according to Lai (2006). Therefore, the research presented in this paper builds upon the aforementioned feature but goes beyond using hierarchy relationships that are used in traditional classifications to address the semantic issues of classification systems. Specifically, other semantic relationships definable between context descriptors in a context ontology are leveraged to identify clusters of information relevant to the search criteria defined by users.

Context Awareness, Modeling, and Reasoning

Many researchers in the computer science domain have developed their own definitions of contexts. The most prevalent definition was given by Dey and Abowd (2000). They defined contexts as "any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application" or "situational information" for short. In the AEC/FM domain, the notion of contexts was adopted in different research work and different definitions of them were given. Kiliccote

J. Comput. Civ. Eng. 2011.25:331-346.

(1994) defined contexts as collections of classes that represent basic vocabulary, such as wall, floor, etc., of design specifications, and a similar definition was adopted in the representation of construction specifications by Boukamp (2006). Aziz et al. (2006) defined contexts of a user as different parameters that characterize who and where the user is and what the user is doing, such as the users' profile, preference, location, task, and existing project conditions. Wang and Boukamp (2008) defined contexts as specific project conditions on site, such as components built, activities performed, and resources used.

Dey and Abowd (2000) also defined context awareness as a feature of a system that the system can utilize contexts to provide relevant information and/or services to the user. Context awareness has been drawing much attention among researchers because it is important for pervasive computing environments and human-computer interaction, especially when users' contexts are changing constantly and rapidly (Wang et al. 2004). In the AEC/FM domain, context awareness means identifying and having knowledge about specific contexts in which an activity takes place or a component is situated (Boukamp and Ergen 2008). Awareness of contexts in which a construction process takes place is important for managing construction projects because the contextual information can help determine information relevant to performing a specific task, such as inspecting a building component or generating and retrieving context-related documents. In addition, the correct identification of contexts will be required to file information under correct context indexes, enabling valid context-specific information retrieval.

Context modeling of a domain is concerned with specifying all the entities of this domain with relationships between these entities to fully describe the domain context semantics (Feruzan 2007). That is, context modeling is to formalize context descriptors and their semantic relationships. Wang et al. (2004) classified context modeling into both informal and formal modeling. Informal context modeling is often on the basis of proprietary representation schemes that have no facilities to allow reasoning about contexts in a single system or to ease dissemination of knowledge about contexts in different systems. However, formal context modeling commonly employs formal approaches to represent and manipulate contexts to enable the dissemination of and reasoning about contextual knowledge.

A variety of context models ranging from domain dictionaries to specialized taxonomies have been developed in the construction industry. Among them are BS6100 (Glossary of Building and Civil Engineering terms produced by the British Standards Institution); bcXML (an XML vocabulary developed by the eConstruct IST project for the construction industry); Industry Foundation Classes (IFC, developed by the buildingSMART); OmniClass Classification System (OCCS), BARBi (Norwegian Building and Construction Reference Data Library); and e-COGNOS (COnsistent knowledGe maNagement across prOjects and between enterpriSes in the construction domain) (Rezgui 2006; Wang and Boukamp 2007). Most of the context models use classification systems to structure contextual information whereas only a few of them also allow association relationships between contextual information, such as IFC and e-COGNOS. However, to explore the semantics of contextual information, it is insufficient to consider only classifications and association relationships but ignore other semantic relationships between the contextual information. This is where ontological modeling plays a key role. It will be discussed in the "Proposed Framework" section.

Context reasoning is a technique used to automatically deduce implicit knowledge (i.e., deduce applicability of context descriptors) from explicitly given contextual information (i.e., given applicability values of context descriptors). Aziz et al. (2006) presented a context-based information delivery system that captures, interprets, and reasons about workers' contexts, such as location, time, and profile. The system then delivers context-aware information to workers upon which their decision-making can be based. Elghamrawy and Boukamp (2008) discussed the applicability of using ontological reasoning mechanism for retrieving on-site construction problem information. Wang and Boukamp (2009) discussed an ontology-based context-reasoning mechanism for supporting construction safety planning. In this paper, the writers discuss in detail the ontology-based reasoning mechanism underlying the ideas discussed in the last two previously mentioned papers (i.e., Elghamrawy and Boukamp 2008; Wang and Boukamp 2009) and its general applicability to construction information.

Ontology and Ontological Modeling

Ontological modeling is a systematic approach for representing knowledge in ontologies. Gruber (1993) defined an ontology as "an explicit and formal specification of a conceptualization." Specifically, an ontology can model a set of concepts within a knowledge domain and relationships between these concepts; ontological modeling, therefore, is a process to model concepts and relationships into ontologies. Ontological modeling originates from the philosophy domain and is widely adopted in research efforts in the artificial intelligence and computer science domain in the recent 2 decades (Gruber 1993). The main areas in which ontological modeling is applied include communication and knowledge sharing, logic inference and reasoning, and knowledge reuse (Feruzan 2007).

Ontological modeling is a well-suited approach to context modeling and reasoning for two primary reasons. First, context descriptors and their semantic relationships can be easily represented in the form of classes and properties in an ontology. Second, the applicability of a context descriptor used to describe a specific project situation can imply applicability or inapplicability of other context descriptors that are semantically related to the first descriptor. Much of the research work in the computer science domain has adopted ontological modeling for context modeling and reasoning. Korpipää et al. (2004) utilized an ontology to offer scalable representation and easy navigation of contexts for personalizing mobile device applications. Souza et al. (2006) proposed using an ontology to formally represent contexts to improve geospatial data integration and query processing. Wang et al. (2004) and Kim and Choi (2006) proposed ontology-based frameworks for modeling and reasoning about contexts in pervasive computing environments. In the AEC/FM domain, however, related research efforts applying ontological modeling to context modeling and reasoning is sparse. The e-COGNOS project was the first project to deploy a domain ontology for knowledge management and context modeling and reasoning in the construction industry (Lima et al. 2005). Aziz et al. (2005) used an ontology to represent and deliver contextual information, such as location, time and profile. Dolenc et al. (2007) developed an ontology framework, which includes four ontologies: business process ontology, organizational ontology, service ontology, and resource ontology, for modeling contextual information in the InteliGrid project.

Web Ontology Language (OWL)

To create a context ontology, an ontology language is required to provide formal syntactic structure and modeling rules with which contexts can be represented. Ontology languages allow users to explicitly formalize and conceptualize their domain knowledge (El-Diraby et al. 2005; Lima et al. 2005; Rezgui 2006). Antoniou and van Harmelen (2004) pointed out that ontology languages should equip with the following features: a well-defined syntax,

J. Comput. Civ. Eng. 2011.25:331-346.

efficient reasoning support, formal semantics, sufficient expressive power, and convenience of expressions.

The most common ontology languages include Resource Description Framework (RDF) and RDF Schema (RDFS), DAML + OIL (DARPA Agent Markup Language + Ontology Inference Layer), and Web Ontology Language (OWL) (W3C 2004). RDF provides data model specifications and XML (Extensible Markup Language)-based serialization syntax for ontological modeling; RDFS provides specifications of class and property hierarchies for RDF. DAML+OIL is a combined ontology language effort of DAML of the United States and OIL of Europe in 2000. It builds upon RDF/RDFS and provides more powerful modeling capability. OWL is a specification by the World Wide Web Consortium (W3C 2004) and serves as a fundamental component of the Semantic Web initiative (Antoniou and van Harmelen 2004). OWL is based on the DAML+OIL language and therefore, has many features of DAML+OIL, such as adopting RDF as the modeling language to define ontology vocabularies and using XML-based RDF syntax for representing information (Bechhofer et al. 2004). OWL is divided into three expressiveness-increasing sublanguages: OWL-Lite, OWL-DL (Description Logic), and OWL-Full. OWL-DL is most often used because it provides strong expressiveness without losing computational reasoning efficiency and can exploit the considerable body of description logic reasoning that exists (Bechhofer et al. 2004; Wang et al. 2004).

The main modeling primitives of RDF/RDFS concern the organization of vocabularies in typed hierarchies: class and property subsumption relationships, domain and range restrictions, and classes' instances. However, a number of important features are missing. Antoniou and van Harmelen (2004) listed the following expressiveness discrepancies: range restrictions, disjointedness of classes, combinations of classes, and cardinality restrictions. OWL is an extension of RDFS, in the sense that OWL uses the RDF meaning of classes and properties (*rdfs:Class*, *rdfs:subClassOf*, and other rdfs syntax) and adds language primitives to support the richer expressiveness required and to overcome the previous discrepancies.

Chen et al. (2003) pointed out three advantages brought about by using OWL to define ontologies for ontological applications: (1) better expressiveness than other ontology languages; (2) backing of a well-known and regarded standards organization; and (3) promising opportunities for expanding current applications owing to the emergence of ontology inference engines in support of OWL. In this research, the writers adopt OWL to leverage its powerful expressiveness upon which the proposed context-reasoning mechanism can exploit and build.

