Toward a Formal Common Information Model Ontology

Stephen Quirolgico[1], Pedro Assis[2], Andrea Westerinen[3], Michael Baskey[4], and Ellen Stokes[5]

[1] National Institute of Standards and Technology, Gaithersburg MD 20899, USA
[2] Instituto Politécnico do Porto, Dept. of Electrical Engineering, 4200-072 Porto, Portugal
[3] Cisco Systems, San Jose CA 95134, USA
[4] IBM, Poughkeepsie NY 12601, USA
[5] IBM, Austin TX 78758, USA

Abstract. Self-managing systems will be highly dependent upon information acquired from disparate applications, devices, components and subsystems. To be effectively managed, such information will need to conform to a common model. One standard that provides a common model for describing disparate computer and network information is the Common Information Model (CIM). Although CIM defines the models necessary for inferring properties about distributed systems, its specification as a semi-formal ontology limits its ability to support important requirements of a self-managing distributed system including knowledge interoperability and aggregation, as well as reasoning. To support these requirements, there is a need to model, represent and share CIM as a formal ontology. In this paper, we propose a framework for constructing a CIM ontology based upon previous research that identified mappings from Unified Modeling Language (UML) constructs to ontology language constructs. We extend and apply these mappings to a UML representation of the CIM Schema in order to derive a semantically valid and consistent formal CIM ontology.

1 Introduction

The increasing complexity of modern distributed systems has recently led to large-scale research initiatives in self-managing distributed systems; that is, distributed systems capable of configuring, optimizing, healing and protecting themselves [1]. A self-managing distributed system will, in general, be dependent upon reasoning mechanisms for inferring properties about its (distributed) operational domain. Such reasoning mechanisms will, in turn, be highly dependent upon information acquired from nodes within the distributed system. For example, a self-managing distributed system tasked with optimizing network performance between multiple nodes may need to reason over information (e.g., processor speed, memory capacity, packet loss, network configurations and other instrumentation) acquired from its nodes' CPUs, operating systems, network applications and network devices.

In order to effectively manage information generated by disparate applications, devices, components and subsystems from multiple nodes in a self-managing distributed system, such information must conform to a common model [2]. One standard that is expected to provide such a model is the Common Information Model (CIM) [3]. CIM is a comprehensive set of object-oriented models that specify concepts about a computing or network environment; it comprises a *core* model that defines a basic classification of elements and associations for a managed environment (e.g., logical and physical elements, capabilities, settings and profiles) as well as *common* models that define concepts that are common to particular management areas (e.g., applications, systems, devices and policies). The core and common models together are referred to as the *CIM Schema* [4].

Although the CIM Schema defines the models necessary for inferring properties about distributed systems, its specification as a *semi-formal* ontology [5,6,7,8] limits its ability to support important requirements of a self-managing distributed system including knowledge interoperability, knowledge aggregation and reasoning. These limitations are due, in part, to the constraints imposed by the languages (e.g., XML and XML Schema) used to model, represent and share semi-formal ontologies. With respect to knowledge interoperability, for example, such languages do not (1) provide globally-understood constructs for expressing semantics nor (2) impose a common interpretation of the meta-data contained within the model [9]. Thus, semi-formal ontologies encoded by these languages can only be used by those systems that have a complete and *a priori* understanding of the semantics surrounding the ontology. In an open self-managing distributed system environment, however, knowledge from a network node will need to interoperate among possibly several heterogeneous nodes. With respect to knowledge aggregation, XML-encoded ontologies cannot be arbitrarily combined with other ontologies in a flexible manner [9]. In a self-managing distributed system, however, knowledge about network nodes will need to be aggregated with other knowledge (e.g., domain knowledge). With respect to reasoning, XML-encoded ontologies do not embody the constructs for facilitating parsing, logical deduction or semantic interpretation. However, in a self-managing distributed system, reasoning over knowledge from network nodes will be necessary to infer the operational state of the distributed system.

