International Journal of Digital Information and Wireless Communications (IJDIWC) 1(1): 94-108
The Society of Digital Information and Wireless Communications, 2011(ISSN 2225-658X)

Towards the Use of a Computing Ontology:
A Curricula Management System Proposal

Adelina Tang
Sunway University
No. 5, Jalan Universiti, Bandar Sunway,
46150 Petaling Jaya, Selangor Darul Ehsan, Malaysia
adelina.tang@ieee.org

ABSTRACT

Semantic Web components are proposed to develop a Computing Curricula Management System. Such a system is envisaged to alleviate the labour-intensive and time-consuming activities that are involved in curricula development, maintenance and management. Emphasis is placed on the outcomes-based approach to curricula development. An ontology is used as a knowledge source that can be used to interact with a curricula wiki facility through ontological agents. The ontology is created with the Protégé Ontology Editor, whereas the major features of the agents are designed according to the MAS-CommonKADS Agent Oriented Methodology. However, manipulation of the completed Computing Ontology is found to be limited. As the ontology could be viewed as a Resource Description Framework document that is editable using OWL-DL, access and manipulation of the ontological objects and relationships could then be conducted through an Application Programming Interface that is yet to be developed. This article examines the potential of doing so and concludes with a discussion of the resulting contribution, limitation and future work.

KEYWORDS

Ontology manipulation, agents, MAS-CommonKADS, API, computing curricula

1 INTRODUCTION

An ontology facilitates the sharing of knowledge. It is a specification of a representational vocabulary for a shared domain of discourse, as defined by Gruber [1]. The shared domain of discourse would consist of classes, relations, functions, and similar objects of interest. It works with the Resource Description Framework (RDF) which has its roots in eXtensible Markup Language (XML) to form the basis of the Semantic Web envisaged by Berners-Lee *et al.* [2].

The key to successful Semantic Web applications is the capability to facilitate knowledge-sharing with minimum or, ideally, no human intervention. This forms the motivation for adopting this approach to develop online versions of standalone applications that have feature-rich web services.

One such type of application is the Computing Ontology project by Cassel *et al.* [3]. At the time of their project, some limitations existed. This provided

the motivation for Tang and Lee's Computing Curricula Management System (CCMS) project [4]. The CCMS adopted the scheme suggested by Cassel *et al.* [5] and deployed the MAS-CommonKADS Agent Oriented Methodology [6, 7].
Although the Computing Ontology had been tested and verified with the FACT++ reasoner, there remain issues with manipulating and extracting ontology information from within the Protégé ontology editor [7].

This article will discuss the background, supporting literature and motivation for the CCMS, with emphasis on the Design phase of the methodology. It will then propose ways to resolve the manipulation and extraction of ontology information externally, outside the ontology editor, through an Application Programming Interface (API). It will conclude with resulting contribution, limitation and future work possibilities.

2 BACKGROUND

2.1 Semantic Web Components

In [2], Berners-Lee *et al.* famously declared that three critical factors are required to realise the Semantic Web. Two are technologies, eXtensible Markup Language (XML) and Resource Description Framework (RDF), and the third is the ontology as a collection of information.

XML allows tags to be created by users for their own specialized needs. This flexibility enables user-defined structures. However, it does not provide any additional information regarding the real meaning of those structures [8].

RDF, on the other hand, can be used to express meaning which is encoded in a set of triples, which are XML tags [8]. The information contained in a RDF document would make assertions that particular objects (e.g. a person, a web page) have certain properties (e.g. "is friend of") with particular values (e.g. another person, another web page) [2]. In this way, it becomes easy to create machine readable semantics that facilitate computers across the Semantic Web to access, extract, and manipulate information to produce a more useful, meaningful and customized experience for a human user.

An ontology is a document or file that formally defines objects in a particular domain and the relationships among those objects. These objects are grouped within classes and subclasses [2]. The attributes of these objects and the relationships arising from such groupings provide useful meaning (semantics) to other computers when required. Ontologies may be created using ontology editors like WebOnto, Ontolingua, Protégé, among the myriad that are available [9]. In addition, they could also be developed using the Web Ontology Language based on Description Logics (OWL-DL), where DL is a decidable fragment of the first order logic, and therefore, more expressive and amenable to automated reasoning [8].

In the context of this project, the Computing Ontology, created in Protégé, could not be manipulated within the

editor. However, if one was to save the ontology as a text file, it could then be accessed and manipulated as easily as any regular RDF document.

Nonetheless, the issue of actually extracting and manipulating the ontological relationships remains challenging and would require a different approach.