Proposed Framework

The framework proposed in this research presents a systematic approach to the representation of and reasoning about context descriptors on the strength of ontological modeling and aims at supporting the retrieval of construction information on the basis of their applicable context descriptors and related descriptors that can be inferred. Although different approaches to develop ontologies have been proposed and discussed (Darlington and Culley 2008), each has its own use depending on its application and one approach adequate for an application domain does not mean it is adequate for another domain (Breitman et al. 2006). For example, the approach articulated in Darlington and Culley (2008) cannot satisfy the need of this research as it does not mention the role logical relationships play in an ontology, which is a significant part in the proposed reasoning mechanism. Therefore, the writers first discuss the representation of context descriptors and explain the steps of representing descriptors in ontologies in this section. The discussion on how context descriptors are represented is significant as the representation aims for supporting the later proposed reasoning mechanism, and improper representation can undermine the following reasoning process by resulting in incorrect and potentially useless reasoning results. After this discussion of representation, the writers propose a reasoning mechanism in which reasoning rules are established and discuss how context descriptors can be reasoned about by deploying these reasoning rules.

Representation of Context Descriptors

OWL-based ontologies are the basis for the representation of context descriptors that can be used to describe the situations to which the construction information is applicable. These ontologies also allow representing the semantic relationships between identified context descriptors. The writers suggest adopting the following two steps to represent context descriptors in ontologies: (1) store context descriptors in a formal structure; and (2) represent the structure of context descriptors in ontologies. The details are illustrated as follows:

Step 1: Store Context Descriptors in a Formal Structure
How to determine context descriptors to be represented, which is out of the scope of this research, depends on users' need and the application domain for which the context descriptors are required. Users can refer to existing construction ontologies, data models, and taxonomies, such as IFC (buildingSMART 2010) and OmniClass (OCCS 2010), for terms that can be used as context descriptors. If context descriptors have to be extracted from text documents, some semiautomatic approaches, such as Text2Onto (Cimiano and Völker 2005) and TerMine, can be deployed and an application of one of the approaches was detailed in Wang and Boukamp (2008).

When context descriptors are initially identified, they need to be formally structured to support later reasoning about them. To achieve this goal, two substeps are suggested: (*a*) use classifications to group context descriptors in hierarchies; and (*b*) define associations to represent other semantic relationships between the represented context descriptors.

Step 1-a: Use Classifications to Group Context Descriptors in Hierarchies Context descriptors are first structured in the form of classifications to use class hierarchies to specify subsumption relationships among the context descriptors. First, modelers have to determine the groupings for context descriptors. For example, if context descriptors *Wall*, *Column*, *Slab*, and *Concrete* have been identified, a new grouping context descriptor *Building Component* that groups the descriptors *Column*, *Slab*, and *Wall* is defined. The grouping context descriptors are then used as the main classes, and all other descriptors under the grouping descriptors become subclasses. For example, three classes *Column*, *Slab*, and *Wall* that represent the three respective context descriptors can be defined as subclasses of the class *Building Component*, which represents the grouping context descriptor.

The representation of context descriptors in class hierarchies should be flexible. That is, a new generalized class should be added into a classification if it can make the classification more comprehensive and structured. For example, the following context descriptors belong to the grouping context descriptor *Building Component*: *Bearing Wall*, *Collar Beam*, *Drywall*, *Floating Wall*, *Joist*, *Nonbearing Wall*, *Parapet*, *Retaining Wall*, and *Tail Beam*. To structure these descriptors, it is useful to define two new classes *Beam* and *Wall* as subclasses of the main class *Building Component* instead of representing all the context descriptors as direct

J. Comput. Civ. Eng. 2011.25:331-346.

subclasses of the main class *Building Component* in the classification. The new class *Beam* can act as a superclass of the classes *Collar Beam*, *Joist*, and *Tail Beam*, and the new class *Wall* can act as a superclass of the others. This change is beneficial from the representation viewpoint, as adding these new classes will improve the structure of the context descriptors and reduces the number of subclasses on one hierarchy level by distributing them into lower, more specific hierarchy levels. This becomes especially important when the number of the subclasses in a class is too large to be easily maintained and manipulated. Additionally, this improved structure is beneficial from the reasoning viewpoint, as some documents may contain contextual information only relevant to the introduced generalizations of these context descriptors, i.e., *Beam* and *Wall* in the example. Without the generalized contexts, those relevant documents cannot be retrieved and hence, the reasoning about contexts will be detrimentally affected.

The writers assume in the proposed framework that single inheritance is deployed between context descriptors; that is, each context descriptor can have only one superclass when represented in a classification. This assumption aims to benefit the reasoning mechanism by streamlining the inference process, which is discussed later in the section "Reasoning about Context Descriptors."

Step 1-b: Define Associations to Represent Other Semantic Relationships between the Represented Context Descriptors When the context descriptors are organized in class hierarchies, subsumption relationships are specified between the context descriptors to support reasoning about them. Another type of relationships, i.e., associations, needs to be specified between them as well to enable reasoning about other nonhierarchical relationships between the descriptors. Two main types of associations are used in the proposed framework: nonlogical associations (called *association relationships*) and logical associations (called *logical relationships*).

Association relationships are relationships that connect context descriptors in pairs. When defined, association relationships should be given semantically rich names to facilitate the understanding of how the two descriptors are related (Wang and Boukamp 2008). What association relationships are required depends on what context descriptors are represented in the classification and how users want to represent the relations between them. For example, users can define an association relationship named *uses* to connect the grouping context descriptor *Action* to *Equipment* to represent the fact that "an Action uses a piece of Equipment." In addition, when an association relationship is defined between two context descriptors, the inverse association relationship also can be defined to reversely represent the relation between the descriptors. For example, the fact that "A piece of Equipment is used in an Action" can be represented by connecting the grouping context descriptor *Equipment* to *Action* through an association relationship *isUsedIn*, which is inverse to the relationship *uses*.

To facilitate the determination and presentation of association relationships, the writers propose to use a semantic relationship matrix. This matrix provides a straightforward means to structure context descriptors to be connected and to then help define association relationships for the descriptors. Fig. 2 shows an example of semantic relationship matrix that illustrates how the matrix works. First, context descriptors that are planned to be linked are printed in the cells of both the heading row and column of the matrix to respectively represent their connecting and connected roles, i.e., the subject and object of the association relationships to be defined (in the example, four types of context descriptors are used for illustration: *Building Component*, *Action*, *Equipment*, and *Material*). Thereby, the upper triangular area in the matrix is for representing association relationships that link the connecting descriptors to the connected ones whereas the lower triangular area is for representing their respective inverse relationships. For instance, an association relationship *actsOn* shown in Fig. 2 is for linking the connecting context descriptor *Action* to the connected one *Building Component*, and its inverse relationship *isActedOnBy* works reversely. Finally, the diagonal cells in the matrix are used to represent association relationships linking context descriptors of the same type. For instance, as shown in Fig. 2, association relationships *supports* and *isSupportedBy* can be defined for the context descriptor *Building Component* to represent a structural support relation between two components.

Another occasion in which association relationships can be used is when combined context descriptors exist. A combined context descriptor is a descriptor that is composed of multiple single descriptors. For example, a job context descriptor *Frame Column* is a combined context descriptor because it consists of two single descriptors, the action descriptor *Frame* and the building component descriptor *Column*. These single descriptors are called constituent context descriptors in this research.

Combined context descriptors are important and useful because their applicability can imply the applicability of their constituent descriptors (discussed in the next section). Therefore, if one wants to explore such an implication, association relationships can be used to connect a combined context descriptor to its constituents. Following the previous example, association relationships *comprisesAction* and *comprisesComponent* can be defined to connect the descriptor *Frame Column* to the descriptors *Frame* and *Column*, respectively. In addition, a variant of the semantic relationship

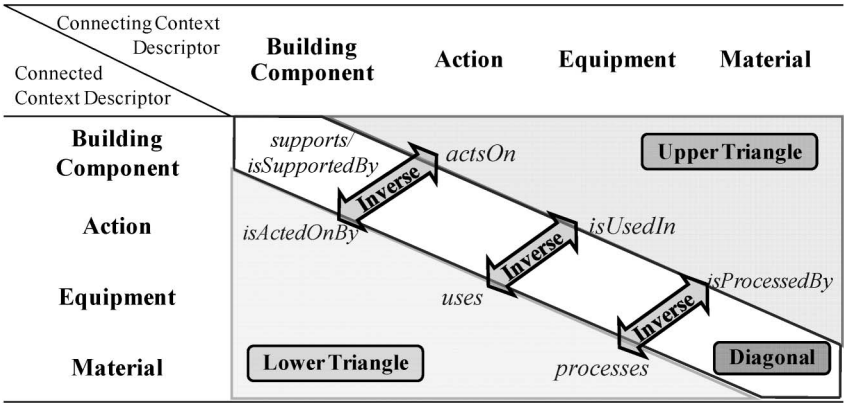

Fig. 2. Semantic relationship matrix for association relationship determination

J. Comput. Civ. Eng. 2011.25:331-346.

matrix can be used to facilitate the determination of the association relationships for combined context descriptors. Table 1 shows the matrix variant for the *Frame Column* and two other combined context descriptors. The heading row and column of the matrix in Table 1 represent combined and constituent context descriptors, respectively; the body of the matrix lists the association relationships connecting the combined descriptors to the constituents. Because unidirectional relations for combined context descriptors are the focus in this research, i.e., the relations from the combined to the constituent descriptors, no inverse association relationships need to be defined. Hence, they will not be shown in the matrix variant.

