International Journal of Innovative Computing, Information and Control
Volume **14**, Number **1**, February **2018**

ICIC International ©2018 ISSN 1349-4198
pp. **197–209**

AN OPEN ONTOLOGY REPOSITORY AT PRINCE SULTAN UNIVERSITY

Ahmed Sameh, Abdulla Al-Masri and Nahlah Al-Mahshouq

Department of Computer Science and Information Systems
Prince Sultan University
P.O. Box No. 66833 Rafha Street, Riyadh 11586, Saudi Arabia
asameh@psu.edu.sa

Received February 2017; revised June 2017

Abstract. *We are reporting on our experience in building and deploying OWL ontologies from within a newly introduced PSU ontology repository (POR) into a number of application software. These ontologies are also exposed to public outsiders to edit and extend (Open) and allow access to similar Internet ontologies using public ontology search engines such as SWOOGLE and IBM WATSON. This environment is mainly used to empower software application developers with semantics (analogues to Semantic Web) to limit development complexity, provide knowledge management, and organize software project information. OWL ontologies are machine readable; they can be incorporated into the software application development as accessible knowledge base shared resources which parts of the software can access as needed knowledge units via regular protégé programming interfaces: Script Tab, and Java API. These shared resources reduce development complexity, and promote re-use through linking similar and cross domains ontologies. We have been using and testing them within software applications to model certain artifacts (such as context and profile management, Internet of Things, sensor networks, cinema production, and risk and security management components) during the application software development life cycle. In this paper we report on our experience with each of these software applications: their artifacts ontology structure, representation and engineering, validation and verification, development and testing, and ontology querying and usage. We introduce a new framework of mixing "ontology engineering and access" with "application software development environment". In order to measure the gains achieved through this proposed framework, we ran a "Questionnaire" among developers to evaluate – What are the benefits gained of mixing "Ontology Engineering and Access" with "Application Software Development Environments"? – Is the "Framework" presented perceived useful by developers? – Are the "Consolidated Customized" ontologies constructed using the framework better, in some modeling quality sense, than the ontologies constructed without the framework? – Are the tasks given to developers done faster when using the framework? – How do developers use the framework provided, and what support would be beneficial? Subjective opinions show that 2/3 of developers perceive the framework as useful or very useful. We found out that developers feel that they have constructed better quality ontologies and that they are "guided" by the framework. As such the main purpose of the open POR to facilitate inhouse and out-of-house resources for building consolidated customized ontologies has been fulfilled.*

Keywords: Software development framework, Ontology engineering, Domain modeling, Knowledge management, Security management, Sensor networks, Software risk management

1. **Introduction.** At the moment PSU ontology repository (POR) is hosting a number of domain ontologies and is exposed to outsiders to edit and extend [1]. Nowadays one can find on the Internet many such "ontology repositories" such as NCBO BioPortal [2] of Biomedical ontologies, Cupboard [3] powered by IBM Watson search engine, TONES

[4], DERI [5], Knoodl [6], Ontology Design Patterns [7], SchemaPedia [8], and LOV [9] linked open vocabularies. These along with number of search engines such as "Ontologies Search Engines": Swoogle [10], and IBM Watson [11] are used regularly by several software developers. Almost all these ontologies are built, separately from the software application, using the "Protégé" editor in the famous "OWL" programing language. OWL has been chosen from among a number of alternative languages: Dublin Core, Resource Description Framework (RDF), and Learning Object Metadata (LOM) [12]. OWL was chosen for its high expressive power and its logical reasoning and inference capabilities in testing the produced ontologies.

POR's objective is to build a new "ontology repository" environment that is different from other previous repositories and/or at least consolidates, and complements them. The purpose of the proposed new repository environment is to provide "advice" and "assistance" to non-domain expert software developers to choose mechanisms fitting their needs and provide domain knowledge base (a kind of ontology lookup/guidance service). It is meant to expand the idea of STAC [12] from "security" domain (promotes re-use through linking similar and cross domains ontologies) to other domains. Stored ontologies are used to annotate resources with domain-related information. OWL ontologies are machine readable; they can be incorporated into the software application development as accessible knowledge base shared resources which parts of the software can access as needed knowledge units (using Protégé Programmable interfaces – Script Tab and Java API). These shared resources will reduce development complexity. SPARQL (Protégé add-on) then serves as an interface to software end users and developers to answer their domain context, technologies, and reasoning questions (inference). At the moment, the proposed repository also links to famous ontology research engines: Swoogle and IBM Watson to locate similar domain ontologies on the Internet, present them to the developers and promote knowledge re-use. We demonstrate the setup of POR: The POR's Hub that provides a gateway to various software domain ontologies: security, agriculture, weather, emotion, healthcare, security, tourism, etc. Ontology construction tools (basically Protégé and its many add-ons) are used by software developers to semi-automate ontology construction and customization As far as we know, this is the first framework that mixes ontology building and access with application software development life cycle. Also access through "Swoogle" and "Watson" provide possibilities of importing other "Internet" ontologies (as Web resources) that can be referred to or integrated into currently constructed consolidated ontologies. In fact, the building of these ontologies is based on the well-known "ontology engineering" methodology [4]. Ontology engineering is merged with the application software development life cycle. It consists of a set of iterative tasks of: defining terms (artifacts) in the software domain and relations among them; identifying concepts (classes), and arranging them in hierarchy (subclasses – superclasses); defining which attributes and properties (slots) classes can have constraints on their values, defining individuals and filling in property values. Ontologies can refer to each other through multiple relationships. All this is done within the application software development programming environment. SPARQL [15] (another Protégé plug-in) is then used for validation and verification of the built ontologies. It accesses the OWL ontology produced by Protégé editor. We validate for consistency, completeness, clarity, conciseness, generality, and instantiation of real cases. SPARQL also serves as the interface to end users/developers to answer their domain context, technologies, and reasoning questions. Ontology search engines Swoogle and Watson are used to locate and import similar domain Internet ontologies that are then imported and exposed to the same and/or other SPARQL queries so that they result in a rich extended up-to-date informed environment for end users and application developers. As such we emphasize implementing

knowledge re-use. The crust of this paper is that current software applications can be improved and/or gain more user/developer satisfaction only if they start incubating "ontologies" within their traditional/agile development strategies. The POR repository has been active since late 2015. The structure of this paper goes as follows. Sections 2-6 show samples from the POR repository ontologies/applications: Section 2 presents "enterprise risk management" ontology/application, Section 3 presents "Internet of Things – IoT" ontology/application, Section 4 presents "sensor network" ontology/application, Section 5 presents "cinema production" ontology/application, Section 6 presents "security management" ontology/application, Section 7 presents the evaluation of the proposed framework, and then we conclude in Section 8.