2.2 Ontologies And Software Agents

Ontologies require operators to retrieve relevant information from them and to learn useful relationships that may exist among their objects. Medina *et al.* [10] utilized agents and ontologies to retrieve information from a set of federated digital libraries. Their approach was to adopt the MAS-CommonKADS Agent-Oriented Methodology (AOM) to model their software agents. These agents were called ontological agents as they operated with ontologies.

The agent approach is attractive as it is a natural way to model systems consisting of multiple, distinct and independent components. Agents enable functionalities such as planning, learning, communication or coordination and are perceived as a natural extension of current component-based approaches [11]. A system consisting of several agents that support inter-agent and agent-environment interaction within a digital system is called a Multi-Agent System (MAS).

The MAS-CommonKADS AOM is an extension of the CommonKADS methodology [6]. CommonKADS uses Object-Oriented concepts and techniques to facilitate Knowledge Acquisition, Engineering and Management under the ESPRIT IT initiative [12]. It provides the methods to perform a detailed analysis of knowledge-intensive tasks and processes. It is little wonder, then, that this has become the de facto European standard for knowledge analysis and knowledge-intensive system development.

2.3 Similar Work

Ontologies have gained a wide following among researchers. The application of ontologies in education management is a popular combination owing to its intrinsic structure around which knowledge bases can be built [1]. It also provides a systematic design rationale of a knowledge base according to its particular domain or context of interest. Some of the more notable projects are discussed here.

2.3.1 The Knowledge Web

One of the largest ones to-date, the Knowledge Web (KW) project, a 6th European Union Framework Programme (FP6) project, is a prime example [13]. The objective of this project was to ensure the successful transition of ontology technology from academia to industry to enable e-work and e-commerce.

In its outreach to education, the KW committee utilized novel semantic web technologies in combination with more traditional e-learning approaches. They

also contributed to the establishment of teaching standards through the communication of methodologies and best practices. Other standards, like those in the World Wide Web consortium (W3C), are also complied with.

2.3.2 Ontologies For Education

Some projects were more focussed. Dicheva *et al.* [14] presented the Ontologies for Education (O4E) facility overview that also reported on the web portal's coverage of research projects and successfully implemented practices. Their ontological overview presents two major aspects, one is in "building" ontologies for education and the other is in "using" them.

In the context of this project, the "using" aspect appears to be more appropriate. It is divided into the technological and the application perspectives. The technological perspective defines three research areas, with two of them, Knowledge Representation (KR) and Information Retrieval (IR) directly contributing to the third, the Semantic Web. The application perspective defines two areas, that of applying the ontology as a cognitive tool and that of applying it according to the type of domain knowledge (see Figure 1).

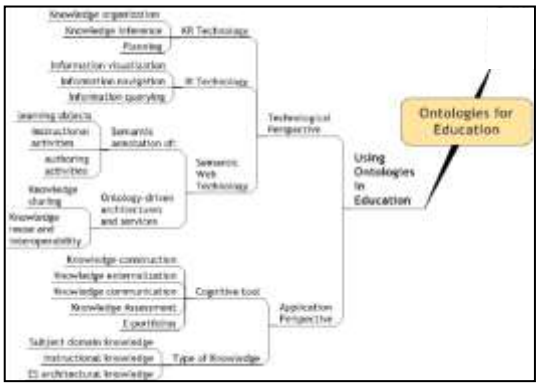

Figure 1. State-of-the-art ontology of Ontological Technologies for Education [14].

2.3.3 Computing Ontology

Further to these perspectives, Cassel *et al.* proposed their Computing Ontology project that compressed the five distinct fields, Computer Engineering, Computer Science, Information Systems, Information Technology, and Software Engineering into one generic Computing field [3]. Their primary objective was to connect the comprehensive list of typical computing topics with curriculum development and subject planning activities. Thereafter, a prototype system for matching subject topics and outcomes would emerge. In an earlier paper, the authors had proposed a web-based utility to enable a subject developer to select or create outcomes as well as to select suitable topics that could achieve those outcomes [5].

A study of Cassel's ontology revealed that their "Flash" displays were not linked with actual curricula objects in their ontology. This indicated that actual curricula management could not be carried out successfully. In addition, the authors also admitted that their ontology

did not specify curriculum requirements at that time [15].

