An Ontology-Based Context Model for Managing Security Knowledge in Software Development

Shao-Fang Wen, Basel Katt
Norwegian University of Science and Technology
Gjøvik, Norway
shao-fang.wen, basel.katt @ntnu.no

***Abstract*—Software security has been the focus of the security community and practitioners over the past decades. Much security information is widely available in books, open literature or on the internet. We argue that the generated huge mass of information has resulted in a form of information overload to software engineers who usually finish reading it without being able to apply those principles clearly to their own application context. Our research tackles software security issues from a knowledge management perspective. In this paper, we present an ontology approach to model the knowledge of software security in a context-sensitive manner, supporting software engineers and learners to enable the correlation process between security domain knowledge and their working context. We also propose a web-based application for security knowledge sharing and learning where the ontology is adopted as the central knowledge repository.**

I. INTRODUCTION

Software security has been the subject of plethora studies for at least 40 years, and a steady stream of innovations has improved software engineers' ability to develop secure software and to protect applications. Improving software security requires that software engineers have the knowledge and skills to secure software development lifecycle (SDLC) such that they can resist attacks and handle security errors appropriately [1]. There are security committees or groups of experts that identify vulnerability patterns (e.g., CVE and CWE maintained by the MITRE Cooperation) and standards and guidelines generally applicable to secure software development (e.g., SEI CERT and OWASP) in order to leverage the experience and knowledge of many software development organizations.

Although much security information is widely available in the form of checklists, standards, and best practices in books, open literature or on the Internet, it remains difficult for software engineers to extract relevant pieces of knowledge to apply to their application-specific decision-making situations during SDLC. We argue that the huge mass of information has resulted in a form of information overload to software engineers who usually finish studying it without being able to apply those principles clearly to their own working context. Even if educated in software security knowledge before joining software development, these software workers still fail to associate the general security knowledge with their developing applications and fix vulnerabilities in the code when given a chance. It especially relates to what is known as a knowledge gap between knowledge available and knowledge required to build secure systems in the context of the application development.

Software development not only requires knowledge about its own domain but also about the domain for which software is being developed [2]. Each software product and process is different in terms of goals and contexts. To develop software for Mars landing is not the same as to develop software for a mobile phone. Software developers are often exposed to this diversity, which makes the software discipline inherently experimental [3, 4]. To mitigate this knowledge gap, we suggest that there is a need to manage security knowledge in a coherent way of describing it in a universal way and incorporating enough details to make the description more useful. Context is a way of giving knowledge focus and meaning [5].

Our motivation in this paper is to enhance software security knowledge management by modeling the knowledge in a context-sensitive manner where the software security knowledge can be retrieved taking the context of the application in hand into consideration. Ontologies make it possible to give this kind of purpose [6]. The ontological representation not only supports the integration of knowledge resources at various abstraction and semantic levels, but it can also be used by knowledge sharing services such as knowledge integration and interoperability, advanced knowledge search, knowledge visualization and therefore support the sharing and learning process about software security. In this paper, we proposed an ontology-based context-sensitive model to unify the concepts and terminology of security knowledge that can be adapted to the various context of the software development.

The rest of the paper is structured as follows: In section 2, we introduce the basic concepts needed in this paper. Section 3 presents the related works on ontology approaches in the software security domain. Our design of the ontology-based context model is explained in section 4. Section 5 describes the implementation of the ontology model and the evaluation result. Section 6 presents our proposed web application adopting the ontology. Lastly, conclusions and future works are presented in section 7.

II. BACKGROUND

A. Security Knowledge Management in Software Development

Software security is more than just security features. Security features, such as password encryption and SSL (Secure Socket Layer) between the web server and a browser, are functions of an application to prevent malicious attacks. Security is an emergent, system-wide property of a software system, which means that one cannot presume to achieve a high

ISSN 2305-7254

level of security by simply introducing security-related features into the software [7], [8]. This is because most security problems arise from bugs and flaws during the development process [9- 11]. Software security aims to avoid security errors in software by considering security aspects throughout the software development lifecycle. To train software engineers on critical software security issues, security knowledge should be spread in an effective manner.