In order to facilitate the interoperability and aggregation of, as well as the reasoning over, CIM-based knowledge in self-managing distributed systems, there is a need to model, represent and share CIM as a *formal* ontology; that is, an ontology that defines the semantics of its vocabulary by a complete and sound axiomatization [5,6,7,8]. One language that can be used to construct a formal CIM ontology is the Resource Description Framework (RDF) [10]. RDF is an assertional language that defines (domain-agnostic) semantic constructs for expressing propositions using precise formal vocabularies. In RDF, meta-data is defined using the vocabulary description language RDF Schema (RDFS) [11] that defines not how instances will be expressed (as is the case, for example, with XML Schema), but rather, provides a vocabulary for describing certain features of the data. This vocabulary can be reused in any setting allowing applications to infer properties about RDF/S-specified knowledge without having a prior understanding of the semantics surrounding that knowledge [12]. By using an RDF/S-based CIM ontology, the semantics of CIM-based instances can be partially deduced by systems that have no prior knowledge of the CIM Schema thereby fulfilling the interoperability requirement of a self-managing distributed system. In addition, different vocabularies can be arbitrarily combined to form new knowledge. This feature of RDF/S could allow, for example, knowledge from a CIM ontology to be more easily combined with knowledge from a specific domain ontology (e.g., grid systems) thereby fulfilling the aggregation requirement of a self-managing distributed system. RDF/S also allows for the expression of logical propositions from meta-data descriptions. These propositions can be arbitrarily combined to form a set of (semantically) connected propositions that, in turn, can directly serve as knowledge in a form required for reasoning thereby fulfilling the reasoning requirement of a self-managing distributed system.

In this paper, we aim toward the construction of a formal CIM ontology by proposing a framework based upon previous research that identified mappings from Unified Modeling Language (UML) [13] constructs to ontology language constructs. Here, we extend and apply these mappings to a UML representation of the CIM Schema in order to facilitate the derivation of a semantically valid and consistent CIM ontology. Although our approach provides a first step for constructing a consistent and semantically valid CIM ontology, we identify some issues that must be resolved before a complete, valid and consistent CIM ontology can be derived. We begin by describing the the mapping of CIM UML to CIM RDF/S and also describe limitations associated with these mappings. Next, we describe how the limitations of mapping CIM UML to CIM RDF/S can be overcome by mapping CIM UML to the more powerful Web Ontology Language (OWL). Finally, we identify issues that must be resolved before a complete, valid and consistent CIM ontology can be derived.

2 Constructing a CIM RDF/S Ontology

The construction of an ontology for describing the CIM Schema requires the mapping of CIM concepts to ontology language constructs. When mapping between CIM concepts to ontology language constructs, it is important to determine whether the mapping can preserve the semantics of the original CIM model. This is determined largely by whether the modeling languages used are *semantically equivalent*; that is, given two modeling languages L_1 and L_2, there is a one-to-one correspondence between the semantics of constructs in L_1 and the semantics of constructs in L_2 [14]. In an effort to construct a formal CIM ontology that preserves the semantics of the CIM Schema, we leverage previous research [15,16] that identified (one-way) mappings from UML constructs to RDF/S constructs by applying these mappings to a UML representation of the CIM Schema. By using these mappings, we not only increase the (semantic) validity of the resulting ontology but also ensure the consistency of mappings from CIM UML constructs including classes, attributes and relationships to CIM RDF/S ontology constructs including classes and properties. Table 1 shows an overview of some CIM concepts and their mappings to UML and RDF/S constructs. Note that mappings from a CIM concept to a specific construct in the target language should reflect a full (or at least an approximate) semantic correspondence. In some cases, a CIM concept is mapped to a set of constructs in the target language that fully (or approximately) reflects the semantics of the CIM concept. In other cases, there may be no constructs in the target language that semantically correspond to the CIM concept.