2.4 Outcome-Based Curricula

An outcome-based curriculum has its roots in Outcome-Based Education (OBE). According to Acharya [16], OBE is a method of curriculum design and teaching that focuses on arming students with knowledge, skills and related professional capabilities after undergoing the particular program. The desired outcome is selected first and the corresponding curriculum and other teaching and learning aids would then be created to support the required outcome [17]. In this manner, lower level subject outcomes would collectively contribute to the top level program outcomes.

Professional bodies and government agencies have moved away from subject objectives to a specification of characteristics of graduates, often in the form of program learning outcomes [18-24]. In Malaysia, the MQA introduced the Malaysian Qualifications Framework (MQF) as a point of reference to explain and clarify qualifications and academic achievement in Malaysian higher education and how these qualifications might be linked. The MQF is intended to provide a detailed description of the Malaysian education system to an international audience [25].

The MQF advises educators to shift their Teaching-Learning focus from the conventional Teacher-centered emphasis to the more progressive Student-centred emphasis. With the focus sharply on the role played by "Learning Outcomes" within Student-centered Teaching-Learning, all curricula are to be designed through the mapping of the subject and program outcomes onto the eight Learning Outcome Domains (LODs) (see Figure 2).

Of particular relevance to outcome-based curricula and to this project is Cassel *et al.*'s proposal [5], in which they suggested that their Computing Ontology could be used to fulfil the requirements of outcome-based curricula.

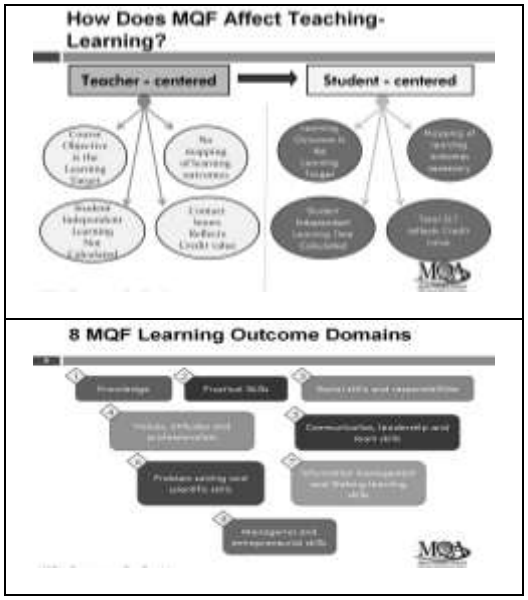

Figure 2. (*top*) Shift of focus to *Student-centered* Teaching-Learning. (*bottom*) *Learning Outcome Domains* defined by the MQF.

Closer to home in Malaysia, such requirements are established through the MQF [25]. However, the use of technology to maintain such standards in curriculum management is less common. This project proposes this approach and hopes to introduce a different perspective to local practitioners / education providers and managers.

2.5 Application Programming Interface

An Application Programming Interface (API) is a set of rules and specifications, in the form of program code that other programs can follow to communicate with each other [26]. As its name suggests, it interfaces between different programs and facilitates their interaction. It defines the proper way for a developer to request services from that program [27].

APIs can be used in many ways. They can be created to act for regular applications, to facilitate the calling of function libraries, as well as to request for special services from operating systems. Regardless of the need, the main responsibility of an API is to ensure that it is linkable to another program to perform a set of specified tasks.

An API can be considered a great asset by an organization. From the user's perspective, a good API should be easy to learn and to use, even without documentation [28]. From the developer's perspective, it should be easy to read, to extend and to maintain code that interfaces with it.

Specific to this project, it is proposed that the API deal with the scenario involving object-oriented languages as it is likely that Java or C++ would be used to develop the rest of the CCMS that interacts with the ontology [26].

However, one must remember that it is unlikely that the API would remain in its proposed form for the lifetime of the CCMS. A few years of real-world use would surely expose any mistakes that might have been introduced in its development. It is a natural expectation that all APIs will evolve with usage and a changing operating environment [28].

3 METHODOLOGY

Further to [4], this project proposes to exploit its resulting agent-interaction models to design and develop the Computing Curricula Management System (CCMS) based on the scheme proposed by Cassel *et al.* [5] (see Figure 3).

Figure 3. Proposed CCMS scheme from [5].

In Figure 3, the "Existing Curriculum" is fixed and not editable by the larger Computing community. The "Curriculum Wiki" is an editable environment which allows suggested additions and revisions to be introduced into a copy of the existing curriculum, i.e. a working/discussion copy. These activities are made transparent to all members of the Computing community. The third component, the "Discussion Forum", enables discussions regarding the aforementioned additions and revisions.