Knowledge management has been defined as "the capability by which communities capture the knowledge that is critical to their success, constantly improve it, and make it available in the most effective manner to those who need it" [12]. Managing knowledge in software development is crucial to allow developers to capture, locate and share knowledge of code and methods throughout the project to maintain a competitive advantage. In order to increase the development staff's security knowledge, software project management needs to employ knowledge management mechanisms in encapsulating and spreading the emerging security discipline more efficiently in the software development process. As the software lifecycle unfolds, security-related knowledge could be directly applied to a knowledge-intensive best practice that can support software engineers to prevent, spot and mitigate these security errors.

B. Context and Knowledge Management

According to Brézillon [13], ''context is a set of information used to characterize a situation in which human and computational agents interact". The context has the capacity to provide a major meaning to knowledge, promoting a more effective comprehension about a determined situation in the collaborative work [14]. Contextual information is a crucial component of fully understanding knowledge [15-17]. Without proper contextual information, knowledge can be isolated from other relevant knowledge resulting in limited or distorted understanding [18], [19]. In knowledge management, the context has been considered as a relevant concept. Any architecture of knowledge management should include the design of knowledge items as well as the design of the overall contextual elements of the knowledge and what is the relationship among them [20].

Subsequently, researchers of psychology and education indicate that when knowledge is learned in a context similar to that in which the skills will actually be needed, the application of learning to the new context may be more likely [21], [22]. In software development, studying from a context and then abstracting the knowledge gained to be able to use it in a new context is a common way of learning programming that has been observed extensively in both new and experienced programmers [23], [24]. In order to capture and use knowledge appropriately, it is necessary to specify which context information is to be handled in the organization, and then represent this in a format that is understandable and acceptable to the individuals. In this situation, knowledge tools should be equipped with context-carrying functions so that it can effectively disseminate information to spread the application domain and other specific knowledge more evenly across the organization [25].

In the setting of software development, knowledge can be both dynamic and situation specific, and the complexity of knowledge usually exceeds the capacity of individuals to solve problems by themselves [26]. With the growth of complexity in the development of software projects, it is hard for software developers to master the expertise required to cope with the variety of concepts, frameworks, and libraries that are involved with software projects. Proper context helps identify appropriate levels of security, improves the precision of security decisions and makes security information more meaningful for the software development. When developers deal with the security errors within the context they are already familiar with, the consequences of exploiting the code's vulnerabilities will be understood with a strong and personal effect, which become more real and less theoretical.

C. Ontology

According to Gruber [27], an ontology is "an explicit and formal specification of a conceptualization", that is, a formal description of the relevant concepts and relationships in an area of interest, simplifying and abstracting the view of the world for some purpose [28]. Ontology facilitates capture and construction of domain knowledge and enables representation of skeletal knowledge to facilitate the integration of knowledge bases irrespective of the heterogeneity of knowledge sources [29]. Ontologies are now central to many applications such as scientific knowledge portals, information management, and integration systems, electronic commerce and web services. The main areas, in which ontological modeling is applied, include communication and knowledge sharing, logic inference and reasoning, and knowledge base.

With analyzing and extending several types of research [27, 30-33], we can identify and summarize the reasons for and benefits of developing and using ontologies in knowledge modeling.

- Ontologies share a common understanding of the structured information among people or software agents.
- Ontologies make domain knowledge reusable.
- Ontologies enable the interoperability among models or specific domain vocabularies.
- Ontologies allow and simplify the communication among humans, computational systems, and between humans and systems.
- Ontologies have the expressive power for acquiring context from diverse and heterogeneous source.
- It is possible to apply reasoning and inference mechanisms by means of explicit representation of semantics.

III. Related Work

There have been a number of papers published in the area of ontology modeling and applying semantic technologies to software security. Some effort focused on building security ontology to model the security requirements. Salini and Kanmani [34] present an ontology of security requirements for web applications, including concepts of asset, vulnerabilities, threats, and stakeholders. Their work aims at enabling the reuse of knowledge about security requirements in the development

of different web applications. Buch and Wirsing [35] present the SecWAO ontology with a focus on a secure web application, which aims to support web developers when specifying security requirements or making design decisions. It distinguishes concepts (classes) between methods, notations, tools, categories, assets, security properties, vulnerabilities, and threats.