2.1 Mapping CIM Schema Classes and Properties

In the CIM Schema, concepts related to computing and network environments are represented primarily by UML classes. For example, the notion of a (hardware, software

Table 1. CIM Concepts and Related Mappings.

CIM Concept	*UML*	*RDF/S*	*OWL*
Named Element	✓	✓	✓
Class	✓	✓	✓
Property	✓	✧	✧
Method	✓	✧	✧
Generalization	✓	✓	✓
Association/Aggregation	✧	✧	✧
Cardinality	✓	✗	✓
Qualifiers (multiple)	✓✧✗	✓✧✗	✓✧✗
Datatypes	✓	✓	✓

✓ Maps to a specific construct (full/approx. semantic correspondence)

✧ Maps to a set of constructs (full/approx. semantic correspondence)

✗ No defined mapping (no semantic correspondence)

or service-oriented) `Product` is defined as a UML class as shown in Figure 1. Here, we represent a CIM Schema concept as a `rdfs:Class` class that corresponds to a generic notion of a type or category. To represent a CIM Schema concept as an `rdfs:Class` in an RDF/S statement, we use the `rdf:type` property that defines the resource (e.g., a `Product`) as a member of a particular class (e.g., `rdfs:Class`). In addition, we use the namespace prefix `cim:` to refer to the CIM Schema vocabulary defined by an XML namespace declaration such as `xmlns:cim="http://www.dmtf.org/CIMSchema28#"`. For example, we can represent the class `cim:Product` by the triple as shown in Figure 2 (line 5). In addition, we represent a CIM Schema property or method (defined as a UML attribute or operation, respectively) as an `rdf:Property` class. For example, we can represent the attribute `cim:Product.Vendor` by the triple shown in Figure 2 (line 6).

In RDF/S, each `rdf:Property` can be associated with a domain that specifies the class(es) on whose members a property can be used and a range that specifies the class(es) or datatype(s) to which the values of the property are restricted. For example, the property `cim:Product.Vendor` may be used by any instance of `cim:Product` and the value of `cim:Product.Vendor` must be a member of `xsd:string` as shown in Figure 2 (lines 7-8). Note that a mapping of CIM to RDF/S necessitates the mapping of UML datatypes to datatypes used by RDF/S. In UML, attributes may conform to a variety of intrinsic datatypes (e.g., integers, booleans and string datatypes). These datatypes can be mapped directly to those used by RDF/S as defined in the `xsd:` (XML Schema datatype) namespace [17]. For example, a boolean datatype in UML may be mapped to the `xsd:boolean` datatype in RDF/S.

2.2 Mapping CIM Schema Relationships

The structure of the CIM Schema defines a number of relationships between CIM classes. In Figure 1, for example, there exists a generalization relationship between `cim:Product` and `cim:ManagedElement`. In RDF/S, the UML generaliza-

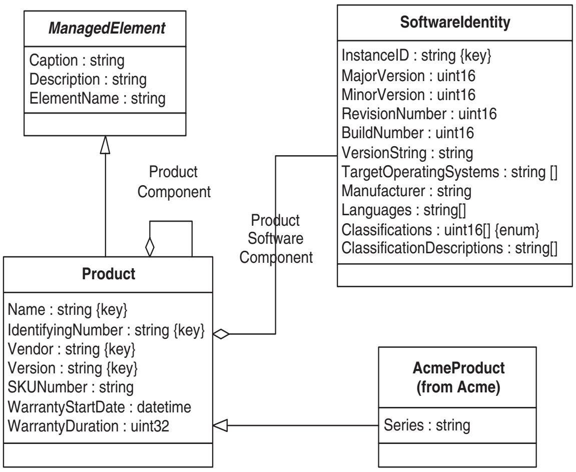

Fig. 1. CIM UML Classes.

tion relationship can be mapped directly to the `rdfs:subClassOf` property. For example, we may represent a generalization relationship between `cim:Product` and `cim:ManagedElement` by the triple shown in Figure 2 (line 9).