2. **Enterprise Risk Management Ontology/Application.** In this section we report on our experience with an enterprise risk management software application: its artifacts ontology structure, representation and engineering, validation and verification, development and testing, and ontology querying and usage. Our previous experience [15] in risk management in software development ontology has resulted in the consolidated software development risks ontology in Figure 1. Swoogle and Watson were able to locate similar ontologies with various details as in Figure 1. An application developer in this area made use of this rich environment. For example, a developer has identified various different threats, volanbilies, matigations, audits, plans, priority, assessment, analysis, control, SWOT, support, etc. Some of these are imported from Internet ontologies through the Swoogle and Watson ontology search engines and thus promote knowledge re-use (Figure 1). In enterprises, risks are organized in a hierarchical structure parallel to the organization structure. Inter-risks relationships have established as well as "Key Risk Indicators". Elements of such hierarchy are observed in Figure 1, where risk elements, their impact, indicators, and relationships to other risks are established. Software developer customizes his/her ontology according to his/her needs. Incorporate the OWL XML-like format of the ontology into the software code via Protégé's programmable interfaces: Script Tab, and Java API (see details in Section 7). This instant customized re-use made development

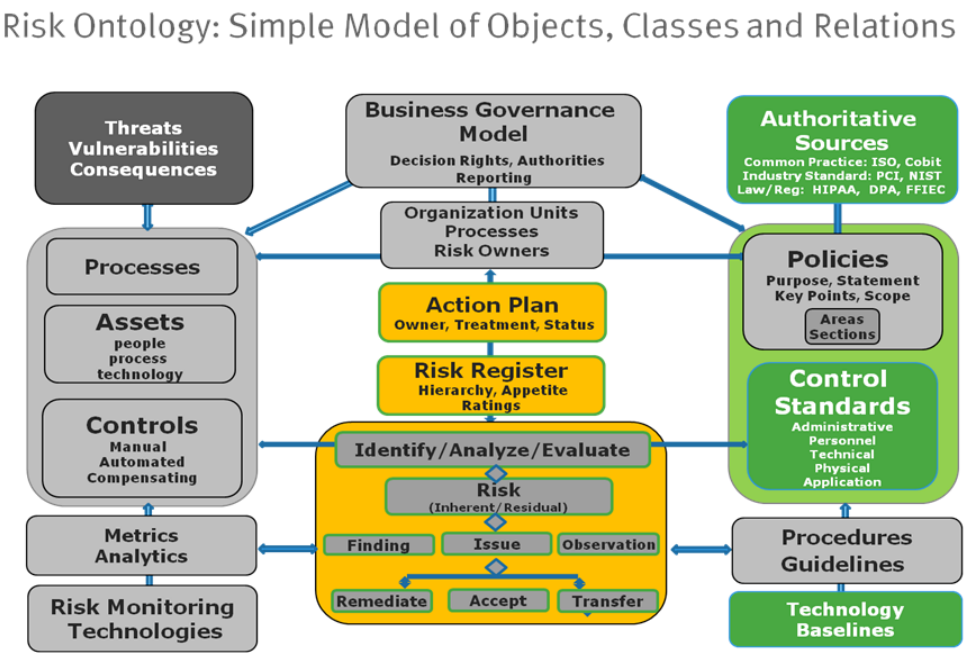

FIGURE 1. Risk ontology: Simple model of objects, classes, and relations

much easier and more organized. OWL ontologies are machine readable; they are incorporated into the application development as accessible knowledge base shared resources which parts of the software access (via Protégé Script and Java API) as needed knowledge units. These shared resources have reduced development complexity, and promoted re-use through linking similar and cross domains ontologies.

Figure 1 shows how the risk tool (Protégé) organizes its register, and action/mitigation plans within the organization's units, policies, procedures, assets, processes, authoritative sources, and guidelines. The context of risks is as important as its hierarchical structure. Based on the above ontology; an ontology-based risk software was developed for Android Mobile phones [15]. The software has incubated the above customized ontology of Figure 1. The OWL-XML ontology has made use of it to develop policies preventing excessive power consumption and promote power saving in these smart phones. The ontology allowed for mixing both preventive and detective approaches together. The ontology made the development, verification, and deployment of the proposed system much easier and more effective. Representation and engineering, validation and verification, development and testing, and querying and usage cases are few usage samples in the figure above. The ontology/application framework made the development, verification, and deployment of the mobile risk software module much easier and effective.

3. **Internet of Things (IoT) Ontology/Application.** In this section we report on our experience with an IoT software ontology/application: its artifacts ontology structure, representation and engineering, validation and verification, development and testing, and ontology querying and usage. Our previous experience [16] in IoT ontology has resulted in the consolidated ontology in Figure 2. Swoogle and Watson were able to locate similar ontologies as in Figure 2. An application developer in this area has made use of this rich environment and used it with the OpenIoT IDE development environment [6]. For