Four typical curricula management operations were identified – two

searches from keywords, one subject exemption request, and one subject syllabus change request.

The MAS-CommonKADS AOM consists of the Conceptualization, Analysis and Design phases [6]. The first two phases were applied to the curricula management operations in [4], to which the reader is directed for a more detailed treatment.

Recall that there was a limitation in manipulating and extracting information for the ontology [7]. The proposed solution then was to develop an API to resolve the issue. It is suggested that the API approach would be designed and developed following recommendations in [28].

This article focuses on the design and development issues of the CCMS.

3.1 Design

This phase develops the Design model that consists of three sub-models.

These are the *agent network design* for designing the relevant aspects of the agent network infrastructure (includes the network, knowledge and coordination/telematics facilities); the *agent design* for dividing/composing the agents according to the CCMS operating criteria and selecting suitable agent architectures, and the *platform design* for selecting the appropriate agent development platform [6].

The API design considerations will also be included. This is not part of the MAS-CommonKADS Design model, but will propose, instead, to extend the ideas presented in the *agent network design* sub-model.

3.1.1 Agent Network Design Sub-Model

The agent network infrastructure requires its own set of network agents to maintain this infrastructure. These agents would ensure that the following required facilities are operable:

Network Facilities. This would provide agent name services, security levels checking and authentication and/or encryption, as well as the transport protocol TCP/IP.

Knowledge Facilities. The Computing Ontology is the source of requisite knowledge for the CCMS (see Figure 3). Access to this facility would depend on the next facility.

Coordination/Telematics Facilities. A "police agent" is required to ensure that the "Existing Curriculum" (see Figure 3) is fixed and not editable by just any user. It is only the "Curriculum Wiki" that is editable as the working/discussion copy.

3.1.2 Agent Design Sub-Model

The agent architecture is dependent upon the Platform Design.

3.1.3 Platform Design Sub-Model

Software includes the Protégé 4.1 OWL ontology editor with built-in FACT++ reasoner and the Ontograf visualization plug-in [29]. The JAVA programming language is used to create the various

agents defined in each phase. The operating system is Microsoft XP, while the proposed hardware set-up consists of a Web server, three client workstations, and a backup facility.

3.1.4 API Considerations

The specifications are kept short to ensure that they will be easy to modify. APIs are expected to do one thing only and they should do that one thing well [28]. The API should be small to maintain its agility and ease of operation, without placing a burden on computational requirements.

The name of the API would reflect its purpose, i.e. `extract_relationship`.

It is good practice to "hide" information from and to the calling functions and the API. This facilitates the building, testing, and debugging in a modular and independent manner [28].

Its classes and members would be designated `private`, while all public classes would have no public fields. Constants peculiar to individual applications are an exception to this rule. In this manner, the method of implementation would not affect the operation of the API [28].

Recall that the *network facilities* of the *agent network design* sub-model provides the transport protocol TCP/IP. In this instance, an API could be used to implement a protocol [26]. This API would be designed to hide the details of the TCP/IP. Therefore, the protocol becomes transparent as if it is the API that is facilitating the function call. In this manner, the API would be providing a library to be used directly, with no physical transport, but only basic information exchange through function calls [26].

3.2 System Development

The development activities are focused on the Computing Ontology, the Curriculum Wiki, and the interface between the two (see Figure 3). These correspond respectively, to the *Knowledge* and *Coordination/Telematics Facilities*, in the *agent network design* sub-model.

3.2.1 Computing Ontology

This was built according to a generic knowledge engineering approach with emphasis on the *Knowledge Acquisition* and *Knowledge Implementation* activities as these are most relevant to this model, according to Studer *et al.* [30]. The source of information for the former was obtained from [31], whereas the latter is central to the design of the Ontology shown in Figure 3.

Knowledge Acquisition. The first step to building the Computing Ontology was to study the relationship between Programme Learning Outcomes (PLOs) and MQF Learning Outcome Domains (LOs). There are nine PLOs and eight LOs, and each PLO fulfils a different set of LOs (see Figure 4). Figure 5 shows the list of subjects in the BSc (Hons) Computer Science of Sunway University mapped onto the PLOs. The final step involved retrieving individual subject syllabus and pre-requisites, if any, from

the respective subject information document (see Figure 6).

Knowledge Implementation. The acquired knowledge was then implemented in a Protégé class hierarchy representation in Figure 7. This structure allows direct retrieval of information, with subjects sorted according to year and every topic is a subclass of its parent subject.