In addition to generalization, the CIM Schema also defines other relationships including association and aggregation relationships.[1] In UML, such relationships are represented by association classes. For example, the `cim:ProductSoftwareComponent` association class represents the "ProductSoftwareComponent" aggregation in Figure 1. Since a UML association class is a subclass of a UML class, we may represent `cim:ProductSoftwareComponent` as an `rdfs:Class` as shown in Figure 2 (line 15). Association classes in the CIM Schema include references that define the set of classes that may be linked by the association or aggregation relationship. Each reference is an `rdf:Property` that is represented by the term `cim:REF`. For example, `cim:ProductSoftwareComponent` contains the `cim:REF` `cim.ProductSoftwareComponent.GroupComponent` that is used by instances of `cim:ProductSoftwareComponent` to define the range of classes (i.e., `cim:Product`) that can be associated with one or more `cim:SoftwareIdentity` classes. Similarly, the `cim:REF` `cim.ProductSoftwareComponent.PartComponent` is used by instances of `cim:ProductSoftwareComponent` to define the range of classes (i.e., `cim:SoftwareIdentity`) that can be used to describe a `cim:Product`. These reference definitions are shown in Figure 2 (lines 16-21).

With respect to associations and aggregations, UML roles can also be represented in RDF/S as an `rdf:Property`. To represent UML roles in the CIM ontology, we augment the CIM Schema vocabularies by defining role names using the convention "<*vocabulary*> : *has*<*RoleName*>". For example, a `cim:Product` may have the role `cim:hasSoftwareComponent` with respect to `cim:SoftwareIdentity`. Conversely,

[1] Note that an aggregation relationship is a specialization of an association relationship.

```
1   // ManagedElement class
2   cim:ManagedElement rdf:type rdfs:Class .
3
4   // Product class
5   cim:Product rdf:type rdfs:Class .
6   cim:Product.Vendor rdf:type rdf:Property .
7   cim:Product.Vendor rdfs:domain cim:Product .
8   cim:Product.Vendor rdfs:range xsd:string .
9   cim:Product rdfs:subClassOf cim:ManagedElement .
10
11  // SoftwareIdentity class
12  cim:SoftwareIdentity rdf:type rdfs:Class
13
14  // ProductSoftwareComponent (association class)
15  cim:ProductSoftwareComponent rdf:type rdfs:Class .
16  cim:ProductSoftwareComponent.GroupComponent rdf:type cim:REF .
17  cim:ProductSoftwareComponent.GroupComponent rdf:domain cim:ProductSoftwareComponent .
18  cim:ProductSoftwareComponent.GroupComponent rdf:range cim:Product .
19  cim:ProductSoftwareComponent.PartComponent rdf:type cim:REF .
20  cim:ProductSoftwareComponent.PartComponent rdf:domain cim:ProductSoftwareComponent .
21  cim:ProductSoftwareComponent.PartComponent rdf:range cim:SoftwareIdentity .
22
23  // roles
24  cim:hasSoftwareComponent rdf:type rdf:Property .
25  cim:hasSoftwareComponent rdf:domain cim:Product .
26  cim:hasSoftwareComponent rdf:range cim:SoftwareIdentity .
27  cim:isSoftwareComponentOf rdf:type rdf:Property .
28  cim:isSoftwareComponentOf rdf:domain cim:SoftwareIdentity .
29  cim:isSoftwareComponentOf rdf:range cim:Product .
30
31  // AcmeProduct class
32  acme:AcmeProduct rdf:type rdfs:Class .
33  acme:AcmeProduct rdfs:subClassOf cim:Product .
```

Fig. 2. CIM RDF/S Statements.

`cim:SoftwareIdentity` may have the role `cim:isSoftwareComponentOf` with respect to `cim:Product`. These examples are shown in Figure 2 (lines 24-29).