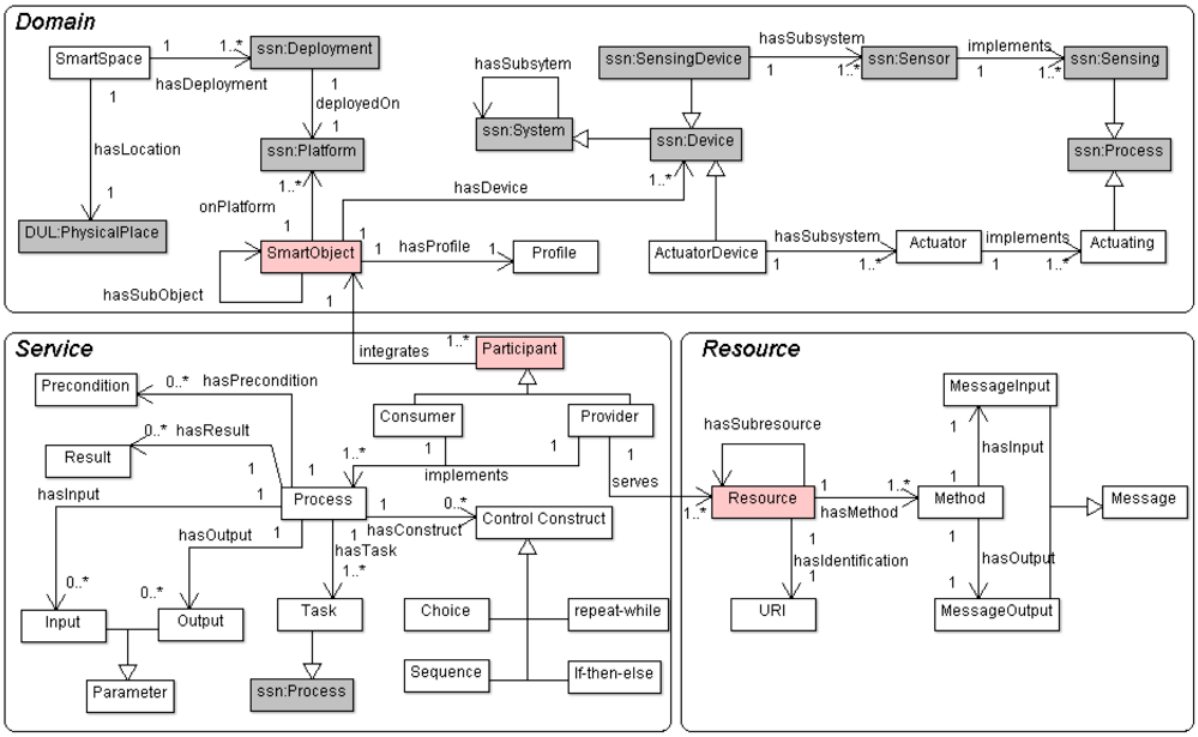

FIGURE 2. Customized consolidated IoT ontology/application

example, a developer has identified various different sensors, devices, aggregations, configuration, data sources, metadata, actuators, controllers, etc., from the ontologies. Some of these have been extended and imported from other Internet ontologies through the Swoogle and Watson ontology search engines that promote knowledge re-use. Ontologies for the IoT can integrate SSN ontology, OpenIoT ontology, Upper Merged SUMO IEEE ontology, Sematic Web for earth and environment terminology (SWEET), and SEEK extensible observation ontology. The OpenIoT ontology relies on W3C SSN ontology. OpenIoT has the following features: integrate sensors and things with the cloud server, configure, deploy and use IoT services, audit, assess privacy issues, semantic annotation capability, energy efficient sensors data harvesting, publish and subscribe for continuous processing and sensor data filtering, and quality of service issues. Heterogeneous sensory information produces 24/7. Layers of the IoT are: 1) functional layer; 2) information layer, and 3) physical layer. The three make the IoT ontology as shown in the figure. OWL ontologies are machine readable; they have been incorporated into the software application development as accessible knowledge base shared resources which parts of the software can access (through "Protégé Script Tab", and "Protégé Java API") as needed knowledge units. These shared resources have reduced development complexity, and promoted re-use through linking similar and cross domains ontologies.

Figure 2 shows the proposed customized consolidated ontology/application integrating the representation of the "domain", "resources", and "service". Smart objects interact through SSN platform and exchange data about the domain. Resources send messages to each other through well-developed methods. Participants in the IoT ontology have processes that allow them to participate in the operations of the system.

Figure 3 shows how the OpenIoT cloud service can be accessed from our customized consolidated IoT ontology/application. Sensing-as-a-service request goes through the OpenIoT cloud infrastructure (Middleware) to OpenIoT ontology to access "Sensor Cloud

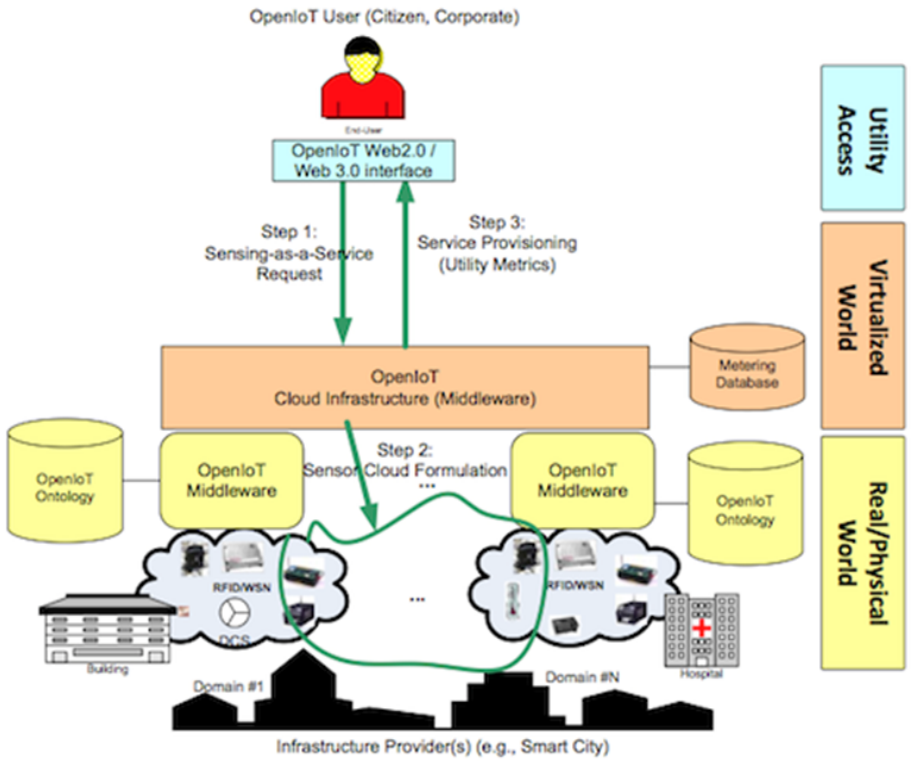

FIGURE 3. IoT ontology development with OpenIoT cloud

Formulations" in the real/physical world. Protégé accesses the OpenIoT cloud middleware through OpenIoT Web 3.0 interfaces to submit requests and receive service provisioning (metrics). Based on the above ontology, a "smart city" software application based on OpenIoT IDE development environment – ontology-based IoT application was developed for validation and verification of the built IoT ontology [16]. It accesses the OWL ontology produced by Protégé editor through Protégé interfaces: Script Tab and Java API. We validate for consistency, completeness, clarity, conciseness, generality, and instantiation of real cases. The software application has the goal of minimizing power consumption on smart city. Thus we emphasize implementing knowledge re-use.