Some research works present their ontology to support security design and risk assessment. Gyrard et al. [36] present the STACK ontology (Security Toolbox: Attacks & Countermeasures) to aid developers in the design of secure applications. STACK defines security concepts such as attacks, countermeasures, security properties, and their relationships. Countermeasures can be cryptographic concepts (encryption algorithm, key management, digital signature, and hash function), security tools, or security protocols. Kang and Liang [37] present a security ontology with the Model Driven Architecture (MDA) approach for the use in the software development process. The proposed ontology shows that the proposed security ontology can be used in modeling and designing security issues and concepts in each phase of the development process with MDA. Marques and Ralha [38] propose an ontology, which is related to the risk management aspect of web-based system development. The model is mainly employed in the design phase of the system development.

Finally, there are some papers focusing on using an ontology to model vulnerabilities and security attacks. Guo and Wang [39] present an ontology-based approach to model security vulnerabilities listed in Common Vulnerabilities and Exposures (CVE). The authors captured important concepts for describing vulnerabilities in the context of software security, providing machine-understandable CVE vulnerability knowledge and reusable security vulnerabilities interoperability. Khairkar et al. [40] present an ontology to detect attacks on web systems. The authors use semantic web concepts and ontologies to analyze security logs to identify potential security issues. This work aims to extract semantic relationships between attacks and intrusions in an Intrusion Detection System (IDS). Razzaq et al. [41] propose an ontology of attacks and an ontology of communication protocols, which provide a construct to improve the detection capability of application-level attacks in web application security. The authors employ the use of semantics in application layer security contrary to tradition signature-based approaches.

Our approach differentiates from the previous work in the following two aspects: a) Our ontology mainly focuses on the problem domain of security errors and the relevant security practice along secure software development lifecycle; and b) our ontology is context-based, which allows security domain knowledge to be described in the situation of the software project.

IV. An Ontology-Based Context Model

A. Design Consideration

The basic concept of our ontology design is to provide a vocabulary for representing knowledge about the software security domain and for providing linkages with specific situations in the application context. In order to effectively regulate the operation of security knowledge and be an essential part of the project knowledge, security knowledge must incorporate additional features. First, there is the requirement of a security domain model, which identifies fundamental entity types and relationships between them. With this model, all concepts of security domain knowledge are described at a level of abstraction, which enables cohesively treating entitles falling under the same conceptualization. Second and most important, knowledge, therefore security practices, must contain norms regarding how to apply them in different applications. In that respect, security knowledge must incorporate context, that is, to be modeled with certain characteristics of applications, such as software paradigms, programming languages and used technologies. The contextual information acts as a filter that defines, in a given context, what security knowledge pieces must be taken into account.

The main advantage of this ontology model is to share a common understanding of the structure of security knowledge along different SDLC activities and among different software development context to enable semantic interoperability. It also enables the reuse of domain knowledge, i.e. building a common security knowledge base integrated with contextual information, which describes portions of the domain knowledge. With this design, software engineers are allowed to find solutions to exceptional situations by searching for similar context. For example, a PHP web application designer can refer to other projects' security setup by looking for the same domain (subject area) and software technologies. Fig. 1 depicts the conceptual design of the context model for software security knowledge.

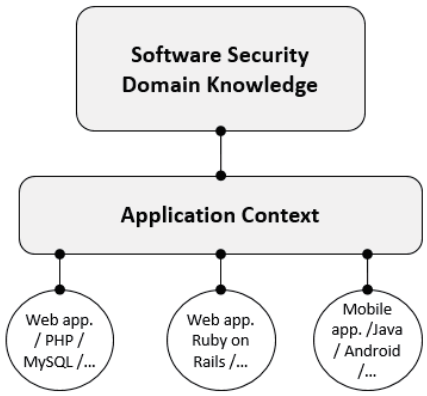

Fig. 1. The conceptual design of a context model for software security knowledge

B. Software Security Domain Model

The design of our security domain model is centralized in the idea of "To train software engineers on critical security errors." In accordance with above design consideration, we first reviewed security knowledge resources, including CWE, OWASP and SEI CERT coding guideline, etc. to identify knowledge catalogs that are important to describe the concepts related to security errors. We then associated security knowledge with the main stages throughout the entire SDLC and considered which terms are critical in explaining software security knowledge without overlap between concepts they

represent, meanwhile, to ease the information overloads for software engineers. Fig. 2 shows a graphical description of our domain ontological model for software security. We explain the major terms used in our ontology in the following:

1) Security Requirement

A software security requirement can be defined as a software requirement needed to avoid a specific software security error during the development [42]. More specifically, a software security requirement is a control or constraint which if not implemented may lead to a vulnerability.