Logical relationships are equivalent and disjoint relationships, which respectively specify the identicalness and exclusiveness between context descriptors. Equivalent relationships connect context descriptors that are of the same meaning. For instance, a context descriptor *Pour* under the grouping context descriptor *Action* can be declared to be equivalent to a context descriptor *Place* under that same descriptor when related to the context descriptor *Concrete*. On the other hand, disjoint relationships connect context descriptors that exclude one another. An example is that context descriptors *Cast-In-Place* and *Precast* should be connected through a disjoint relationship if only one of them can apply in a given context.

Association relationships together with their inverse relationships and logical relationships are important to help string semantically related context descriptors together bidirectionally to enable propagation of applicability values among descriptors. That is, these relationships will be crucial for the effective knowledge inference from given facts about applicabilities of context descriptors. The interpretation of all the relationships for the purpose of ontological reasoning is discussed in detail in the "Reasoning about Context Descriptors" section.

Step 2: Represent the Structure of Context Descriptors in Ontologies

The writers adopt OWL-based ontological modeling in this research to represent the structure of context descriptors developed in the previous steps into ontologies. The following discusses the basics of how to develop context ontologies in OWL. The specific implementation details, which can be found in W3C (2004), are out of this research's scope.

Two types of elements in OWL, class elements and property elements, form the foundation of representing context ontologies.

- Class elements: Class elements in OWL are the basic unit to represent context descriptors. They are defined by using the tags <owl:class> in the representation. In addition, OWL uses the tags <rdfs:subClassOf> to specify one class as another class's subclass. The following example defines two classes *Concrete_Vibrator* and *Pour* for corresponding context descriptors, which are represented as subclasses of classes *Equipment* and *Action*, respectively.

```
<owl:class rdfs:ID="Concrete_Vibrator">
    <rdfs:subClassOf rdf:resource="#Equipment"/>
</owl:class>
<owl:class rdfs:ID="Pour">
    <rdfs:subClassOf rdf:resource="#Action"/>
</owl:class>
```

- Property elements: There are two types of property elements in OWL.

1. Object properties: This type of properties relates classes/objects to other classes/objects and is defined by using tag <owl:ObjectProperty>. The aforementioned association relationships are represented through object properties in OWL. The following example defines the relationship *isUsedIn* with the class *Concrete_Vibrator* as its domain (i.e., the connecting class) and the class *Pour* as its range (i.e., the connected class). It is inverse to the relation *uses*.

```
<owl:ObjectProperty rdf:ID="isUsedIn">
    <rdfs:domain rdf:resource="#Concrete_Vibrator"/>
    <rdfs:range rdf:resource="#Pour"/>
    <owl:inverseOf ref:resource="uses"/>
</owl:ObjectProperty>
```

2. Data type properties: This type of properties defines attributes of classes and is defined by using the tags <owl:DatatypeProperty>. OWL does not have any predefined data type; instead, it allows to use XML Schema data types. The following example defines that the class *Concrete_Vibrator* has an attribute *characteristics* that must be a character string.

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

In addition to class and property elements, equivalent and disjoint relationships can be easily established between classes by using the tags <owl:equivalentClass> and <owl:disjointClass> in OWL when the classes are being defined. The following example shows the definition of a class *Pour*, in which a class *Place* is specified as the equivalent class.

```
<owl:Class rdf:ID="Pour">
    <rdfs:subClassOf rdf:resource="#Action"/>
    <owl:equivalentClass>
        <owl:Class rdf:ID="Place"/>
    </owl:equivalentClass>
</owl:Class>
```

Similarly, the following example shows the definition of a class *Cast-In-Place*, in which a class *Precast* is specified as the disjoint class.

```
<owl:Class rdf:ID="Cast-In-Place">
    <rdfs:subClassOf rdf:resource="#Action"/>
    <owl:disjointClass>
        <owl:Class rdf:ID="Precast"/>
    </owl:disjointClass>
</owl:Class>
```

Table 1. Semantic Relationship Matrix for Association Relationship Determination for Combined Context Descriptors

Constituent context descriptor	Combined context descriptor		
	Frame column	Pour column	Frame wall
Frame	*comprisesAction*		*comprisesAction*
Pour		*comprisesAction*	
Column	*comprisesComponent*	*comprisesComponent*	
Wall			*comprisesComponent*

J. Comput. Civ. Eng. 2011.25:331-346.

Reasoning about Context Descriptors

This research aims to improve the search for relevant information through automated inference of context knowledge from limited context knowledge provided by a user. Thereby, a context ontology where the context descriptors have been explicitly defined along with their semantics is used for demonstrating the inference of context knowledge. In this research, each context descriptor has its own applicability value to describe whether the situation described by the context descriptor applies to the target context or not. Therefore, the reasoning mechanism developed in this research aims to reason about a context ontology and identify the applicabilities of context descriptors in the ontology according to some context descriptors for which their applicabilities are known. To enable the automation of the inference process, the writers define seven reasoning rules that follow two reasoning premises. The reasoning rules articulate how the inference proceeds in the reasoning mechanism, whereas the reasoning premises provide a basis for building the reasoning rules. In the following section, the writers detail the reasoning premises and rules. Predicate logic (also known as first-order logic) is adopted to formalize, if feasible, the reasoning rules in this paper, providing a symbolic representation to facilitate the understanding of the rules. Predicate logic was used because of its expressive power and the unambiguous nature of its logical semantics (Antoniou and van Harmelen 2004).

Predicates

Before detailing the reasoning mechanism, the writers first define the following predicates that are used in the proposed reasoning premises and rules to describe the context semantics:

1. *subClass(x,y)*: context descriptor x is a subclass of context descriptor y. This predicate is the basic one upon which other predicates, if applicable, will build.
2. *superClass(x,y)*: context descriptor x is a superclass of context descriptor y; therefore, context descriptor y is a subclass of context descriptor x.
3. *associate(x,y)*: context descriptor x connects to context descriptor y through association relationships; that is, descriptors x and y are the connecting and connected descriptors, respectively.
4. *disjoint(x,y)*: context descriptor x is disjoint with context descriptor y; therefore, context descriptor y is also disjoint with context descriptor x.
5. *equivalent(x,y)*: context descriptor x is equivalent to context descriptor y; therefore, context descriptor y is equivalent to context descriptor x.

The first two predicates show the subsumption relationships between two context descriptors. The third predicate presents that one context descriptor connects to another descriptor through association relationships. Although inverse association relationships can be defined to connect two descriptors reversely, the third predicate is used only for representing association relationships between descriptors. The last two demonstrate the logical connections of two context descriptors through disjoint relationships and equivalent relationships. These five predicates represent the types of relationships that have to be defined between context descriptors in the context ontology (the details of the required relationships have been discussed in the "Representation of Context Descriptors" section) and are listed in Table 2. The second column of Table 2 lists these predicates' definitions that are, if feasible, based on the basic predicate, i.e., *subClass(x,y)*. For example, the *superClass(x,y)* is defined as *subClass(y,x)* and the *equivalent(x,y)* is defined as *subClass(x,y) &subClass(y,x)*. On the other hand, as the *associate(x,y)* and *disjoint(x,y)* cannot be defined through the basic predicate, their definition cells are left blank in Table 2.

In addition, the following predicates are also used in the reasoning premises and rules to represent the three types of applicability values a context descriptor can possess:

6. *applicable(x)*: context descriptor x carries the applicability value of *applicable*
7. *notApplicable(x)*: context descriptor x carries the applicability value of *not applicable*
8. *possiblyApplicable(x)*: context descriptor x carries the applicability value of *possibly applicable*

A context descriptor carrying the applicability value of *applicable* means the situation described by the context descriptor applies to the target context. From the perspective of planning in advance, the target context is planned to appear in a project at a scheduled time in the future; from the perspective of construction, the target context is experienced and can be currently observed on sites; from the retrospective/postanalysis perspective, the target context appeared at a specific time in the past and was recorded. For example, if an engineer observes that a bulldozer currently is being used on a construction site, then a context descriptor *Bulldozer* in a context ontology can be flagged *applicable*. Similarly, a *not applicable* context descriptor means the situation described by the context descriptor does not apply to the target context; in other words, the situation will not occur in the future, does not take place currently, or did not appear in the past. For example, if an engineer is sure that no bulldozer will be used today, then the descriptor *Bulldozer* can be flagged *not applicable*. Lastly, a context descriptor carrying the applicability value of *possibly applicable* means for all an worker or engineer knows, the available information is insufficient to determine whether the situation described by the context descriptor applies to the target context or not. For example, if an engineer does not know whether or not a bulldozer currently is being used on a site, then the context descriptor *Bulldozer* should be flagged *possibly applicable*, representing the fact is unknown to the engineer.