4. **Sensor Network Ontology/Application.** In this section we report on our experience with a sensor network management software application: its artifacts ontology structure, representation and engineering, validation and verification, development and testing, and ontology querying and usage. Our previous experience [17] in sensor networks ontology has resulted in the consolidated ontology in Figure 4. The produced ontology is based on W3C SSN ontology and has a number of extensions. Swoogle and Watson were able to locate similar ontologies as in Figure 4. An application developer in this area has made use of this rich environment. For example, the developer has identified various different components of his/her customized ontology. Some of these have been imported from Internet ontologies through the Swoogle and Watson ontology search engines and thus promote knowledge re-use. The context ontology was added to W3C SSN to enhance its context awareness capabilities. OWL ontologies are machine readable; they are incorporated into the application development as accessible knowledge base shared resources which parts of the software can access as needed knowledge units (see details of Protégé programmable interfaces – Protégé Script Tab, and Protégé Java API in Section 7). These shared resources have reduced development complexity, and promoted re-use through linking similar and cross domains ontologies.

Figure 4 shows a high-level detail of the customized consolidated SSN ontology hierarchy. All features of interest are recorded in the appropriate places. This SSN ontology can be referenced in the IoT ontology presented in the previous section as well. Based on the above ontology, a customized consolidated ontology-based SSN was developed for Android Mobile phones [17]. The software has incubated the above customized ontology

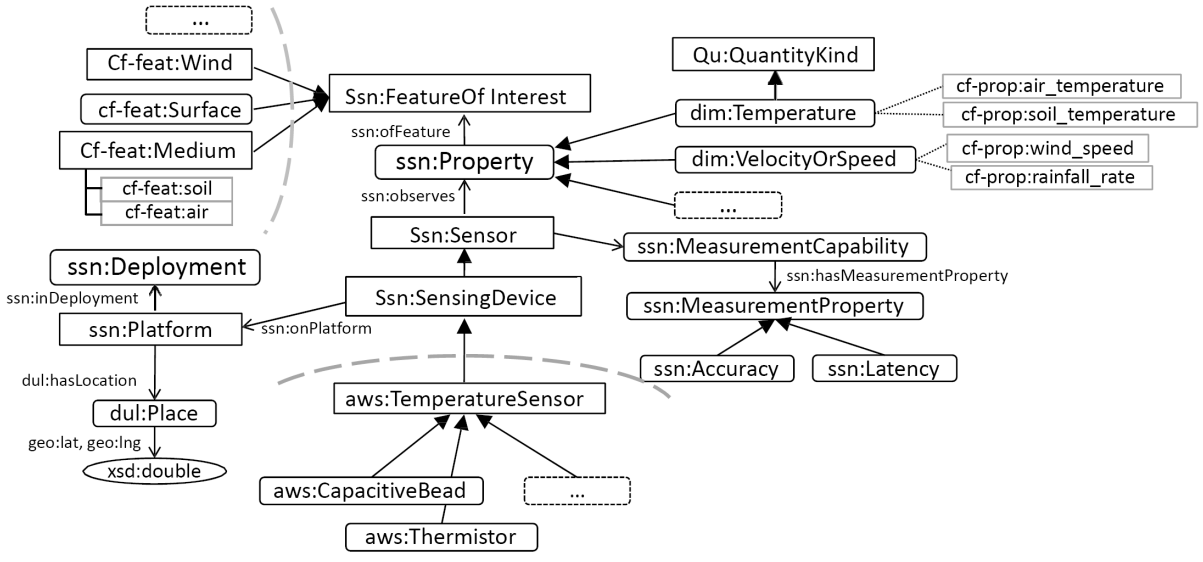

FIGURE 4. Customized OpenIoT and SSN combined ontology/application [5]

and made use of it to develop policies for controlling the various IoT components. The ontology allowed for mixing both preventive and detective approaches to limit the power consumption of the IoT components. The ontology made the development, verification, and deployment of the proposed system much easier and more effective. Representation and engineering, validation and verification, development and testing, and querying and usage activities are demonstrated using the Protégé programming interfaces: "Protégé Script Tab", and "Protégé Java API". SPARQL [15] is also used for validation and verification of the built ontologies. It accesses the OWL ontology produced by Protégé editor. We validate for consistency, completeness, clarity, conciseness, generality, and instantiation of real cases. Thus we emphasize implementing knowledge re-use.

5. **Cinema Production Ontology/Application.** In this section we report on our experience with a Cinema production management software application: its artifacts ontology structure, representation and engineering, validation and verification, development and testing, and ontology querying and usage. Our previous experience [18] in cinema production ontology has resulted in the consolidated ontology in Figure 5. Swoogle and Watson were able to locate similar ontology in Figure 5. An application developer in this area has made use of this rich environment through "Protégé Script Tab", and "Protégé Java API" programming interfaces. For example, a developer has identified various different subjects in the film industry such as actors, producers, theaters, titles, post office, records, and Oscare. Some of these are imported from Internet ontologies through the Swoogle and Watson ontology search engines and thus promote knowledge re-use.

The cinema ontology provides a controlled vocabulary to semantically describe and specify everything related to cinema industry. The cinema ontology is built using the Protégé editor with the famous OWL programing language and its plug-ins: Protégé