2) Construction Practice

Construction practices focus on proactive activities for an organization to design and build secure software by default [43]. This typically includes design practices and coding practices.

- Design Practice: It represents security practices adopted in the system design time. By introducing the software design practices with secure architectures and services, the overall security risk from software development can be dramatically reduced.
- Coding Practice: It represents a set of rules that are adopted at the code level.

Knowledge elements in both architecture and design practices and coding practices are organized in three catalogs, Strategy, Method and Mechanism.

3) Verification Practice

Software verification is to assure that the software fully satisfies all the expected requirements [43], which typically include the following two practices.

- Code Review Practice: It is focused on the inspection of software at the source code level in order to spot security mistakes.
- Testing Practice: It is focused on the inspection of software while executed in order to find security problems.

Knowledge elements in both architecture and design practices and coding practices are organized in two catalogs, Technique and Approach.

4) Security Error

A software security error is a tangible manifestation of a mistake in the software development artifacts of a piece of software that causes a software weakness [42], [44]. In our ontology, a software security error can be one of the following:

- Design Flaw: An incorrect logical decision or oversight at the design level.
- Coding Error: A mistake (bug) occurs at the code level.

C. Application Context Model

The domain knowledge of software security needs to be put in a context so that it can adapt itself to different situations of software development. Our context ontologies are a collection of characteristics, which describe the properties of an application that the software project develops. Capturing the context is important in the context ontology modeling process where context representation depends on these characteristics and relationships created between them. The characteristics are listed in Table I.

TABLE I. IDENTIFIED CHARACTERISTICS (CLASSES) IN THE APPLICATION CONTEXT

Class	Definition	Example
Software paradigm	It represents the categories of software applications that share common development characteristics.	Web application, desktop application, mobile application
Subject area	It represents domains that an application belongs.	Banking, health, Travel
Software feature	It represents the essential elements of software with security concerns. The software feature is associated with the software paradigm and the subject area of the application.	User authentication, credit card processing, file upload.
Language	It represents programming language used to develop an application.	C, C++, Java, JavaScript, PHP
Technology	It represents the combination of frameworks or tools used to create an application. Technologies are built based on programming languages.	Web framework (Symfony for PHP, Angular for JavaScript, etc.) toolkit, SDK
System structure	It represents the fundamental structure to operate the application.	Database management system, runtime platform (MS Windows, Android)
Security Tool	It represents a concrete solution to implement construction mechanisms or verification techniques.	HTML Purifier, PHPUnit for PHP testing

Fig. 2 shows the completed ontology including the interrelationships of the security domain model and the application context model. We associate domain knowledge elements of software security, security requirements, and construction practices, with software features in the application context model. The linkage ensures that project management understands the security-relevant aspects of critical functional requirements. For example, if the software handles dynamic web page generation with user-supplied data, it is likely subject to the output encoding requirements, and the architecture and design can be verified they are prepared for this. Furthermore, software development staff can be primed with the possible security errors associated with the software feature. Subsequently, construction practices (rules and mechanisms) are set up by the used programming language, technologies and relevant security tools of the application. Thus, software engineers can get practical and concrete solutions according to the application's technology set.

V. IMPLEMENTATION AND EVALUATION OF THE ONTOLOGY

In order to evaluate the ontology, all the above classes and corresponding relationships are implemented using OWL (Web Ontology Language), a markup language based on RDF/XML (Resource Description Framework/Extensible Markup Language), and we used the Protégé OWL tool [42] to create the ontology. This web language has been developed by the Web Ontology Group as a part of the W3C Semantic Web Activity [43], [44]. For presenting our ontology model, a