The CIM Schema may also be extended by vendors to suit their particular needs. These vendor-specified extensions to the CIM Schema are referred to as *extension schemas*. Extension schemas extend the CIM Schema by defining classes that inherit from CIM Schema classes. In Figure 1, for example, the vendor-defined class `AcmeProduct` inherits from the CIM class `Product`. Because an RDF/S ontology may comprise multiple vocabularies, relationships between concepts from different vocabularies can be defined. This feature of RDF/S allows for the construction of an ontology that defines relationships between concepts in the CIM Schema and concepts defined in a vendor-specified extension schema. For example, we may express the generalization relationship between `acme:AcmeProduct` and `cim:Product` as shown in Figure 2 (lines 32-33).

3 Constructing a CIM OWL Ontology

Although RDF/S may be used to express *some* types of knowledge, it deliberately lacks sufficient constructs to adequately express all types of knowledge [15,18]. As shown in Table 1, for example, RDF/S does not provide constructs for expressing cardinality restrictions such as those used for describing association or aggregation relationships between CIM classes. In addition, RDF/S cannot adequately express some CIM *qualifiers* that are used to define constraints on CIM properties. For example, RDF/S cannot directly represent the semantics of the CIM 'key' qualifier which is used to indicate that the value of a CIM property must be unique for all instances of a particular class. Such qualifiers are currently denoted in CIM UML as shown in Figure 1.

The limited expressivity of RDF/S motivated the development of the (semantically-richer) DAML+OIL (DARPA Agent Markup Language and Ontology Inference Layer) ontology markup language [19] which, in turn, formed the basis of the Web Ontology Language (OWL) [20]. OWL is an RDF/S-based language that can explicitly represent the meaning of terms in vocabularies and the relationships between those terms. OWL enhances the expressivity of RDF/S (and DAML+OIL) by adding more vocabulary for describing properties and classes including relations between classes (e.g. disjointness), cardinality, equality, richer typing of properties, characteristics of properties (e.g. symmetry) and enumerated classes. To map the CIM Schema to OWL[2], we extend previous work that identified mappings from UML constructs to DAML+OIL constructs [14,21] as well as preliminary work that identified mappings from UML constructs to OWL constructs [22]. An overview of the mappings from CIM concepts to OWL constructs are shown in Table 1.

One concept that is required for expressing the CIM Schema, but not included in RDF/S, is the notion of cardinality. In OWL, cardinality is stated on a property with respect to a class. For example, `cim:Product` has a minimum and maximum cardinality of one on the `rdf:Property` `cim:hasSoftwareComponent`. This cardinality restriction can be expressed by the triples shown in Figure 2 (lines 2-4).

Another important CIM concept that cannot be adequately expressed using RDF/S is the concept of uniqueness. In OWL, the `owl:FunctionalProperty` restriction may be used to define mutually distinct properties. With respect to CIM, this is important for ensuring that (1) class names (including association names) are unique within the schema, (2) method names are unique within the domain class and (3) reference names are unique within the scope of the defining association. For example, `owl:FunctionalProperty` may be used to realize the CIM 'key' qualifier on the attribute `cim:Product.Name` as shown in Figure 2 (line 7).

Yet another limitation of RDF/S is its inability to express local range restrictions associated with a particular property for a particular class. Here, if an instance of a class is related by a property to a second object, then the second object can be inferred to be an instance of the local range restriction class. Local range restrictions on a property can be represented using the `owl:allValuesFrom` restriction. For example, `cim:Product` may have `rdf:Property` `cim:hasSoftwareComponent` restricted to have `owl:allValuesFrom` `cim:SoftwareIdentity`. This means that if an instance of `cim:Product` is related by the `rdf:Property` `cim:hasSoftwareComponent` to the instance “Printer Driver”, then from this a reasoner can deduce that “Printer Driver” is an instance of the class `cim:SoftwareIdentity`. This example is shown in Figure 2 (lines 10-12).