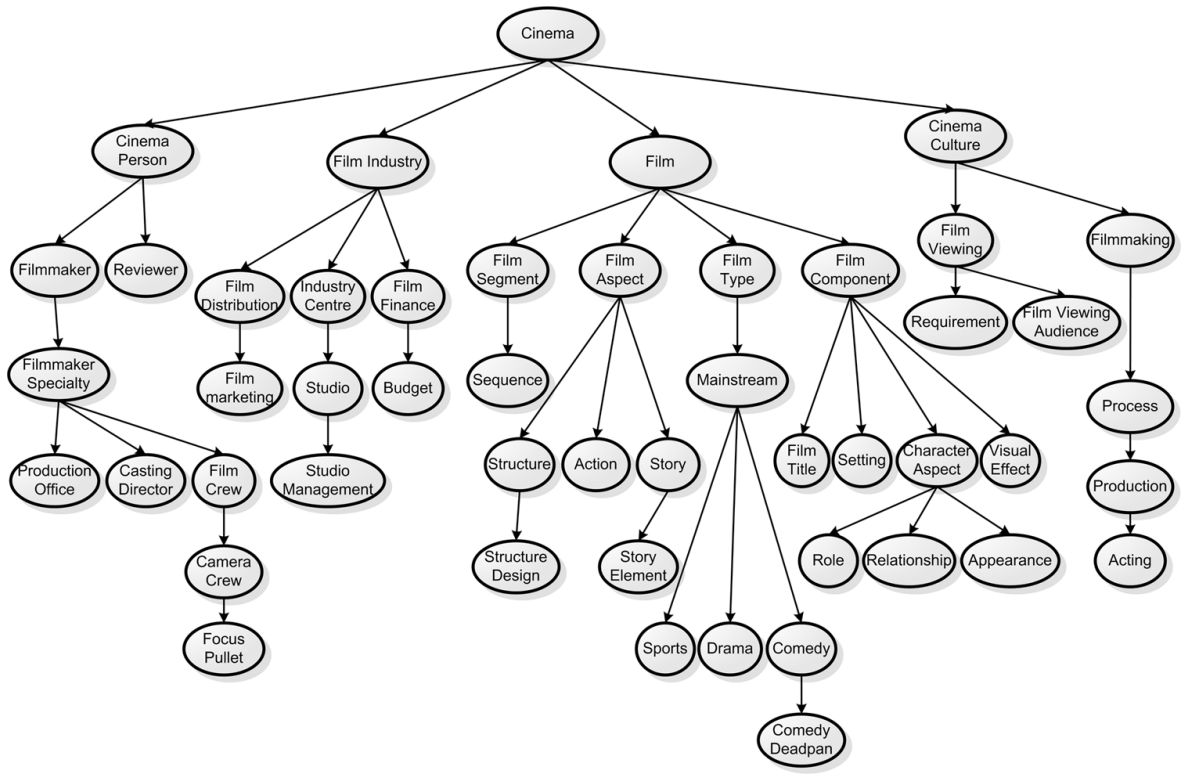

FIGURE 5. Customized consolidated cinema ontology

Script Tab, and Java API. The building of this ontology is based on the well-known "ontology engineering" methodology [9]. Figure 5 shows the cinema ontology structure that is based on the famous model [18]. We went through an iterative process of: scope/re-scope, consider reuse, enumerate terms, defining properties, add constraints, create instances, etc., to build such ontology. Figure 5 shows Protégé screen shot. It also shows the tree structure of the Protégé editor of the "cinema" ontology. OWL ontologies are machine readable; they are incorporated into the application development as accessible knowledge base shared resources which parts of the software can access as needed knowledge units (using both "Protégé Script Tab", and "Protégé Java API"). These shared resources have reduced development complexity, and promoted re-use through linking similar and cross domains ontologies.

Figure 5 shows the tree structure of the Protégé editor of the customized consolidated "cinema" ontology. Protégé uses tree structures to organize knowledge pieces inside the OWL ontology. Based on the above ontology, an ontology-based movie reviewer software App was developed for Android Mobile phones [13,18]. The software has incubated the above customized ontology and made use of it to develop a movies search engine App for the mobile phone. The ontology made the development, verification, and deployment of the proposed system much easier and more effective. SPARQL [15] is used for validation and verification of the built ontologies. It accesses the OWL ontology produced by Protégé editor through both the "Protégé Script Tab", and the "Protégé Java API". We validate for consistency, completeness, clarity, conciseness, generality, and instantiation of real cases. SPARQL also serves as the interface to end users/developers to answer their domain context, technologies, and reasoning questions. Ontology search engines Swoogle and Watson are used to locate and import similar domain Internet ontologies that are then imported and exposed to the same and/or other SPARQL queries so that they result in a rich extended up-to-date informed environment for end users and application developers.

6. **Security Management Ontology/Application.** In this section we report on our experience with a security management software application: its artifacts ontology structure, ontology representation and engineering, validation and verification, development and testing, and ontology querying and usage. Our previous experience [19] in software security services ontology has resulted in the consolidated customized ontology in Figure 6. Swoogle and Watson were able to locate similar ontologies as in Figure 6. An application developer in this area has made use of this rich ontology environment. For example, the developer has identified various different security related attributes for building the customized ontology. Some of these have been imported from Internet ontologies through the Swoogle and Watson ontology search engines and thus promote knowledge re-use. These ontologies have been accessable to the application developers through two Protégé progamming interfaces: "Protégé Script Tab", and "Protégé Java API". The security ontology is built using Protégé editor with the famous OWL programing language. The building of this ontology is based on the well-known "ontology engineering" methodology [11]. Figure 6 shows the security ontology structure that is based on the famous STAC model [19]. We went through an iterative process of: scope/re-scope, consider reuse, enumerate terms, defining properties, add constraints, create instances, etc. For example, the proposed model has been used to describe various cryptographic algorithms under the class: security algorithms. OWL ontologies are machine readable; they have been incorporated into the application development as accessible knowledge base shared resources which parts of the software can access as needed knowledge units (through "Protégé Script Tab", and "Protégé Java API"). These shared resources have reduced

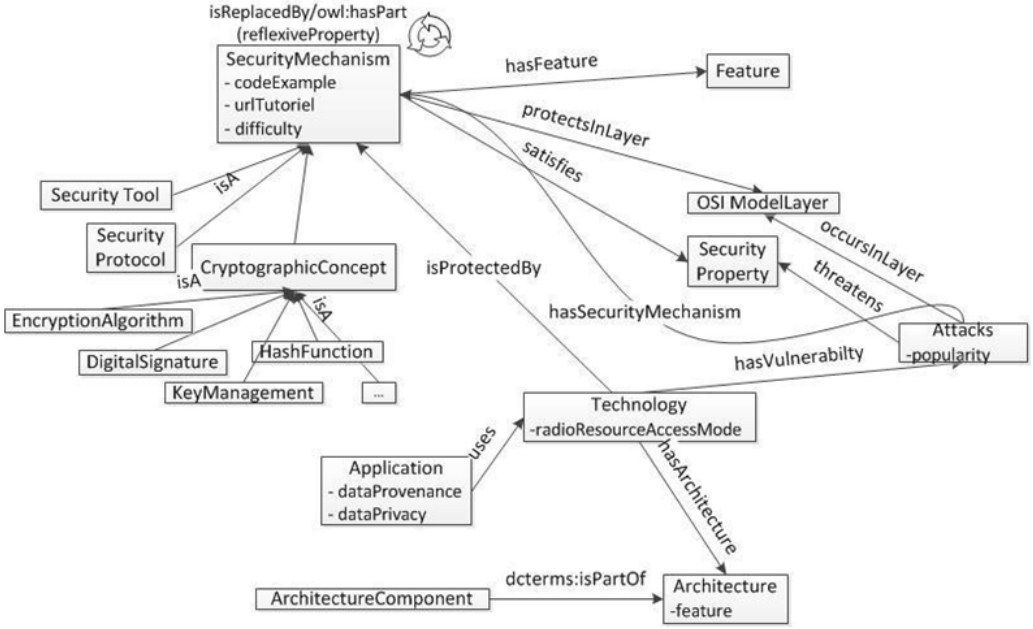

FIGURE 6. STAC ontology [7]

development complexity, and promoted re-use through linking similar and cross domains ontologies.