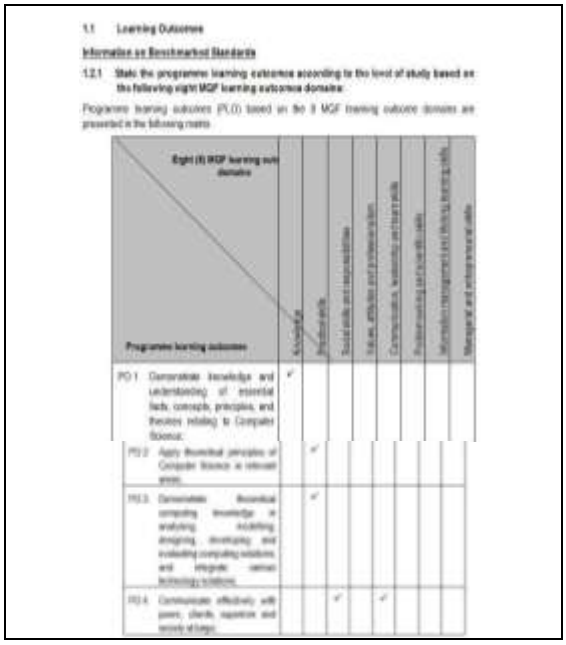

Figure 4. Sample of the mapping of PLOs onto MQF LOs [31].

No	Code	Subject title	Credits	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9
1.	CSC1101	Computer Fundamentals	3	✓	✓			✓				
2.	PRG1101	Programming Concepts and Problem Solving	3	✓	✓					✓		
3.	PRG1102	Programming Principles	3	✓	✓	✓						
4.	MTH1101	Calculus	3	✓	✓	✓				✓		
5.	UCB1004	English for Computer Technology Studies	4				✓	✓	✓			
6.	IEG1201	Database Fundamentals	4	✓	✓	✓						
7.	MTH1202	Probability and Statistics	4	✓	✓				✓	✓		

Figure 5. Sample of the list of BSc (Hons) Computer Science subjects mapped onto PLOs [31].

The relationships that exist among classes in the ontology are defined by object properties shown in Figure 8. As an example, the property *hasTopic* describes *Calculus hasTopic Integrals*, while its inverse property *isTopicOf* describes *Integrals isTopicOf Calculus*.

12. Subject code and name: CSC2103 Data Structures and Algorithms

11.	Name(s) of academic staff	Tan Ee Wee
12.	Rationale for the inclusion of the subject in the programme	This subject teaches the student: • The concept of data structures and its use in programming • The practical usage of different types of data structures, and its effects on the computer program architecture • The methods of sorting and searching data collections, and basic understanding of graph theory and path finding algorithms • Various algorithmic strategies, their design and implementation through some basic computing algorithms
13.	Semester and year offered	Semester 1, Year 2
14.	Credit value	4
15.	Prerequisite (if any)	PRG1102
16.	Subject learning outcomes and mapping to programme learning outcomes (PO)	

	Subject learning outcomes	Programme learning outcomes								
		PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9
	Upon conclusion of this subject, the student should be able to:									
11.	Explain the different types of data structures and its usages in certain	✓	✓							
12.	Solve practical and complex problems where choice of data structures and algorithms are essential to implement effective programs with acceptable time and space requirements	✓	✓	✓				✓		
13.	Evaluate the computational efficiency of the principal algorithms for sorting, searching and hashing	✓	✓	✓				✓		
14.	Decide if a given problem can be given an efficient implementation	✓	✓	✓				✓		

17.	Transferable skills • An ability to use skills and techniques applicable in computing • Ability to derive appropriate algorithms for problem-solving • An ability to function effectively in a programming team
18.	Synopsis This course introduces the students to a study of stacks, trees, graphs, hash tables and priority queues, together with algorithms for traversing and manipulating these structures. Different implementations are covered, with emphasis being put on efficiency, program structure and abstract data types. Different sorting algorithms will be examined. Important elements in the course include recursive programming as well as evaluation and description of program efficiency.
19.	Mode of delivery (lecture, tutorial, workshop, seminar, etc.) Lectures, tutorials, practicals

Figure 6. Sample of the subject information document [31].

It is now time to define the restrictions in order to specify the boundaries for every object property. Such restrictions

describe the extent of the relationships that exist between and among classes. The Computing Ontology requires the use of *universal restrictions* that denote exclusive relationships. Members of a universal restriction do not participate in any other relationships. In Figure 9, the keyword *only* is used to define a universal restriction, while *or* indicates that every class participating in the restriction is unique and distinct.