[2] In this paper, we consider only OWL DL.

```
1   // cardinality
2   _:SWIdentityRole rdf:type owl:Restriction .
3   _:SWIdentityRole owl:onProperty cim:Product.hasSoftwareComponent .
4   _:SWIdentityRole owl:cardinality "1"^^xsd:nonNegativeInteger .
5
6   // uniqueness constraints
7   cim:Product.Name rdf:type owl:FunctionalProperty .
8
9   // local range restrictions
10  _:SWIdentityRole rdf:type owl:Restriction .
11  _:SWIdentityRole owl:onProperty cim:Product.hasSoftwareComponent .
12  _:SWIdentityRole owl:allValuesFrom cim:SoftwareIdentity .
13
14  // transitivity
15  _:ProductCompRole rdf:type owl:Restriction .
16  _:ProductCompRole owl:onProperty cim:Product.hasProductComponent .
17  _:ProductCompRole rdf:type owl:TransitiveProperty .
18
19  // inverse
20  cim:hasSoftwareComponent owl:inverseOf cim:isSoftwareComponentOf .
21
22  // property equivalence
23  cim:Product.Vendor owl:equivalentProperty cim:SoftwareIdentity.Manufacturer .
24
25  // imports
26  # owl:imports rdf:resource="http://www.dmtf.org/CIMSchema28" .
```

Fig. 3. CIM OWL Statements.

OWL can also be used to enhance the semantics surrounding relationships between CIM classes. For example, the `owl:TransitiveProperty` property may be used to express a chain of `cim:ProductComponent` associations on a `cim:Product` as shown in Figure 2 (lines 15-17). In addition, the `owl:inverseOf` property may be used to state the inverse of a relationship. For example, an instance of `cim:Product` may have a `cim:hasSoftwareComponent` role with respect to one or more instances of `cim:SoftwareIdentity`. If `cim:isSoftwareComponentOf` is the inverse of `cim:hasSoftwareComponent`, and there exists a `cim:Product` with property `cim:hasSWFeature` and property value `cim:SoftwareFeature`, then a reasoner can deduce that `cim:SoftwareIdentity` `cim:isSoftwareComponentOf` `cim:Product` where `cim:isSoftwareComponentOf` is the `owl:inverseOf` `cim:hasSoftwareComponent` as shown in Figure 2 (line 20).

An important advantage of using OWL to define a CIM ontology is that OWL provides constructs for defining equivalence between classes and properties. In OWL, class and property equivalence is represented using the `owl:equivalentClass` and `owl:equivalentProperty` axioms. For example, the property `cim:Product.Vendor` may be synonymous with `cim:SoftwareIdentity.Manufacturer` in those environments where all products from a specific company are developed in-house (and thus, the name of the vendor is the same as the manufacturer). This equivalence relationship is shown in Figure 2 (line 23). The OWL equivalence axioms may also be used to facilitate interoperability between, and the merging of, ontologies by describing equivalence relationships between classes and properties defined in CIM and those defined, for example, in the IEEE Standard Upper Ontology (SUO) [23]. In addition, the `owl:imports` construct may be used to reference another OWL ontology containing definitions whose semantics are considered to be part of the meaning of the importing ontology. This ontology-importing feature can be used to combine a CIM ontology with other ontologies. For example, an ontology that describes military systems may import the CIM ontology to describe the management of its computer and network systems by using the OWL statement as shown in Figure 2 (line 26).

4 Current Limitations

Although the proposed framework provides a first step toward the construction of a formal CIM ontology, a number of issues must first be resolved before such an ontology can be considered consistent, semantically valid, and complete. One issue concerns the mapping of some CIM concepts to UML constructs. Currently, while many CIM concepts can be mapped directly to UML constructs (e.g., CIM Class), some CIM concepts (e.g., some CIM qualifiers) cannot; thus, such concepts must be added to the ontology in an ad-hoc fashion leading to a possibly inconsistent or invalid ontology.