Table 2. Predicates Representing the Relationships between Context Descriptors Leveraged by the Reasoning Rules

Predicate interpretation	Definition	Explanation
subClass(x,y)	—	descriptor x is a subclass of descriptor y
superClass(x,y)	*subclass(y,x)*	descriptor x is a superclass of descriptor y
associate(x,y)	—	descriptor x connects to descriptor y through association relationships
disjoint(x,y)	—	descriptor x is disjoint with descriptor y
equivalent(x,y)	*subClass(x,y)&subClass(y,x)*	descriptor x is equivalent to descriptor y
applicable(x)	—	descriptor x carries the applicability value of *applicable*
notApplicable(x)	$\sim$*applicable(x)*	descriptor x carries the applicability value of *not applicable*
possiblyApplicable(x)	$\Diamond$*applicable(x)*&$\Diamond\sim$*applicable(x)*	descriptor x carries the applicability value of *possibly applicable*

J. Comput. Civ. Eng. 2011.25:331-346.

These three predicates are also listed in Table 2. The *notApplicable(x)* is defined as $\sim applicable(x)$ where "$\sim$" is a negation operator for negating the predicate *applicable(x)*. The *possiblyApplicable(x)* is defined as $\Diamond applicable(x) \& \Diamond \sim applicable(x)$ where $\Diamond$ is an epistemic modal operator for representing the epistemic possibility of predicates. Hence, the definition of the *possiblyApplicable(x)* is interpreted as "the situation described by the context descriptor is possibly applicable to the target context, and it also is possibly not applicable." It is notable that the domain of discourse of the individual variables in the aforementioned predicates 1–8 (i.e., x and/or y) are the context descriptors represented in the context ontology.

Reasoning Premises

1. Each context descriptor has three options of applicability: *applicable*, *not applicable*, and *possibly applicable*, and each context descriptor must carry exactly one of these applicability options at any given time.

   The premise specifies the applicability values a context descriptor can carry: a context descriptor is either *applicable* or *not applicable* or *possibly applicable*. The premise is important because it lays a foundation for the reasoning mechanism by enabling the conceptualization of context descriptors' applicabilities.
2. Before the reasoning process starts, at least one context descriptor is assigned an applicability value of either *applicable* or *not applicable* to describe a project's target context.

   The premise requires that a project's target context is identified and described through at least one context descriptor before the reasoning process begins. The target context can be identified through human observation of the surroundings or automated reality capture technologies (Boukamp and Ergen 2008). The identified target context is described through context descriptors chosen from the context ontology, whereby the applicability value of these context descriptors is set as either *applicable* or *not applicable*. The context descriptors describing the identified target context are used as the initial input into the reasoning process. Therefore, this premise guarantees that initial input into the reasoning mechanism exists and is made available.

Reasoning Rules

1. All context descriptors are initially set to be *possibly applicable* before activating the reasoning process.

   Before the reasoning process is activated, all the context descriptors of the context ontology are initially set to be possibly applicable because there is no information to help judge whether the context descriptors are applicable or not at this moment.
2. If a context descriptor is selected to describe a target context, the descriptor is set to be either *applicable* or *not applicable* according to the input provided.

   Once a target context is known, users can select the context descriptors that best describe the target context from the context ontology. Then the selected descriptors are set to be *applicable* or *not applicable*, according to users' input. After that, the applicability of other context descriptors of the context ontology can be updated accordingly by using the following reasoning rules.
3. If a context descriptor is applicable, its supercontext descriptors must be *applicable*. If a context descriptor is *not applicable*, its subcontext descriptors must be *not applicable*.

   A context descriptor's superclass is the generalization of the context descriptor. If the context descriptor is *applicable*, any generalizing context descriptor must have the same applicability. Similarly, a context descriptor's subclasses are the specialization of the context descriptor. If the context descriptor is *not applicable*, there is no chance for a further specialized context descriptor to be *applicable*; that is, the specialized context descriptors must be *not applicable*. In other words, the applicability value *applicable* of a context descriptor should be propagated upward to context descriptors in higher hierarchy levels of a context ontology whereas the applicability value *not applicable* of a context descriptor should be propagated downward to contexts in lower level of a context ontology. For example, if a context descriptor *Retaining Wall* is *applicable* for a project, its superconcept *Wall* must be *applicable*; if the concept *Wall* is *not applicable*, its subconcept *Retaining Wall* must be *not applicable*. The formulas for the predicate of this reasoning rule are as follows:

$$\forall x \forall y [applicable(x) \wedge subClass(x,y) \rightarrow applicable(y)]$$

$$\forall x \forall y [notApplicable(x) \wedge superClass(x,y) \rightarrow notApplicable(y)]$$

4. If a context descriptor is *applicable*, any associated context descriptor must be *applicable*.

   This rule only applies when a context descriptor is found to be *applicable*. The association relationships are used to connect semantically related context descriptors and, therefore, indicate that when the connecting context descriptor is *applicable*, the connected context descriptor should be *applicable* as well. For example, a context descriptor *Work at Elevation* connects with a context descriptor *Fall* through an association relationship *hasHazard*. Once the descriptor *Work at Elevation* is found *applicable*, the descriptor *Fall* will accordingly become *applicable*. The formula for the predicate of this reasoning rule is as follows:

$$\forall x \forall y [applicable(x) \wedge associate(x,y) \rightarrow applicable(y)]$$

   The applicability value propagation specified by this rule is designed to be unidirectional; that is, the applicability value is propagated through association relationships only from connecting descriptors to connected ones. The purpose is to provide better control over reasoning processes and propagations of applicability values. Users can define another association relationship that is inverse to an association relationship to allow propagating applicability value in the opposite direction by using this reasoning rule.
5. Two context descriptors connected through an equivalent relationship must carry the same applicability value.

   Context descriptors that are connected through equivalent relationships have the same contextual meaning, and therefore, must share the same applicability value, no matter whether the value is *applicable*, *not applicable*, or *possibly applicable*. For example, in the context of construction safety, if the applicability of a context descriptor *Slip* is *applicable*, another descriptor *Trip* will also carry the applicable value as these two descriptors share the same meaning: someone accidently slides or falls and loses his/her balance. The formulas for the predicate of this reasoning rule are as follows:

$$\forall x \forall y [applicable(x) \wedge equivalent(x,y) \rightarrow applicable(y)]$$

J. Comput. Civ. Eng. 2011.25:331-346.

$$\forall x \forall y[notApplicable(x) \wedge equivalent(x,y) \rightarrow notApplicable(y)]$$

$$\forall x \forall y[possiblyApplicable(x) \wedge equivalent(x,y) \rightarrow possiblyApplicable(y)]$$

6. If a context descriptor is *applicable*, any disjoint context descriptor must be *not applicable*.

   This rule only applies when a context descriptor is found *applicable*. Context descriptors that are connected through disjoint relationships meaningfully exclude one another. Therefore, when one context descriptor is *applicable*, it will exclude any other context descriptor with which it connects through a disjoint relationship and these connected descriptors therefore have to become *not applicable*. For example, if a context descriptor *Precast* is *applicable*, another descriptor *Cast-In-Place* that is declared to be disjoint with *Precast* must be *not applicable*. The formula for the predicate of this reasoning rule is as follows:

$$\forall x \forall y[applicable(x) \wedge disjoint(x,y) \rightarrow notApplicable(y)]$$

7. If a combined context descriptor in the context ontology is *applicable*, its constituent context descriptors have to be *applicable*.

   This rule is a variation of the fourth rule. A combined context descriptor uses association relationships to represent the relations between it and the constituent context descriptors. Therefore, if the combined context descriptor is *applicable*, by applying the fourth rule, the constituent context descriptors must be *applicable*. However, if the combined context descriptor is *not applicable*, this applicability value will not be propagated to the constituent contexts. For instance, a combined context descriptor *Frame Column* can have association relationships to connect it to a context descriptor *Frame*, specifying its related action information, and a context descriptor *Column*, specifying its related component information. Once this combined context descriptor is found *applicable*, the context descriptors *Frame* and *Column* must be *applicable*, too.

Following these reasoning rules, engineers can fully evaluate the applicability of all the context descriptors of an ontology. As for how the construction information should be indexed through the context descriptors, engineers have to consider the practical situations that apply to the construction information and find the descriptors that best describe the situations to index the information. In the next section, the writers present two case studies used to validate the proposed framework.

Case Studies

In this research, two case studies were performed to validate the proposed framework's ability to explicitly represent context descriptors in ontologies and to soundly reason about these context descriptors for searching for relevant construction information. The first case study was conducted by the first writer of the paper for retrieving and classifying Job Hazard Analysis information. The second case study was conducted by the third writer of the paper for classifying, archiving and retrieving on-site construction problem documents from a database. An object-oriented, Java-based prototype system was developed in each case study to implement the validation. Although the prototype systems were developed for managing different construction information, they shared the same reasoning mechanism of the proposed framework that is discussed in the previous section.