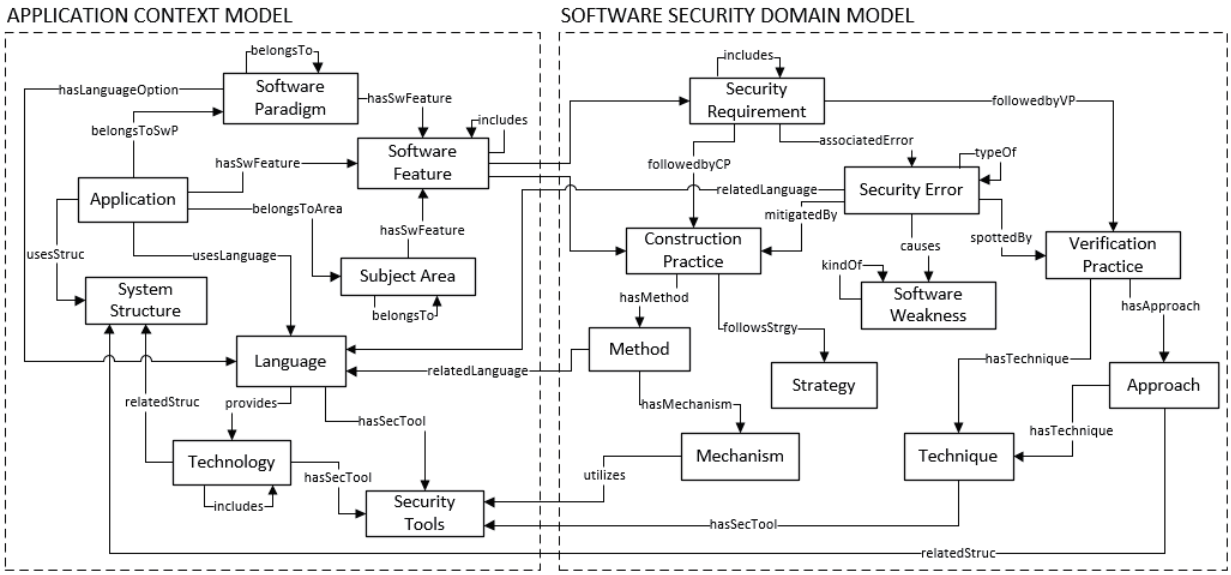

Fig. 2. The ontology-based context-sensitive model for software security knowledge

common security attack of web applications, Cross-Site Scripting (XSS), was considered to demonstrate the knowledge modeling process of software security domain. XSS is a kind of injection that aims at adding malicious script code – usually JavaScript – to a website so that the browser executes the code [45]. According to OWASP's Top 10 Application Security Risks – 2017 [46], it is the third most risky web applications' vulnerability and the most widespread.

Fig. 3 shows a part of the implementation. On the left side, the class hierarchy can be observed, and on the right side, the axioms that define the individuals are listed. In this case, we focused on modeling security errors on the code level (coding errors) and the related security practices. We analyzed and organized security knowledge related to XSS first, inserting knowledge content ("Individual" of Protégé) into each class of the domain model and building the relationships ("Object property" of Protégé). Fig. 4 presents the graphical display of the knowledge elements and their relationships built in the domain ontology.

To put this domain knowledge in context, we assume that a web application is developed with a software feature "generating dynamic web pages by inserting user-supplied data in HTML documents". The application uses PHP as the programming language, Symfony as the web framework and MySQL as the database management system. Other application characteristics, for example, operating system, web server or platform, were omitted in this scenario. We filled the context model with the application characteristics and assigned the corresponding object properties with the individuals (Security Requirement and Technique) in the domain model. Fig. 5 shows the graphical representation of this application scenario, combining context characteristics and security knowledge content. The diagram was simplified in order to highlight the critical relationships between the context model and the domain model. For an ontology to be considered useful, it must give consistent answers to real-world questions. The following exemplary competency question is an example used to reflect the above scenario:

What are the coding practices and relative technologies under PHP/Symfony web framework that can be used to secure the software feature "Generating dynamic web pages"?

Similarly, we modeled other common and critical security errors using the ontology throughout the knowledge modeling and the evaluation process, including SQL Injection, Command Injection and Cross-Site Request Forgery (CSRF), etc., which are all listed in OWASP Top 10. With increasing knowledge content filling in the ontology, the presence of detailed information can enable answering the various types of competency questions, such as (1) What are the possible security errors that developers might have while accessing database with user-supplied data and how can we spot them while doing a code review or system testing? (2) What PHP technologies we can adapt to mitigate SQL injection in terms of the parameterized query (technique)?