Figure 7. Class hierarchy knowledge representation.

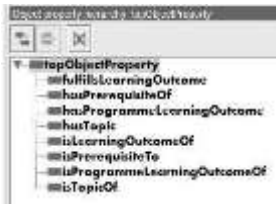

Figure 8. Object properties.

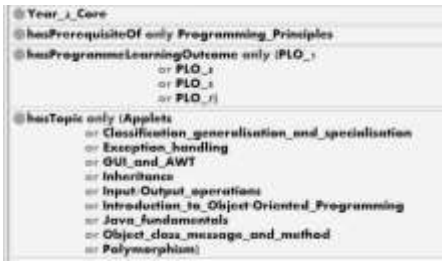

Figure 9. Universal restrictions illustrate prerequisites, learning outcomes and subject topics.

The Computing Ontology is now complete and requires to be processed by a reasoner. Protégé has a built-in reasoner, FACT++, that tests whether one class is a subclass of another to compute the inferred ontology class hierarchy. This ensures that all relationships are well-defined. After the ontology is processed by the reasoner and it is found to be well-defined, Ontograf would be initiated to visualize both asserted and inferred classification hierarchies (see Figure 10).

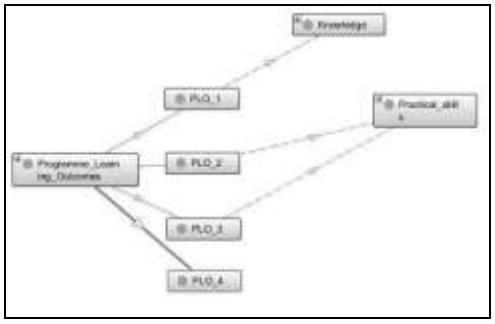

Figure 10. Ontograf visualization of the inferred class Programme_Learning_Outcomes.

3.2.2 Curriculum Wiki

The user interface design is presented in Figures 11-12.

Figure 11. Sample screenshot of a search initiated for "Software Engineering".

3.2.3 API

Recall that four typical curricula management operations that were identified from [4] are dealt with here. These include two operations involving user-initiated searches from keywords, one is a user request for a subject exemption and the last involves a user request for a change in a particular subject syllabus. All four require extraction of information from the Computing Ontology through the Curriculum Wiki.

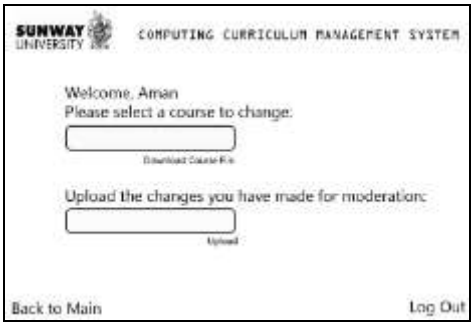

Figure 12. Sample screenshot of a change request.

The API is developed to fulfil these typical requirements. Note that the Computing Ontology is not accessed from within Protégé, but is accessed as an RDF document. This makes the ontology much easier to manipulate and to extract relationship information.

4 RESULTS

The Computing Ontology was processed by the FACT++ reasoner and all relationships and restrictions were confirmed to be properly defined and all classes consistent.

Recall from Figure 4 that Programme Learning Outcome 4 is mapped onto two of the MQF Learning Outcome Domains, *Social skills and responsibilities* and *Communication, leadership and team skills*. However, Figure 10 shows that PLO_4 has no mapping at all. Despite all efforts, Ontograf would not display more than one universal restriction. Communication with the Protégé/Ontograf developers in Stanford University [32] saw acknowledgement of this problem. At the time, they indicated a willingness to release a Protégé/Ontograf upgrade if warranted. The version used in [7] was the Beta release of the Protégé 4.1.

In [7], another obstacle was reported during the design of the Curriculum Wiki. Current sets of Application Programming Interfaces (APIs) for OWL and Web 3.0 do not allow the searching and the modifying of the ontology to be implemented. At the time of writing [7], the search API is limited to querying and reproducing the ontology in its entirety, and there are only APIs that can add and remove classes for the ontology. Full-featured

ontology editing APIs did not yet exist then. As a consequence, the Curriculum Wiki could not be linked to the Computing Ontology at the time [7].

Since then, the project has moved ahead with the development of an API in an attempt to address the manipulation and extraction issues involving the Computing Ontology.

The results have been somewhat encouraging, although it is still necessary to continue to improve the functionality of the API. Still in its infancy, the API can only carry out simplistic keyword searches, whereas the other two user requests for exemption and syllabus change have yet to be realized.