Case Study 1: Retrieval and Classification of Safety Rules from Job Hazard Analysis Information

Job hazard analysis (JHA) is a technique that identifies potential hazards for each step of a job and proposes safety rules to prevent these hazards. The Occupational Safety and Health Administration (OSHA) recommends conducting JHAs in projects to prevent hazards in workplaces and to reduce worker injuries and illnesses (U.S. Dept. of Labor 2002). Retrieving relevant JHA information from a company's archive is important and beneficial to safety engineers. First, engineers conducting JHA rely on brainstorming sessions to identify steps within different construction jobs and to identify associated hazards. Retrieving relevant JHA information allows them to quickly revisit the safety knowledge in the form of safety rules in the documents of previous projects during the brainstorming sessions and to modify and/or reuse the knowledge when engineers prepare new JHA information. Second, it enables them to quickly identify safety rules applicable to current project contexts when these rules are requested on sites. Hence, the case study here aims to improve access to a company's JHA knowledge via the proposed framework and the developed prototype system. Eight concrete job-related JHA documents were acquired for a school building project from a private construction company. There were eight jobs, 32 job steps, 58 potential hazards, and a total of 136 rules in the eight JHA documents. Fig. 3 shows a snippet of one of the acquired JHA documents.

JHA documents include two kinds of information: JHA context and JHA safety rules. The former consists of context descriptors, which describe jobs, job steps, and the identified potential hazards and site conditions; the latter are the safety rules imposed to address the identified hazards. In the example shown in Fig. 3, the JHA context descriptors include *Frame Column* (job context descriptor), *Stand Forms into Place*, *Set Pins* (job step context descriptors), and *Sprain/Strain of Back*, *Pinched Fingers*, *Sharp Edges*, and *Fall* (potential hazard context descriptors). The JHA safety rules are all the rules listed in the "Recommended Safety Rules" column and each safety rule has its corresponding potential hazard, job step, and job context descriptors in which the rule becomes applicable.

Representation of JHA Context Descriptors and Safety Rules

Job, job step, and potential hazard descriptors describing the context of JHA documents should be represented in context ontologies. Three grouping context descriptors are defined: *Job*, *Job Step*, and *Potential Hazard*, to group all the context descriptors, such as *Frame Column* being a group member of grouping context *Job*. When further represented in class hierarchies, grouping contexts *Job*, *Job Step*, and *Potential Hazard* become the primary classes and their group members become the respective subclasses. In addition, an association relationship *comprise* and its inverse relationship *isPartOf* are defined to semantically link the primary classes *Job* and *Job Step*. Similarly, another association relationship *hasHazard* and its inverse relationship *isRelatedTo* are defined to enable the semantic connection of the primary classes *Job Step* and *Potential Hazard*. In this case study, equivalent relationships were defined between some of the potential hazard context descriptors. For instance, the potential hazard context descriptor *Slip* was declared to be equivalent to the descriptor *Trip* because they both mean that someone accidently slides or falls and loses his/her balance in the context of construction safety.

In the case study, the writers also represented the JHA safety rules together with their related context descriptors in Extensible Markup Language (XML) format to facilitate the retrieval of the contextual and safety rule information. Each safety rule in the representation should be indexed conjunctionally by its related context

J. Comput. Civ. Eng. 2011.25:331-346.

Job Hazard Analysis

Contractor: ABC **Project:** XYZ project
Job: Frame Column **Job Plan Dated:** April 5, 2007

Job Steps	Potential Hazards	Recommended Safety Rules
Stand forms into place	Sprain/Strain of back	• Use proper lifting technique. • Get assistance; Work with a partner.
	Pinched fingers	• Wear slip resistant gloves. • Set form on ground away from adjacent form then grab form in a place where your fingers will not get pinched.
	Sharp edges	• Review rebars' conditions for sharp edges or tie wire hazards.
	Fall	• Use ladder or scaffold; do not use top 2 rungs of ladder. • Ensure area around ladder/ scaffold is clear of debris and flat.
Set pins	Fall	• Use a portable ladder in the proper manner. • Get a partner to hold the form when needed. • Use scaffolding where possible; Scaffold must be erected under supervision of a competent person; all guardrails must be installed and pins used; No substitute materials! • If climbing form, must use retractable lanyard anchored to top of form when feet are higher than 6' off working surface.

Fig. 3. JHA document example

descriptors as its applicability conditions. For example, the safety rule "Use a portable ladder in the proper manner" shown in Fig. 3 is indexed conjunctionally by the following contexts: *Frame column* (job context descriptor), *Set pins* (job step context descriptor), and *Fall* (potential hazard context descriptor). That means safety rules do not have to be tied to job and/or job step context descriptors only, but can also be tied to potential hazard context descriptors. Hence, to evaluate a safety rule's applicability is to evaluate its applicability conditions. When one or more of the context descriptors in an applicability condition are found applicable, the applicability condition is determined to be satisfied and therefore the safety rule carrying that condition should apply as well. For example, if the job context descriptor *Frame column* is found applicable, the safety rule "Use a portable ladder in the proper manner" must be applicable because *Frame column* is one of the descriptors describing the rule's applicability condition.

Reasoning Process

Fig. 4 shows the screenshot of the running prototype system for the case study. The context ontology for the case study is shown on the left of the window whereas the safety rules together with their related context descriptors are shown on the right. Once a context descriptor of the context ontology is specified as applicable (with a check mark in front of the descriptor), such as the job context descriptor *Frame Column* circled in Fig. 4, the reasoning mechanism helps evaluate other context descriptors and also evaluates the safety rules' applicabilities. Context descriptors found applicable are shown with check marks; possibly applicable context descriptors are preceded with a question mark; and not applicable descriptors—which are not shown in the Fig. 4—are indicated with cross mark. For instance, when the job context descriptor *Frame Column* is selected to be applicable, the job step context descriptor *Set pins* are evaluated to be applicable according to the reasoning rule no. 4 because it is connected to the descriptor *Frame Column* through an association relationship. In addition, the potential hazard *Fall* (not shown in Fig. 4) becomes applicable when applying reasoning rule no. 4 because of an association with the descriptor *Set pins*. These reasoning results help conclude that the five safety rules related to these context descriptors are applicable (shown in a rectangle in Fig. 4). The context descriptors and safety rules are color coded within the system prototype, with each of the same applicability values (e. g., *applicable*, *possibly applicable*, or *not applicable*) represented by a different color. After the reasoning process evaluated the applicability of the context descriptors, the safety rules can be classified according to their applicability and then be output as a file in which applicable, possibly applicable and not applicable safety rules can be printed in separate sections for engineers' use.

Summary

In this case study, the selection of applicable contexts was on the basis of manual input, i.e., applicability values were assigned to context descriptors in the context ontology. The writers observed that the applicability values were correctly propagated among the context ontology according to the proposed reasoning rules. In addition, the writers also found that the safety rules related to the specified or inferred context descriptors were successfully retrieved and classified on the basis of their applicability values. For example, Table 3 shows the reasoning results of specifying each job context descriptor as applicable, which include the number of inferred applicable job step and potential hazard context descriptors and safety rules. The job step context descriptors were obtained from inference through association relationships defined between job and job step context descriptors; most of the potential hazard context descriptors were obtained from inference through association relationships although some were identified through equivalent relationships.

J. Comput. Civ. Eng. 2011.25:331-346.

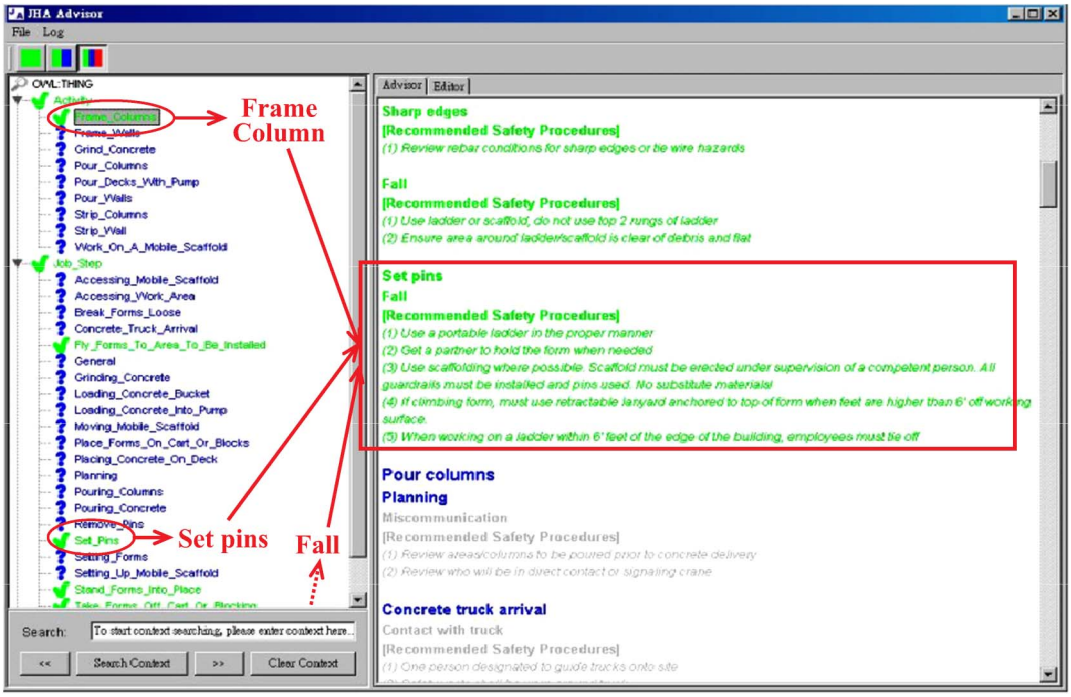

Fig. 4. Applying the framework to the JHA case study in the system prototype

There are two major advantages brought by the reasoning mechanism in this case study. First, implicit context descriptors can be inferred through associations and by means of the reasoning rules; then, these context descriptors can help identify related safety rules that may otherwise have been ignored or unattended. Second, engineers are able to more quickly search out applicable safety rules by selecting context descriptors from the context ontology (as shown on the left in Fig. 4) and specifying their applicability values rather than by inputting keywords to search through the whole collection of JHA documents. Once engineers are done with selecting context descriptors, the reasoning mechanism follows to carry on the remaining retrieval tasks. Thereby, search time can be reduced for the engineers and they can easily reconduct the process of retrieving safety rules and quickly respond when jobs of a project change and require revisions of JHA documents. The results of this case study were shown to construction practitioners, and they found the proposed framework to be helpful and useful for efficiently and effectively identifying applicable JHA safety rules and preparing JHA documents.