Another related issue concerns CIM concepts (particularly, CIM qualifiers) that have no mappings to either UML constructs or OWL constructs. For example, the notion of a CIM default value does not have a corresponding UML construct nor a corresponding OWL construct. In cases where no direct mapping from CIM concepts to specific terms in the RDF/S or OWL vocabulary can be derived, we define such concepts within the `cim:` vocabulary (e.g., we define `cim:default` as an `rdf:Property` that is used to represent the notion of a default value).

Another issue concerns the inability of OWL to fully express some UML constructs [14,21,22]. For example, OWL provides no constructs for adequately representing a UML abstract class. Thus, some CIM concepts (e.g., `ManagedElement` that is represented by a UML abstract class) cannot be mapped directly to RDF/S or OWL constructs and must be manually included in the `cim:` vocabulary.

Finally, both RDF/S and OWL can only partially express some CIM Schema semantics. For example, CIM methods can be defined as an `rdf:Property`, but cannot be mapped to OWL construct(s) that more closely match the notion of a UML operation.

5 Related Work

The idea of using CIM for facilitating self-managing systems has been previously described by Bantz [24] who proposed the use of CIM information to facilitate decision making in autonomic computing, and by Ganek [2] who proposed the use of CIM to facilitate interoperability between heterogeneous autonomic computing elements.

In addition, the idea of using CIM ontologies for describing management knowledge has been previously proposed. For example, López de Vergara *et. al* [25] proposed an algorithm for mapping CIM, as well as other information models, into a common DAML+OIL ontology. Also, Lavinal *et. al* [26] proposed the construction of a CIM ontology using OKBC while Tangmunarunkit [27] proposed the use of an RDF/S-based CIM ontology for grid computing. Finally, Lanfranchi *et. al* [28] defined a mapping of CIM to the description logic DLR.

The work presented in this paper, however, is distinguished from these previous efforts in that this work leverages and extends research that identified mappings from UML constructs to ontology language constructs in order to achieve semantically valid and consistent mappings from CIM UML to a formal CIM ontology. Previous research in mapping UML to RDF/S and DAML+OIL included works by Chang [15] who described mappings from UML to RDF/S and Cranefield [16] who described an automatic mapping from UML to RDF/S using XMI. In addition, Backlawski *et. al* [14], Falkoych *et. al* [21] and Kogut *et. al* [29] described mappings from UML to DAML+OIL constructs while AT&T [22] described a preliminary analysis of mapping UML to OWL Full.

6 Conclusion

This paper proposed a framework for constructing a formal CIM ontology based upon previously-identified mappings from UML constructs to RDF/S and OWL constructs. We began by presenting details about the mapping of CIM classes, properties and relationships to RDF/S constructs as well as defined a vocabulary for representing concepts associated with the CIM Schema. Given the limitations of RDF/S for expressing the semantics of CIM concepts, we described the mapping of these concepts to the RDF/S-based ontology language OWL. Although OWL provides enhanced expressivity over RDF/S, there exists some CIM concepts that cannot be directly mapped to OWL constructs. In such cases, we proposed the definition of these concepts within the context of the `cim:` vocabulary.

Currently, the specifications for CIM, UML, and OWL are fluid; thus, future research in this area will consider how changes to these specifications affect the derivation of a CIM ontology. Future versions of CIM, for example, are expected to reflect possibly significant changes to the CIM Schema as well as its representation in the forthcoming UML 2.0 (e.g., the use of UML roles for representing CIM qualifiers). Such changes might facilitate the derivation of a more complete CIM ontology by providing a more complete mapping between CIM concepts and UML constructs (and thus, a more direct mapping of CIM concepts, such as qualifiers, to RDF/S and OWL). In addition, research continues on mapping UML constructs to OWL constructs that will directly impact the derivation of a CIM ontology. It is expected that the framework proposed in this paper could be easily extended to support these anticipated changes.