VI. Application of the Ontology

In previous sections, we have described the design, development, and evaluation of the ontological context model. In this section, we present a proposed application based on the ontology, a web-based knowledge management system for software security knowledge sharing and learning. The objective of this application is to allows learners or software project staff adaptively retrieve and present security knowledge content according to their working context. The functional and technical architecture along with the ontology are covered in this section.

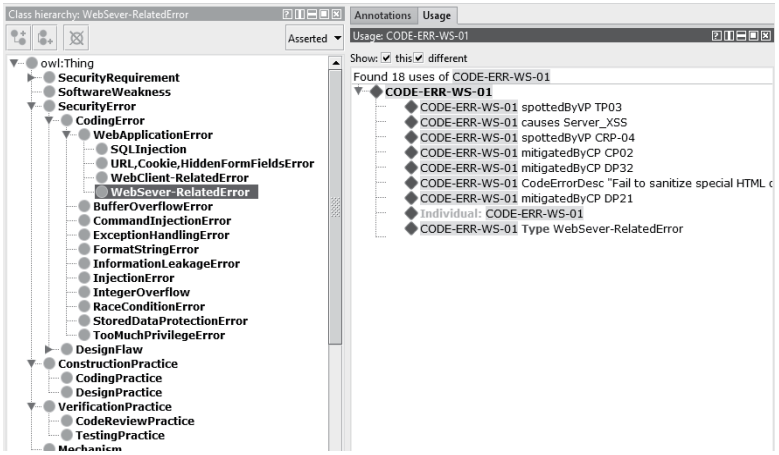

Fig. 3. Implementation of the ontology in Protégé

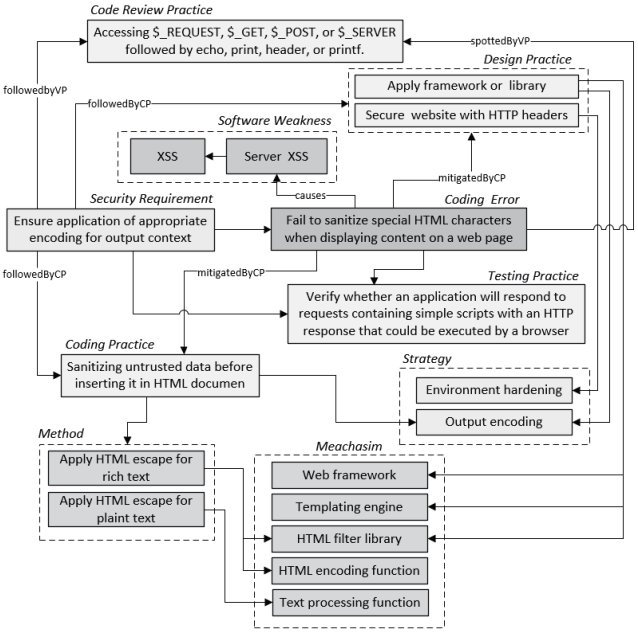

Fig. 4. The graphical display of the security domain model (a partial view)