Case Study 2: Retrieval of On-Site Construction Problem Documents

Construction problem documents record construction problems encountered in previous projects along with proposed solutions to tackle these problems. The experience in solving construction problems recorded in these documents provides valuable information to engineers about how to address similar problems in the future. This case study illustrated the application of the proposed framework to help retrieve construction problem documents according to identified and inferred contexts of construction problems.

A typical construction problem document consists of the following information: project name, date, project basic information, personnel involved, problem description, and corresponding solutions. To apply the proposed framework to retrieve such documents, the situations to which documents apply should be identified and represented in the form of context descriptors, i.e., by using context descriptors to index the applicability conditions of the documents. For example, if a construction problem document is related to the problem of a concrete wall crack and prescribes the solution of using epoxy to seal the crack, the context descriptors describing the applicability conditions may include *Concrete*, *Wall*, and *Crack*.

Table 3. Reasoning Results of Specifying Job Context Descriptors as Applicable in Case Study 1

Name of job	Number of inferred applicable context descriptor: Job step	Number of inferred applicable context descriptor: Potential hazard	Number of inferred applicable safety rule
Frame column	4	9	20
Pour column	4	13	19
Strip column	4	5	11
Frame wall	2	9	13
Pour wall	4	12	19
Pour deck with pump	4	16	25
Grind concrete	2	7	11
Working on a mobile scaffold	4	7	18

Scenario Definition

In a target scenario of this case study shrinkage cracks were found in a cast-in-place foundation wall. These types of cracks can be easily identified in cast-in-place concrete products and distinguished from other types of cracks that take place in the later life of a foundation wall. A construction engineer aims to retrieve information about similar problem reports that were stored in the corporate database to identify possible and appropriate actions to address the problem of shrinkage cracks. In the case study, a reinforced concrete wall under construction in Newmark Civil Engineering Laboratory at the University of Illinois Urbana–Champaign was chosen for performing the study.

J. Comput. Civ. Eng. 2011.25:331-346.

Representation of Construction Problem Document Contexts into Ontologies

In the case study, four types of ontologies were developed: product, process, resource, and problem-sources ontologies, to respectively represent context descriptors of product types, construction processes, applicable resources, and problem sources. For instance, context descriptors, such as *Management Product*, *Construction Complex*, and *Basic Product*, were represented as different classes on the same level and context descriptors *Building Product*, *Civil Product*, and *Industrial Product* were represented as subclasses of the class *Construction Complex* in the product ontology. In addition, association and logical relationships were also defined for linking context descriptors defined in the ontologies.

Reasoning Process

Fig. 5 shows the screenshot of running the system prototype for the second case study. The context ontologies for the case study are shown on the left side of the window whereas the information of construction problem documents with their related contexts is shown on the right. This case study first utilized RFID as a technique to acquire contextual information. RFID tags were attached to different components on site. These tags carried information about context descriptors from the ontologies used to describe the component to which the tags were attached. Twenty-two context descriptors related to the scenario were identified through the RFID technique and suggested to the construction engineer. Of these, the construction engineer selected three context descriptors, *FoundationWall*, *Footing*, and *Formwork*, for retrieving relevant documents from the corporate document database. The three context descriptors were specified as applicable in the two respective context ontologies: product ontology and resource ontology as user-specified contexts. The proposed reasoning mechanism then inferred seven other context descriptors that are applicable on the basis of the relationships defined in the context ontologies.

In addition to the context descriptors suggested through RFID, the construction engineer manually selected three other context descriptors: *Concrete*, *Cast-In-Place*, and *Temperature* to be applicable. This led to 13 additional context descriptors being inferred to be applicable by reasoning about the 3 selected context descriptors with the proposed reasoning mechanism. For example, by selecting the context descriptor *Concrete* as applicable, the reasoning mechanism automatically inferred contexts such as *Liner*, *Plaster*, *Scaffold*, and *Scaffolding* as applicable according to the association relationships defined in the resource ontology. Table 4 shows the context descriptors selected and inferred through the reasoning mechanism from both context selection approaches. Table 4 also shows the ontological relationships through which the applicabilities of additional context descriptors were obtained.

After the engineer reviewed the selected and inferred context descriptors to ensure that they describe the targeted context, relevant construction problem documents, which were indexed with the selected or inferred context descriptors, were retrieved from the corporate database. After reviewing the retrieved records, the construction engineer determined to apply one of the recorded problems' actions to the current situation. The action recommended the usage of Polyurethane foam sealant to seal the shrinkage cracks. Finally, this action has been recorded to be effective when it was