References

1. Kephart, J., Chess, D.: The vision of autonomic computing. IEEE Computer **36** (2003)
2. Ganek, A., Corbi, T.: The dawning of the autonomic computing era. IBM Systems Journal **42** (2003)
3. Distributed Management Task Force: Common information model (CIM) specification version 2.2 (1999)
4. Distributed Management Task Force: CIM schema: Version 2.7 (2003)
5. Fox, M.S., Gruninger, M.: Enterprise modeling. AI Magazine **19** (1998)
6. Gruber, T.: It is what it does: The pragmatics of ontology for knowledge sharing. In: Proceedings of the International CIDOC CRM Symposium, Washington DC (2003)

7. López de Vergara, J., Villagrá, V., Asensio, J., Berrocal, J.: Ontologies: Giving semantics to network management models. IEEE Network **17** (2003)
8. Spyns, P., Meersman, R., Jarrar, M.: Data modelling and ontology engineering. ACM SIGMOD Record **31** (2002)
9. Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdman, M., Horrocks, I.: The semantic web: The roles of XML and RDF. IEEE Internet Computing (2002)
10. World Wide Web Consortium: Resource description framework (RDF) model and syntax specification (1999)
11. World Wide Web Consortium: RDF vocabulary description language 1.0: RDF schema (2003)
12. Nilsson, M., Palmer, M., Naeve, A.: Semantic web meta-data for e-learning – some architectural guidelines. In: Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii (2002)
13. Object Management Group: Unified modeling language (UML) version 1.5 (2003)
14. Baclawski, K., Kokar, M.K., Kogut, P.A., Hart, L., Smith, J., Holmes III, W.S., Letkowski, J., Aronson, M.L.: Extending UML to support ontology engineering for the semantic web. Lecture Notes in Computer Science **2185** (2001)
15. Chang, W.W.: A discussion of the relationship between RDF-schema and UML, W3C Note (1998)
16. Cranefield, S.: Networked knowledge representation and exchange using UML and RDF. Journal of Digital Information **1** (2001)
17. World Wide Web Consortium: XML schema part 2: Datatypes (2001)
18. Antoniou, G., van Harmelen, F.: Web ontology language: OWL. In Staab, S., Studer, R., eds.: Handbook on Ontologies in Information Systems, Springer-Verlag (2003)
19. van Harmelen, F., Petel-Schneider, P.F., Horrocks, I.: Reference description of the DAML+OIL ontology markup language (2001)
20. World Wide Web Consortium: OWL web ontology language semantics and abstract syntax (2003)
21. Falkovych, K., Sabou, M., Stuckenschmidt, H.: UML for the semantic web: Transformation-based approaches. In Omelayenko, B., Klein, M., eds.: Knowledge Transformation for the Semantic Web, IOS Press (2003)
22. AT&T: OWL Full and UML 2.0 compared. Whitepaper (2004)
23. Niles, I., Pease, A.: Towards a standard upper ontology. In: Proceedings of the International Conference on Formal Ontology in Information Systems, Ogunquit, Maine (2001)
24. Bantz, D.F., Bisdikian, C., Challener, D., Karidis, J.P., Mastrianni, S., Mohindra, A., Shea, D.G., Vanover, M.: Autonomic personal computing. IBM Systems Journal **42** (2003)
25. López de Vergara, J., Villagrá, V., Berrocal, J.: An ontology-based method to merge and map management information models. In: Proceedings of the HP Openview University Association Tenth Plenary Workshop, Geneva, Switzerland (2003)
26. Lavinal, E., Desprats, T., Raynaud, Y.: A conceptual framework for building CIM-based ontologies. In: Proceedings of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM 2003), Colorado Springs, Colorado (2003)
27. Tangmunarunkit, H., Decker, S., Kesselman, C.: Ontology-based resource matching in the grid – the grid meets the semantic web. In: Proceedings of 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida (2003)
28. Lanfranchi, G., Della Peruta, P., Perrone, A., Calvanese, D.: Toward a new landscape of systems management in an autonomic computing environment. IBM Systems Journal **41** (2003)
29. Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., Smith, J.: UML for ontology development. Knowledge Engineering Review Journal Special Issue on Ontologies in Agent Systems **17** (2002)