Figure 6 shows the STAC [19] ontology hierarchy: security mechanisms, cryptographic algorithms, tools, key management, digital signature, security protocols, properties, attacks, etc., are all included in this ontology. STAC has been around since 2013, and it is popular because it covers all the 7 layers of the standard security/network model as shown at the side of the figure. Figure 6 shows the security toolbox: Attacks & Countermeasures (STAC) components [19] for securing an IoT software application (e.g., Wi-Fi technology). Output is: getting information about the attacks and security mechanisms in place (e.g., Jamming). As the figure shows, the purpose of the ontology is to help non-security experts to secure their software applications with less effort and complexity. Based on the above ontology, an ontology-based IoT protection software application was developed for Android Mobile phones [19]. The software has incubated the above customized ontology and made use of it to develop protections for the IoT artifacts. The ontology allowed for mixing both preventive and detective approaches together. The ontology made the development, verification, and deployment of the proposed system much easier and more effective. SPARQL [11] is used for validation and verification of the built ontologies. It accesses the OWL ontology produced by Protégé editor through both the "Protégé Script Tab", and the "Protégé Java API". We validate for consistency, completeness, clarity, conciseness, generality, and instantiation of real cases.

7. **Experimentations to Measure Enhancements in Ontology/Application Developments.** Among the many Plug-ins available for Protégé, provided are two programming interfaces that developers can use to have direct programmatic access to the ontology content (see Figures 7 and 8). The "Protégé Script Tab" provides a scripting environment in several interpreted script languages such as Python, Pearl, and Ruby. The script commands are applied directly to the ontology currently loaded into Protégé. In addition to the Script Tab, Protégé provides a "Java API" that developers can use to access and programmatically manipulate Protégé ontologies (Figure 8). Developers can access this Protégé Java API from their Java application programs by calling the

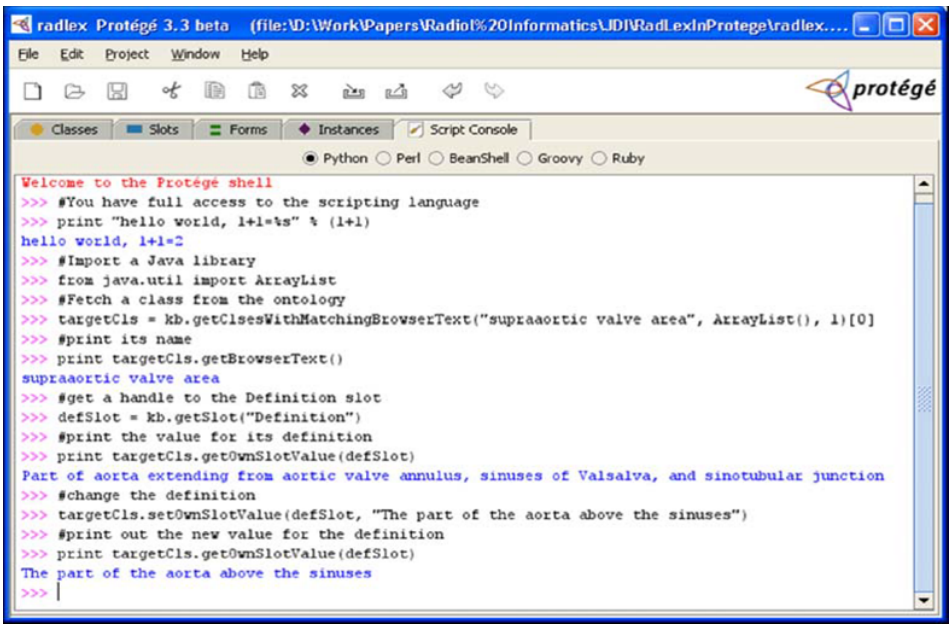

FIGURE 7. Protégé provides two programming interfaces that developers can have direct programmatic access to the ontology content – This figure shows programmatically interacting with ontologies using "Protégé Script tab".