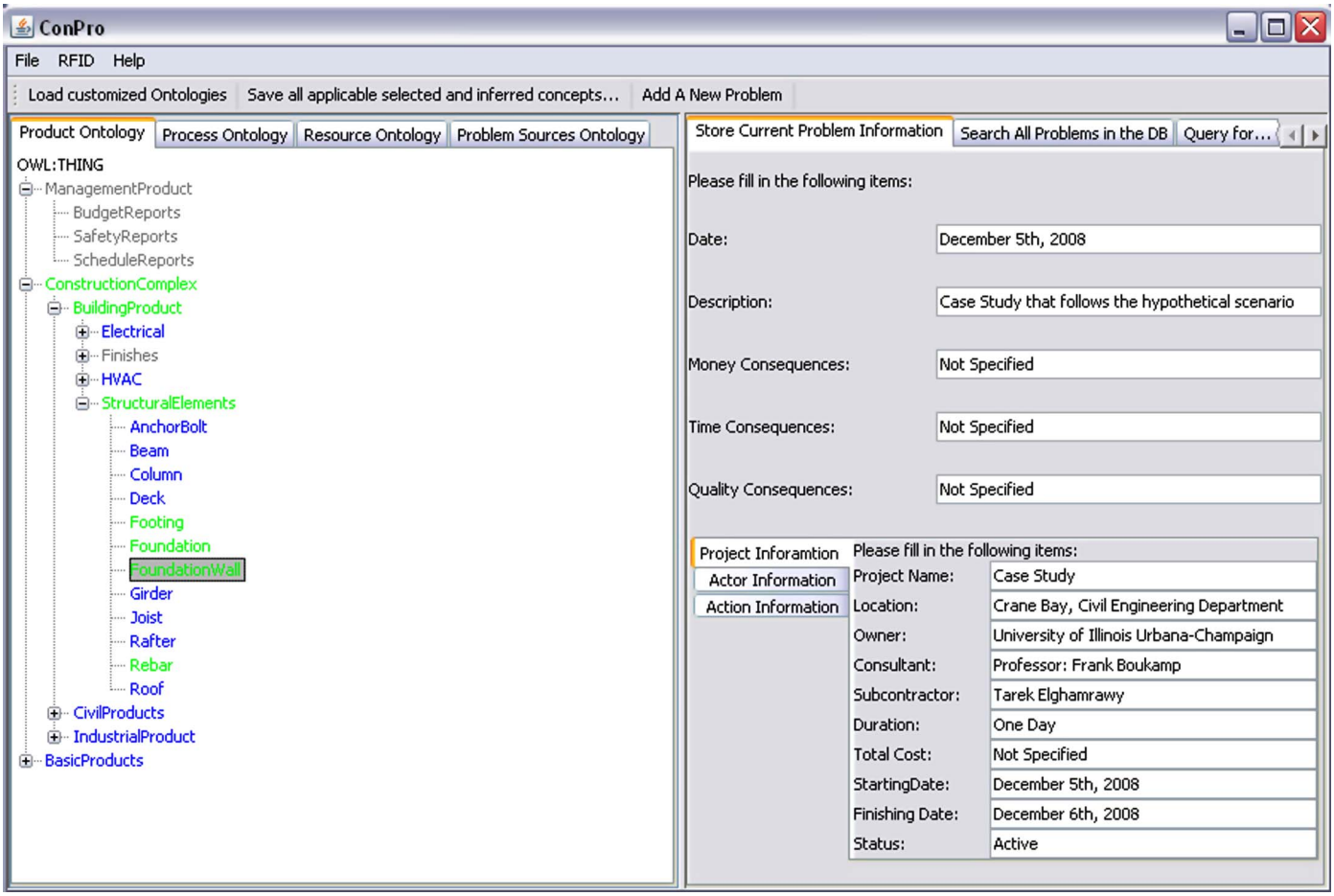

Fig. 5. Applying the framework to the second case study in the system prototype

J. Comput. Civ. Eng. 2011.25:331-346.

Table 4. List of the User-Specified and Inferred Applicable Context Descriptors

Context selection approach	Context ontology	User selected context descriptor	Inferred context descriptor	Type of relationship
Automatically Scanned using RFID technique	Product ontology	*FoundationWall*	*StructuralElements*	Subsumption relationship
			BuildingProduct	
			ConstructionComplex	
			Rebar	Association relationship
		Footing	*Foundation*	Equality relationship
	Resource ontology	*Formwork*	*SiteResources*	Subsumption relationship
			Forms	Equality relationship
Manually user selected	Resource ontology	*Concrete*	*Materials*	Subsumption relationship
			Liner	Association relationship
			Plaster	
			Scaffold	
			Scaffolding	
	Process ontology	*Cast-In-Place*	*ConcreteProcess*	Association relationship
			FieldConstruction	
			Construction	
			EngineeringProcesses	
			Placing	Equality relationship
			Pouring	
			Curing	Association relationship
	Problem sources ontology	*Temperature*	*Environment*	Subsumption relationship

applied to a similar problem that took place in one of the company previous projects.

Summary

In this case study, the writers illustrated the advantage of the reasoning mechanism. Implicit context descriptors can be inferred through associations defined in the ontologies and the reasoning rules, as in the first case study; then, these context descriptors can be used to index construction problem documents from a company's database so they can be retrieved again at a later point in time by using this context-based search mechanism. In this case study, the writers found that the use of ontologies on mobile computers on construction sites becomes more and more difficult, the deeper the class hierarchies of the represented context descriptors are. Having to navigate through the class hierarchies can be cumbersome when the ontologies increase in size and depth. This case study showed the possibility of integrating reality capture technology with the proposed reasoning mechanism to help in the acquisition of initial contextual information and to support the reasoning process. The RFID technology on-site proposed initial context descriptors to the user. Other descriptors were inferred on the basis of the initial selection. The user then was able to review and adjust the list of descriptors to be used for the document search, which reduced the amount of time spent to search for relevant context descriptors for the user.

Discussion

The proposed approach leverages the rich semantics of ontologies. Developing a context ontology for the construction information of an application domain requires identifying the context descriptors describing the domain contexts; structuring the descriptors in classifications; and assigning proper association and logical relationships between the descriptors. These requirements indicate some concerns that need to be highlighted specifically.

- Context descriptors are cornerstones of a context ontology; hence, properly identifying them is important to the proposed approach. The identification of context descriptors can be accomplished by carefully analysing the construction information from which the descriptors are extracted. If ontology modelers are not experts of the application domain, having the experts involved in the identification process is beneficial. In addition, some semiautomated approaches, such as the one proposed in Wang and Boukamp (2008), are also eligible means to identify context descriptors.
- As previously mentioned, classifying context descriptors in hierarchies can facilitate both the representation of and reasoning about the descriptors. In other words, failure to provide proper classifications not only makes the ontology less manageable, it also limits the effectiveness of the reasoning mechanism, e.g., by preventing retrieval of information that is relevant to a generalization of a given context, but cannot be indexed properly, because the generalized context has not been introduced in the context hierarchy (e.g., information that is applicable to *material* in general may not be found if the search context is *concrete* and *concrete* has not been classified as a type of *material*).
- An association relationship helps propagate the applicabilities from one descriptor to another. Not defining existing association relationships hinders the applicability propagation between descriptors. However, defining incorrect association relationships will lead to incorrect applicability propagation among the context descriptors. Therefore, ontology modelers should carefully examine the practical situations described in the construction information for which the ontology is to be developed to take into account all suitable and necessary association relationships between descriptors.
- Disjoint relationships help filter out irrelevant context descriptors; equivalent relationships help find out context descriptors that have the same meanings and applicabilities as the input descriptors. Although failure to specify these logical relationships between descriptors does not harm the information retrieval, it

J. Comput. Civ. Eng. 2011.25:331-346.

hinders leveraging the reasoning mechanism's full capabilities. For example, the lack of a logical equivalent relationship between two context descriptors *Cast-In-Place* and *Cast-In-Situ* would prevent applicability propagation when one of the contexts becomes applicable or not applicable. The other context therefore would remain possibly applicable, which means that the user would still retrieve the associated information, but would have to review personally whether the information applies to the target context or not. Therefore, it is necessary to carefully examine context descriptors to decide what descriptors upon which the logical relationships should deploy.

Conclusions

In this research, the writers propose a new approach to construction information management, which enables information management to be based on not only explicit but also implicit contextual information that relates to and can be deduced from the explicit contextual information. The new approach presents a framework that is based on the notion of context descriptors that can be used to describe project situations and act as indices of construction information. The framework is composed of two parts: the representation and the reasoning of context descriptors. The former discusses a systematic approach for representing context descriptors in OWL-based ontologies; the latter articulates a reasoning mechanism, which presents the rules for reasoning about the context descriptors represented in ontologies to obtain the descriptors' applicabilities. The framework also suggests using the logical concatenations of context descriptors represented in ontologies to index the applicability conditions of construction information. Thereby, when the logical concatenation of context descriptors of a piece of information is evaluated and found applicable, the applicability condition indexed through the descriptors is satisfied and the information is determined to be applicable.

The writers conducted two case studies, one for retrieving JHA information with safety rules and the other for retrieving construction problem documents. Both case studies' results show that the proposed framework provides an organized process to represent context descriptors into ontologies and also successfully enables the reasoning about the descriptors' applicabilities. The case studies also highlight the major merit of the proposed framework: implicit context descriptors can be inferred through defined associations and the reasoning rules. Identifying implicit context descriptors can help retrieve ignored and unattended construction information that has been indexed by using the implicit context descriptors. Moreover, context ontologies are usually illustrated in a hierarchical structure (as shown in Figs. 4 and 5), which makes it easier to navigate context descriptors, specify applicability and identify context descriptors' applicabilities after reasoning (e.g., using check, cross, and question mark and different color coding in the system prototype) than in traditional keyword-based searches.

The two case studies also present two different ways to determine the applicabilities of the descriptors that act as initial input for target context: manual selection in the first and an automated reality capture technology in the second. Although manually determining and selecting context descriptors is straightforward, it becomes less efficient when the number of the descriptors in context ontologies increases. Therefore, having other means to help identify context descriptors from the context ontologies is suggested. For example, reality capture technologies integrated with the proposed framework were successfully applied as the major technique of target context identification on a construction site to determine the initial context descriptors and their applicabilities. In an office setting, the manual selection process was supplemented with a keyword search mechanism that helped searching for context descriptors within the ontologies. Manual selection can act as a supplement means, allowing engineers to flexibly input necessary applicability information of descriptors and to fine tune the reasoning process.

Whereas the proposed framework has been successfully applied to two types of construction information in the case studies, application to other types of construction documents should be conducted in future research to further test the proposed framework. A limitation of the framework is that it currently can only handle contexts that target a single component, resource, or activity. When deploying the proposed applicability propagation, context descriptors that aim to describe multiple components, resources, or activities at once, may lead to multiple context descriptors being found to semantically contradict one another. To address this issue, the implemented framework currently asks users to handle the context contradictions. That is, users have to determine whether to accept or reject the applicabilities of the context descriptors from contradicting contexts. For instance, given a scenario that a precast column is set up on a cast-in-place foundation, the context *Precast Concrete Column* is contradictory to the context *Cast-In-Place Foundation* because of the defined disjointedness between the context descriptors *Precast* and *Cast-In-Place*. When the first context has been processed by the reasoning mechanism, context contradiction occurs once the second one is input into the reasoning process. Such contradiction of context descriptors' applicabilities will detrimentally affect the reasoning efficiency when the proposed framework is deployed in scaling cases in which contexts with huge number of context descriptors are involved. To remove this research limitation and increase the reasoning efficiency, further research on how to address and solve context contradictions in an automated manner in the proposed framework is necessary. A potential solution, for example, is to consider the propagation of applicabilities under conditions, i.e., setting applicability propagation rules for different conditions when context contradictions take place. Finally, how to facilitate the maintenance of context ontologies in the developed system prototypes should be studied. Although ontological modeling is a powerful technique for representing contextual information and initial ontologies can be developed through cooperation of domain experts and ontology modelers, it is difficult for civil engineers to modify the developed ontologies afterward because they usually do not have the relevant background knowledge. A straightforward context ontology editing tool for civil engineers should be available in the prototype systems to further benefit the application of the proposed framework.