5 CONCLUSION

A Computing Curricula Management System (CCMS) was proposed as a design that utilizes a Computing Ontology as a knowledge source that interacts with a Curriculum Wiki facility through ontological agents. The design would then exploit agent-interaction models that had already been analyzed through the application of the Conceptualization and Analysis phases of the MAS-CommonKADS Agent Oriented Methodology [4]. The Design phase was utilized to develop the design of the CCMS.

Certain obstacles reported in an initial part of this project included difficulties in searching, manipulating and extracting ontological relationships from the Computing Ontology operating within the Protégé OWL Editor. There was also a visualization issue with the built-in Ontograf graphical visualization tool. As of the writing of this article, this is not yet resolved.

The API development has addressed two of the four typical curricula management operations identified in [4]. Work on the other two is on-going.

5.1 Contribution

The advantage of adopting the ontology approach is that it facilitates the sharing and re-use of knowledge without being task-specific. Different users can use the same ontology for different applications without re-defining existing relationships and properties. Furthermore, the Protégé editor was developed using the Web Ontology Language (OWL), which is endorsed by the World Wide Web Consortium (W3C). This makes the Computing Ontology ideal for use on the Semantic Web, as it has the potential to facilitate knowledge-sharing without boundaries [2].

The use of Protégé facilitates understanding owing to the highly intuitive nature of a graphical representation and user interface, an example of which is shown in Figure 10. This is because Ontograf is a sophisticated visualization tool. It enables searching of nodes and accurately displays object properties and inter-class relationships.

In addition, one could improve access and manipulation of this ontology outside the editor environment. If one

viewed it as a plain text file, one could access it as a regular RDF document.

As a comparison, database technology does not readily illustrate relationships among data entities. Another advantage of this approach over that of the database is that complex relationships between and among classes may not be so easily defined.

5.2 Limitation

An obstacle was uncovered during the Design phase [7]. It was found that Ontograf would not display more than one universal restriction (see Figure 10). Analysis of this problem by the Protégé/Ontograf developers could lead to a possible upgrade of the existing Protégé/Ontograf package [32]. However, at the time of this article, the latest release of Protégé 4.1 (build 239) still exhibits this limitation.

Furthermore, the API developed specifically to overcome the issue of ontology relationship information extraction and manipulation is still not complete and is not sufficiently mature to be used as a fully functional API. Therefore, the interface between the Computing Ontology and the Curriculum Wiki is still incomplete.

5.3 Future Work

It proposed that development of the API must continue to resolve the other two user requests of exemption and subject syllabus change. It is envisaged that this API should function as a Protégé plug-in. This API would add to the growing list of Protégé plug-ins in the Protégé Plugin Library [33].

In doing so, the combination of the back-end ontology with a customized full-feature API would form a practical alternative to the more conventional database approach.

6 ACKNOWLEDGMENTS

This project was funded by the Sunway University Internal Grant Scheme (SCT-0109-08).

7 REFERENCES

1. Gruber, T.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5, no. 2, 199--220 (1993).
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. ScientificAmerican.Com (2001), http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21
3. Cassel, L.N., Sloan, L.N., Davies, G., Topi, H., McGettrick, A.: The Computing Ontology Project: The Computing Education Application. In: Proc. of the 38th SIGCSE Technical Symposium on Computer Science Education, pp. 519--520. ACM, Covington Kentucky (2007).
4. Tang, A., Lee, A.: A Computing Curriculum Wiki – Analysis and Modeling using the MAS-CommonKADS Agent-Oriented Methodology. Sunway Academic Journal 7, 1--15 (2010).
5. Cassel, L.N., Davies, G., LeBlanc, R., Snyder, L., Topi, H.: Using a Computing Ontology as a Foundation for Curriculum Development. In: Proc. Sixth International Workshop on Ontologies & Semantic Web for E-Learning, Montreal Canada, pp. 21--29. (2008).
6. Iglesias, C.A., Garijo, M., Gonzalez , J.C., Velasco, J.R.: Analysis and Design of Multiagent Systems using MAS-