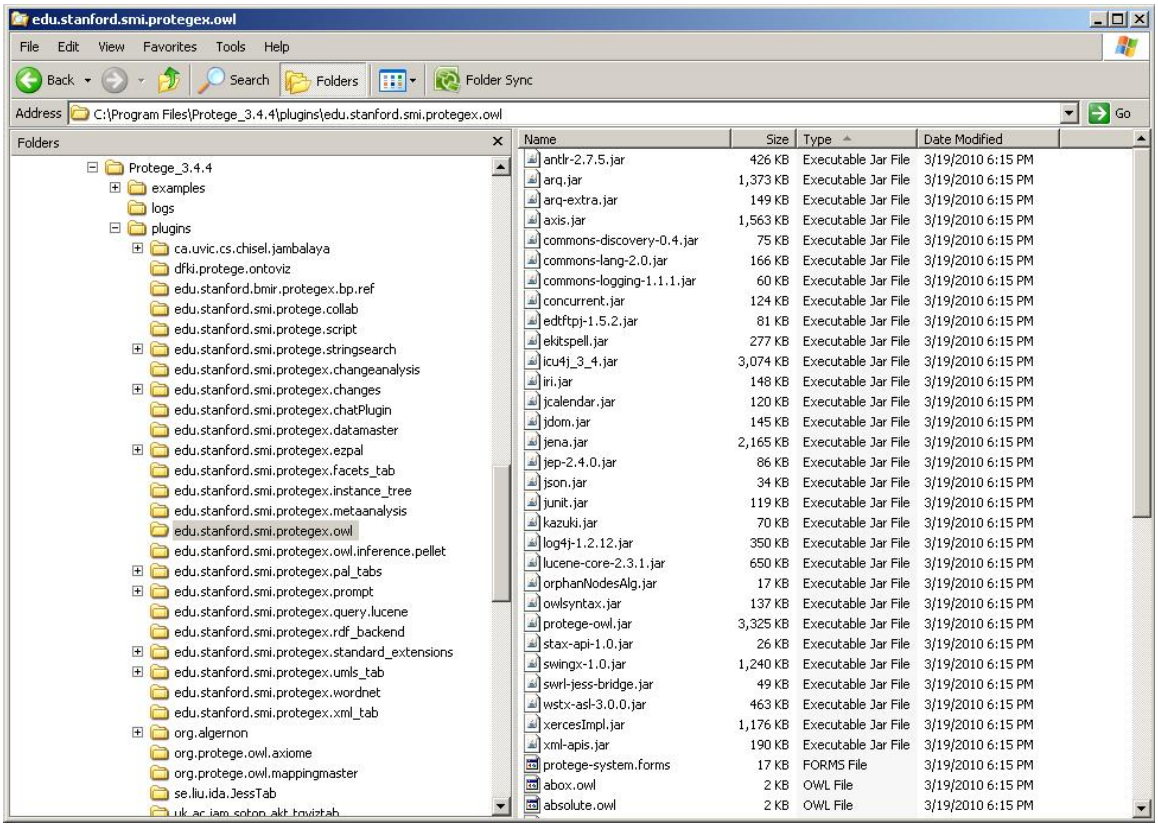

FIGURE 8. After a successful installation of the "Protégé-OWL" API programmer Java Library, the API provides classes and methods to load and save OWL ontology files, to query and manipulate OWL data models, and to perform reasoning based on description logic engines – all from within the Java programming environment.

appropriate methods to access the ontology in Protégé and perform the desired changes. As such software developers have the ontologies accessible and editable from within their programming environments.

Through the above programming interfaces to the ontologies from within the programming environment, we empower software application developers with semantics to limit development complexity, provide knowledge management, and organize software project information. In all the application software described above (Sections 2-6), developers have practiced the use of the above programming interfaces and programmatically accessed and edited the ontology contents from within their Java development environments. Empirical evidence needs to be gathered about how useful these capabilities are. We have identified a group of developers using this mix framework and another group not using the framework. We want to study their behavior, and study the process of using the ontologies in developing the resulting software. As an experimental setup, we have designed a "Questionnaire" based on the questions stated below. The first group of developers is using the suggested mix framework. The second group of developers is not using the framework. Total of 45 participants were randomly chosen on a survey done on two stages (tasks 1, and 2). Developers participated in the development of the applications cited in Sections 2-6 above were the target of the selection. Each developer filled out the "Questionnaire" recording his/her background, previous knowledge, working hours in the software application, and questions related to the above queries. In order to measure the gains achieved through these facilities, we ran the following "Questionnaire" experiment based on the following experimental questions.

1) What are the benefits gained of mixing "Ontology Engineering and Access" with "Application Software Development Environments"? 2) Is the "Mix Framework" presented in this paper perceived useful by developers? 3) Are the "Consolidated Customized" Ontologies constructed using the mix framework better, in some modeling quality sense, than the ontologies constructed without the framework? 4) Are the tasks given to developers done faster when using the mix framework? 5) How do developers use the mix framework provided, and what support would be beneficial?

Evaluation of the results and analysis of the "Questionnaire" responses are shown in Figures 9 and 10. Both figures show valuation of the constructed customized consolidated

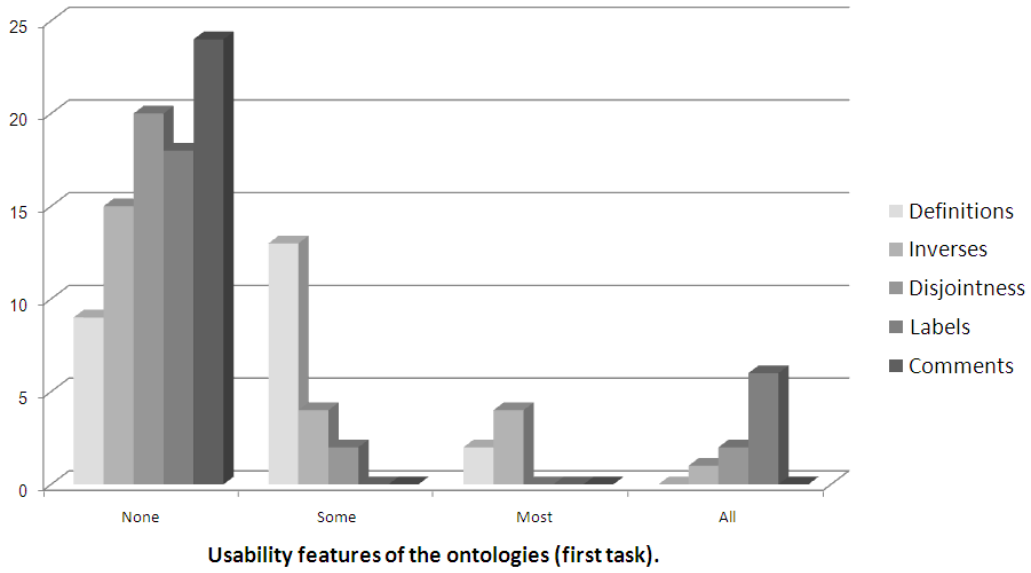

FIGURE 9. Result of the first task questionnaire for evaluating the proposed MIX framework