Acknowledgments

The writers would like to thank Dr. Allen Renear of Graduate School of Library and Information Science of University of Illinois at Urbana–Champaign, and Patrick McGowan, Daniel Ruane, and James Smith from W.E. O'Neil Construction of Chicago for sharing their valuable knowledge and supporting this research.

References

Antoniou, G., and van Harmelen, F. (2004). *A semantic web primer*, MIT Press, Cambridge, MA.

Aziz, Z., Anumba, C., and Law, K. (2006). "Using context-awareness and web-services to enhance construction collaboration." *Joint Int. Conf. on Computing and Decision Making in Civil and Building Engineering*, Int. Council for Research and Innovation in Building and Construction (CIB), Rotterdam, Netherlands, 3010–3019.

J. Comput. Civ. Eng. 2011.25:331-346.

Aziz, Z., Anumba, C. J., Ruikar, D., Carrillo, P. M., and Bouchlaghem, N. M. (2005). "Context aware information delivery for on-site construction operations." *22nd CIB W78 Conf. on Information Technology in Construction*, Int. Council for Research and Innovation in Building and Construction (CIB), Rotterdam, Netherlands, 321–332.

Bechhofer, S., et al. (2004). "OWL web ontology language reference." ⟨http://www.w3.org/TR/owl-ref/⟩ (Sep. 1, 2009).

Boukamp, F. (2006). "Modeling of and reasoning about construction specifications to support automated defect detection." Ph.D. thesis, Carnegie Mellon Univ., Pittsburgh.

Boukamp, F., and Ergen, E. (2008). "A proposed system architecture for context identification support on construction sites." *5th Int. Conf. on Innovation in Architecture*, Centre for Innovative and Collaborative Construction Engineering (CICE), Loughborough, U.K.

Breitman, K. K., Casanova, M. A., and Truszkowski, W. (2006). *Semantic web: Concepts, technologies and applications*, 1st Ed., Springer, London.

buildingSMART. (2010). "Industry foundation classes (IFC)." ⟨http://www.iai-tech.org/⟩ (Apr. 29, 2010).

Caldas, C. H., Soibelman, L., and Gasser, L. (2005). "A methodology for the integration of project documents in model-based information systems." *J. Comput. Civ. Eng.*, 19(1), 25–33.

Caldas, C. H., Soibelman, L., and Han, J. (2002). "Automated classification of construction project documents." *J. Comput. Civ. Eng.*, 16(4), 234–243.

Chen, H., Finin, T., and Joshi, A. (2003). "Using OWL in a pervasive computing broker." *2nd Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems/Workshop on Ontologies in Agent Systems*, Int. Foundation for Autonomous Agents and Multiagent Systems (IFAAMS), Richland, SC, 9–16.

Cimiano, P., and Völker, J. (2005). "Text2Onto: A framework for ontology learning and data-driven change discovery." *Proc. of the 10th Int. Conf. on Applications of Natural Language to Information Systems*, Springer, Heidelberg, Germany.

Darlington, M. J., and Culley, S. J. (2008). "Investigating ontology development for engineering design support." *Adv. Eng. Inform.*, 22(1), 112–134.

Dey, A. K., and Abowd, G. D. (2000). "Towards a better understanding of context and context-awareness." *Proc. of the CHI 2000 Workshop on "The What, Who, Where, and How of Context-Awareness"*, Georgia Institute of Technology, Atlanta.

Dolenc, M., Katranuschkov, P., Gehre, A., Kurowski, K., and Turk, Ž. (2007). "The InteliGrid platform for virtual organisations interoperability." *ITCon*, 12, 459–477.

El-Diraby, T. E., Lima, C., and Feis, B. (2005). "Domain taxonomy for construction concepts: Toward a formal ontology for construction knowledge." *J. Comput. Civ. Eng.*, 19(4), 394–406.

El-Diraby, T. E., and Zhang, J. (2006). "A semantic framework to support corporate memory management in building construction." *Autom. Constr.*, 15(4), 504–521.

Elghamrawy, T., and Boukamp, F. (2008). "A vision for a framework to support management of and learning from construction problems." Int. Council for Research and Innovation in Building and Construction (CIB), Rotterdam, Netherlands.

Feruzan, A. (2007). "Context modeling and reasoning using ontologies." ⟨http://www.aywa.de/cmaruo/cmaruo.pdf⟩ Univ. of Technology Berlin (Apr. 24, 2008).

Furnas, G. W., et al. (2006). "Why do tagging systems work?." *Conf. on Human Factors in Computing Systems*, Association for Computing Machinery (ACM) Special Interest Group on Computer Human Interaction (SIGCHI), New York, 36–39.

Gruber, T. R. (1993). "A translation approach to portable ontology specifications." *Knowl. Acquis.*, 5(2), 199–220.

Haimes, B. (1994). "Document interface." *Interactions*, 1(4), 15–18.

Kartam, N. A. (1996). "Making effective use of construction lessons learned in project life cycle." *J. Constr. Eng. Manage.*, 122(1), 14–21.

Kiliccote, H. (1994). "The context-oriented model: A hybrid approach to modeling and processing design standards." M.S. thesis, Carnegie Mellon Univ., Pittsburgh.

Kim, E., and Choi, J. (2006). "An ontology-based context model in a smart home." *Proc. of the 2006 Int. Conf. on Computational Science and Its Applications*, Springer, Heidelberg, Germany, 11–20.

Korpipää, P., Häkkilä, J., Kela, J., Ronkainen, S., and Känsälä, I. (2004). "Utilising context ontology in mobile device application personalisation." *Proc. of the Third Int. Conf. on Mobile and Ubiquitous Multimedia*, Association for Computing Machinery (ACM), New York, 133–140.

Lai, Y.-C. (2006). "IT-CODE: IT in collaborative design." Ph.D. thesis, Aalborg Univ., Aalborg, Denmark.

Lima, C., El-Diraby, T. E., and Stephens, J. (2005). "Ontology-based optimisation of knowledge management in e-construction." *ITcon*, 10, 305–327.

TerMine [Computer software]. National Centre for Text Mining, Manchester, U.K.

OCCS. (2010). "Omniclass: A strategy for classifying the built environment." ⟨http://www.omniclass.org/⟩ (Apr. 29, 2010).

Rezgui, Y. (2006). "Ontology-centered knowledge management using information retrieval techniques." *J. Comput. Civ. Eng.*, 20(4), 261–270.

Russell, A. D. (1993). "Computerized daily site reporting." *J. Constr. Eng. Manage.*, 119(2), 385–402.

Scherer, R. J., and Reul, S. (2000). "Retrieval of project knowledge from heterogeneous AEC documents." *Proc. of the Eighth Int. Conf. on Computing in Civil and Building Engineering,*, ASCE, Reston VA, 812–819.

Soibelman, L., and Caldas, C. H. (2006). "A combined text mining method to improve document management in construction projects." *Joint Int. Conf. on Computing and Decision Making in Civil and Building Engineering (CIB)*, Rotterdam, Netherlands, 2912–2918.

Soibelman, L., Wu, J., Caldas, C., Brilakis, I., and Lin, K.-Y. (2008). "Management and analysis of unstructured construction data types." *Adv. Eng. Inform.*, 22(1), 15–27.

Sørensen, K. B., Christiansson, P., and Svidt, K. (2010). "Ontologies to support RFID-based link between virtual models and construction components." *Comput. Aided Civ. Infrastruct. Eng.*, 25(4), 285–302.

Souza, D., Salgado, A. C., and Tedesco, P. (2006). "Towards a context ontology for geospatial data integration." *Proc. of On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, Springer, Washington, DC, 1576–1585.

U.S. Dept. of Labor. (2002). *Job hazard analysis: OSHA publication 3071*, Occupational Safety and Health Administration (OSHA), Washington, DC.

W3C. (2004). "OWL: Web Ontology Language." ⟨http://www.w3.org/2004/OWL/⟩ (Dec. 11, 2008).

Wang, H.-H., and Boukamp, F. (2007). "Leveraging project models for automated identification of construction safety requirements." *Proc. of the 2007 ASCE Int. Workshop on Computing in Civil Engineering*, ASCE, Reston, VA, 240–247.

Wang, H.-H., and Boukamp, F. (2008). "A context ontology development process for construction safety." *Joint CIB Conf.: W102 Information and Knowledge Management in Building and W096 Architectural Management*, Int. Council for Research and Innovation in Building and Construction (CIB), Rotterdam, Netherlands, 297–308.

Wang, H.-H., and Boukamp, F. (2009). "Ontology-based job hazard analysis support." *Proc. of the 2009 ASCE Int. Workshop on Computing in Civil Engineering*, ASCE, Reston, VA, 676–685.

Wang, X. H., Zhang, D. Q., and Gu, T. (2004). "Ontology based context modeling and reasoning using OWL." *Second IEEE Annual Conf. on Pervasive Computing and Communications Workshops*, IEEE Computer Society, Washington, DC.

Zhu, Y., Mao, W., and Ahmad, I. (2007). "Capturing implicit structures in unstructured content of construction documents." *J. Comput. Civ. Eng.*, 21(3), 220–227.