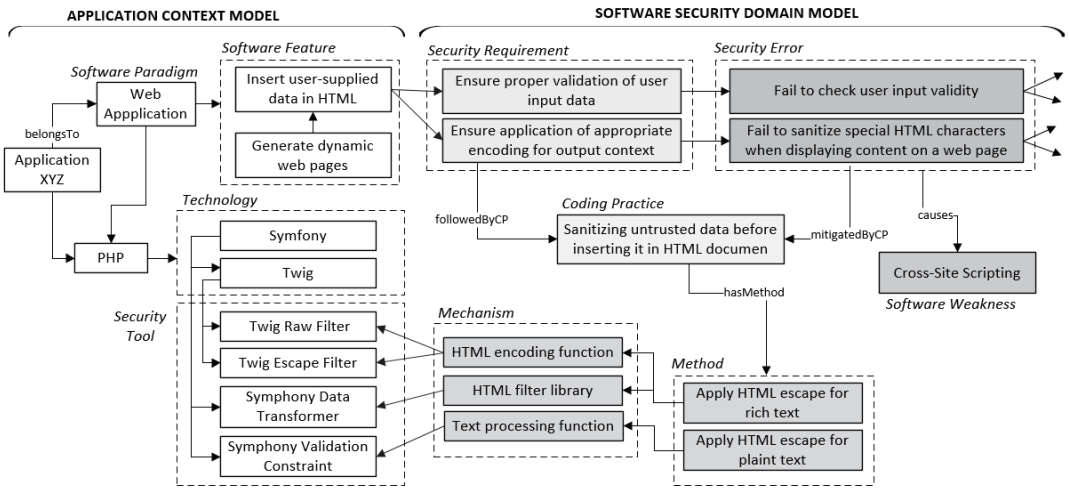

Fig. 5. The graphical display of knowledge content in the application context and security domain model (a partial view)

A. Functional Architecture

The functional architecture describes the functionalities supported in the application (Fig. 6), which are divided into two categories: knowledge presentation modules and system management modules. The knowledge representation module is to provide learners with learning materials. By making better use of the domain knowledge and contextual information, it is planned that the optimal materials are provided. The system management module is responsible for maintaining the ontology repository and loading knowledge content for the needs of the knowledge presentation. The functionalities of each module are briefed in the following sections.

1) Query Module

Users can access the security knowledge through a query interface passing requests to a search engine. The input criteria can be constructed dynamically or use pre-configured question patterns as our demonstrations in section 6.

2) Presentation scheme module

Regarding the knowledge presentation, two knowledge representation schemes are suggested in order to provide a navigation view and integrated information view of the knowledge items: knowledge maps and tutorials. Knowledge maps are the graphical representation of the knowledge items and their relationship. Knowledge maps can be used as primary sources for knowledge acquisition, adjunct aids to text processing, communication tools for organizing ideas, or retrieval cues [47]. Fig. 4 & 5 can be taken as examples to demonstrate a knowledge map of Cross-Site Scripting security errors. In tutorials, knowledge content is presented as a form of static web pages, which allow users to browse the detailed information and relevant resources, such as sample code or hyperlinks.

3) Knowledge service module

Knowledge service module provides services to receive requests from users, to interact with ontology management functions, and to process and display the result set according to the requested knowledge presentation format.

4) Ontology management module

Ontology management module is responsible for maintaining and loading data from the ontology, including specific individuals of classes, object properties, and the data properties.

B. Technical Architecture

Fig. 7 provides an overview of our proposed architecture implementing the main feature of the application outlined above. The front-end has been designed as JSP pages and through them, the users can access the various modules and functions of the application. Clients can interact with the server (Apache Tomcat) using an HTTP request to a Java Servlet. The backend is implemented in Java and access to the ontology repository is provided through the Jena API, a Java framework for building semantic web applications. Jena provides extensive Java libraries for helping developers develop code that handles RDF, OWL, and SPARQL in line with published W3C recommendations. Pellet is an open source OWL DL (descriptive logic) for Java, which is used to infer relevant knowledge from the ontology defined in the OWL. Pellet can also be integrated with Jena or OWL API libraries.

VII. Conclusion and Future Work

This paper presents a formal ontology-based context-sensitive model for knowledge management in the software security domain. The ontology has become a crucial role in enhancing the value of knowledge management in the software development, which facilitates reuse, sharing, and management knowledge in an efficient and effective manner. Our contributions in this paper are twofold. First, we develop an ontology to model software security related knowledge through combining domain knowledge of software security and the contextual characteristics of the application development. This proposal compared to the currently available knowledge models introduces a new perspective to model domain knowledge of software security. Second, we propose a knowledge management system, using our ontology as the central knowledge repository, where the functional architecture and the