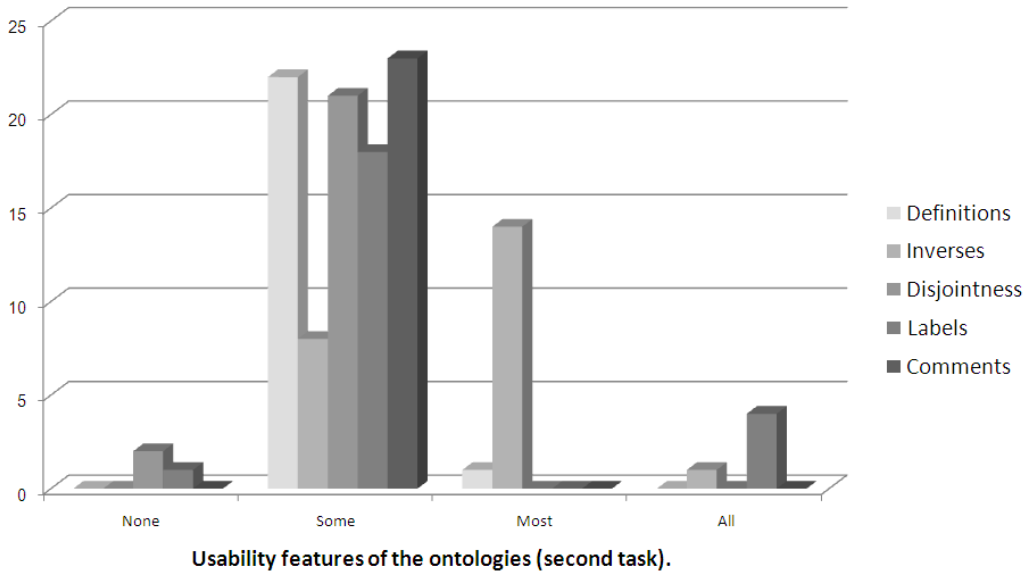

FIGURE 10. Result of the second task questionnaire

ontologies (mainly functional and usability evaluations). Coverage of the application needs. As for the "usability" measurement, we measure issues such as: presence of a naming convention, labels, comments, inverse relations, disjoint, level of axioms, modeling mistakes, etc. Figures 9 and 10 show the results of the surveys.

The following can be observed from the figures: -Is the framework useful? Subjective opinions: 2/3 of developers perceive the framework as useful or very useful. -Only 11% (5 developers) feel the framework was not useful. -Developers feel that they have constructed better quality ontologies and that they are "guided" by the framework. -Training on the framework is needed. -Faster development uses the framework. -Perceived as reducing routine work. -Increased discussion activity- pointing at new aspects. -Learning curve is high. The overall conclusion: -Suggested "Mix Framework" is perceived as useful, and -It increased "perceived quality" and "objective quality".

8. **Conclusions.** The crust of this paper is that current software applications can be improved and/or gain more user/developer satisfaction only if they start incubating "ontologies" within their traditional/agile development strategies. OWL ontologies are machine readable; they can be incorporated (practically via "Protégé Script Tab", or "Protégé Java API") into the software application development as accessible knowledge base shared resources. An application developer can then make use of this rich environment in incubating ontologies into software development life cycle. In order to measure the gains achieved through this proposed mix framework, we ran a "Questionnaire" to evaluate -What are the benefits gained of mixing "Ontology Engineering and Access" with "Application Software Development Environments"? Subjective opinions stated that: 2/3 of developers perceive the framework as useful or very useful. We found out that: -Only 11% (5 developers) feel the framework was not useful. -Developers feel that they have constructed better quality ontologies and that they are "guided" by the framework. The overall conclusion of the questionnaire: -Suggested "Framework" is perceived as useful. As such the main purpose of the open POR to facilitate inhouse and out-of-house resources for building during software development consolidated customized ontologies has been fulfilled. Future work will involve adding more ontologies to the repository and building a search engine for it.

REFERENCES

[1] *URL of the PSU Ontology Repository (POR)*, https://lms.psu.edu.sa/login/index.php.

[2] W. Yuan and K. Nahrstedt, Energy-efficient soft real-time CPU scheduling for mobile multimedia systems, *Proc. of the 19th ACM Symposium on Operating Systems Principles*, New York, 2003.

[3] Y. Yakas, Design and evaluation of a cross-layer adaptation framework for mobile multimedia systems, *SPIE/ACM Multimedia Computing and Networking Conference (MMCN)*, 2003.

[4] R. Vehvilainen, What is preventive software maintenance?, *Conference on Software Maintenance and Reengineering (CSMR)*, San Jose, CA, 2000.

[5] K. Schmid, A comprehensive product line scoping approach and its validation, *International Conference on Software Engineering*, Orlando, FL, 2002.

[6] R. N. Sulgrove, Scoping software projects, *At&T Technical Journal*, 1996.

[7] Z. Feng, W. Li and X. Li, *Ontologies Engineering and Its Application*, Tsinghua University Press, Beijing, 2007.

[8] B. A. Kitchenham et al., Towards an ontology of software maintenance, *Journal of Software Maintenance: Research and Practice Archive*, vol.11, no.6, pp.365-389, 1999.

[9] A. Vizcaíno, N. Anquetil, K. Oliveira, F. Ruiz and M. Piattini, Merging software maintenance ontologies: Our experience, *Conference on Software Maintenance and Reengineering (CSMR)*, Madrid, 2005.

[10] M. G. B. Dias, N. Anquetil and K. M. de Oliveira, Organizing the knowledge used in software maintenance, *Journal of Universal Computer Science*, vol.9, no.7, pp.641-658, 2003.

[11] A. Alain, H. H. Jane, A. Alain and D. Reiner, Software maintenance maturity model (SM^{mm}): The software maintenance process model, *Conference on Software Maintenance and Reengineering (CSMR)*, 2004.

[12] G. A. Junio, M. N. Malta, M. H. de Almeida, H. T. Marques-Neto and M. T. Valente, On the benefits of planning and grouping software maintenance requests, *Conference on Software Maintenance and Reengineering (CSMR)*, 2011.

[13] R. Francisco and V. Aurora, An ontology for the management of software maintenance projects, *International Journal of Software Engineering and Knowledge Engineering*, vol.14, no.3, 2003.

[14] K. Diana and V. Olegas, *Survey on Ontology Languages*, G. Jains and K. Mariti (eds.), Springer Berlin Heidelberg, 2011.

[15] A. Sameh, F. Khan and N. El-Hakim, Overview of the E-accreditation project at Prince Sultan University, *Lebanese American University Assessment and Planning in Higher Education*, Byblos, Lebanon, 2015.

[16] A. Sameh, F. Khan and N. El-Hakim, An agile quality assessment process for program-level accreditation, *The 10th Annual Conference in Smart Learning, in Innovation Arabia*, Dubai Marina, 2017.

[17] A. Sameh and A. Al-Masri, Smartphone preventive customized power saving modes, *International Journal of UbiComp*, no.1, 2017.

[18] A. Sameh and N. Al-Mashouq, Ontologies for software maintenance with extension on change requests scoping, *The 10th Annual Conference in Smart Learning, in Innovation Arabia*, Dubai Marina, 2017.

[19] A. Sameh, Social networks role in political elections, *The Journal of the Japanese Society for Artificial Intelligence*, vol.2, 2013.