CommonKADS. In: Singh, M.P., Rao, A.S., Wooldridge, M. (eds.) ATAL-97. LNCS, vol. 1365, pp. 313--327. Springer, London (1998).
7. Tang, A., Abdur Rahman, A.: Proof-of-Concept Design of an Ontology-Based Computing Curricula Management System. In: Cherifi, H., Zain, J.M., El-Qawasmeh, E. (eds.) DICTAP2011. CCIS (LNCS) vol. 167, pp. 280--292. Springer, Heidelberg (2011).
8. W3C Recommendation, OWL Web Ontology Language Guide, http://www.w3.org/TR/owl-guide/
9. Ontology editor survey results, http://www.xml.com/2002/11/06/Ontology_Editor_Survey.html/
10. Medina, M.A., Sanchez, A., Castellanos, N.: Ontological Agents Model Based on MAS-CommonKADS Methodology. In: Proc. 14th International Conference on Electronics, Communications and Computers (CONIELECOMP 2004), pp. 260--263. Veracruz, Mexico (2004).
11. Luck, M., McBurney, P., Preist, C.: Agent Technology: Enabling Next Generation Computing: A Roadmap for Agent-Based Computing. Agentlink report (2003), http://www.agentlink.org/roadmap
12. CommonKADS: What is CommonKADS? (1995), http://www.commonkads.uva.nl/frameset-commonkads.html
13. Knowledge Web (2007), http://knowledgeweb.semanticweb.org/
14. Dicheva, D., Sosnovsky, S., Gavrilova, T., Brusilovsky, P.: Ontological Web Portal for Educational Ontologies. In: SWEL@AIED'05, AI-ED 2005. Amsterdam, The Netherlands (2005).
15. Cassel, L.N., Davies, G., LeBlanc, R., Snyder, L., Topi, H.: The Computing Ontology Project (2008), http://what.csc.villanova.edu/twiki/bin/view/Main/OntologyProject
16. Acharya, C.: Outcome-Based Education (OBE): A New Paradigm for Learning (2007), http://www.cdtl.nus.edu.sg/link/nov2003/obe.htm
17. Spady, W.: Outcome-based Education. Australian Curriculum Studies Association, Belconnen, ACT (1993).
18. Bagert, D.J., Hilburn, T.B., Hislop, G., Lutz, M., McCracken, M., Mengel, S.: Guidelines for Software Engineering Education Version 1.0 (CMU/SEI-99-TR-032). Technical report, Working Group Software Engineering Education and Training (WGSEET), Software Engineering Institute, Carnegie Mellon University (1999).
19. The Joint Task Force on Computing Curricula: Computer Science Curriculum 2008: An interim revision of CS 2001. Institute of Electrical & Electronics Engineers – Computer Science, Association for Computing Machinery (2008).
20. Association for Computing Machinery, Association for Information Systems and Association for Information Technology Professionals: Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems 1997 & 2002. Association for Information Systems (2002).
21. The Joint Task Force on Computing Curricula: Computing Curricula Software Engineering Volume. Institute of Electrical & Electronics Engineers – Computer Science, Association for Computing Machinery (2004).
22. The Joint Task Force for Computing Curricula: Computing Curricula 2005: The overview report. Association for Computing Machinery, Institute of Electrical & Electronics Engineers (2005).
23. The Joint Task Force on Computing Curricula: Computing Curricula Information Technology Volume. Association for Computing Machinery, Institute of Electrical & Electronics Engineers (2008).
24. Fernandez-Chung, R. M.: MQF in Programmes. Malaysian Qualifications Agency, Ministry of Higher Education, Malaysia (2008).
25. Malaysian Qualifications Agency: Malaysia Qualifications Framework. Ministry of Higher Education, Malaysia (2008).
26. Application programming interface, http://en.wikipedia.org/wiki/Application_programming_interface

27. Orenstein, D.: QuickStudy: Application Programming Interface (API). Computerworld (2000), http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleID=43487
28. Bloch, J.: How to Design a Good API and Why it Matters, http://lcsd05.cs.tamu.edu/slides/keynote.pdf
29. Protégé OWL Editor, Version 4.1 (build 239), Stanford Center for Biomedical Informatics Research (2010), http://protege.stanford.edu/download/protege/4.1/installanywhere/Web_Installers/
30. Studer, R., Benjamins, V.R., Fensel, D.: Knowledge Engineering: Principles and Methods. Data & Knowledge Engineering 25, 161--197 (1998).
31. School of Computer Technology: Application to Conduct Programme of Study to the Malaysian Qualifications Agency, Ministry of Higher Education, Malaysia, BSc (Hons) Computer Science. Sunway University, Petaling Jaya (2011).
32. Redmond, T.: Email correspondence, Protégé/Ontograf development team, Stanford University (2011).
33. Protégé Plugin Library, http://protegewiki.stanford.edu/wiki/Protege_Plugin_Library