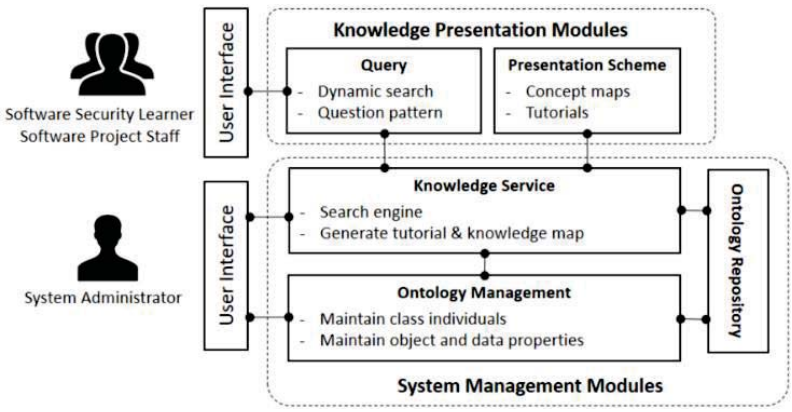

Fig. 6. The functional architecture of the proposed application

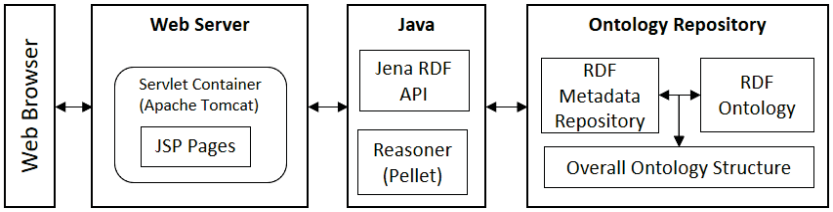

Fig. 7. The proposed architecture for the ontology application

system architecture are both presented. The ontology model can facilitate knowledge sharing services such as knowledge integration, advanced knowledge search, and knowledge visualization. It also supports the sharing and learning process about software security.

Security has become an important part of today's software development projects. A major portion of research and practical development in security software engineering is dedicated to developing security knowledge repositories, patterns, security taxonomies, and ontologies. However, these security practices and knowledge still cannot be effectively adopted and spread in software development. We argue that one of the prime reason for this is "Too much information, and lacking proper security knowledge management." As Gary McGraw points out "Security knowledge is more than simply a list of things we know or a collection of fact." [8]. Building secure applications is a complex and demanding task that developers often face, especially because the domain is rather context-specific, and the real project situation is necessary to apply the security concepts within the specific system. Since software engineers are not experts on security in general, there is an ever-increasing need to organize security knowledge in a fashion manner, helping developers, project staff, and learners learn the necessary security knowledge to fulfill the need of their work.

As the future work, we intend to implement the proposed intelligent application and to evaluate its usability in educational paradigms and software development projects. Our ultimate goal is to provide the software development community, e.g. open source software communities, a set of advanced services for efficiently handling and disseminating software security knowledge within the community.

REFERENCES

[1] Bishop, M. (2010), "A Clinic for" Secure" Programming". IEEE Security & Privacy, volume 8, issue 2, pages.

[2] Rus, I. and M. Lindvall (2002), "Knowledge management in software engineering". IEEE software, volume 19, issue 3, pages 26.

[3] Basili, V.R. and H.D. Rombach (1991), "Support for comprehensive reuse". Software engineering journal, volume 6, issue 5, pages 303-316.

[4] Lindvall, M. and I. Rus (2000), "Process diversity in software development". IEEE software, volume 17, issue 4, pages 14-18.

[5] De Araujo, R.M., et al. (2004), "Context Models for managing collaborative software development knowledge". in Workshop on Modeling and Retrieval of Context (MRC).

[6] Davies, J., D. Fensel, and F. Van Harmelen (2003), "Towards the semantic web: ontology-driven knowledge management". volume: John Wiley & Sons.

[7] Mead, N.R., et al. (2004), "Software security engineering: a guide for project managers". volume: Addison-Wesley Professional.

[8] McGraw, G. (2006), "Software security: building security in". volume 1. Addison-Wesley Professional.

[9] Viega, J. and G.R. McGraw (2001), "Building secure software: how to avoid security problems the right way". volume: Pearson Education.

[10] Xie, J., H.R. Lipford, and B. Chu (2011), "Why do programmers make security errors?". in Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on. IEEE.

[11] Graff, M. and K.R. Van Wyk (2003), "Secure coding: principles and practices". volume: " O'Reilly Media, Inc.".

[12] Birkenkrahe, M. (2002), "How large multi-nationals manage their knowledge". Business Review, volume 4, issue 2, pages 2-12.

[13] Brézillon, P. (2003), "Making context explicit in communicating objects". Communicating with Smart Objects: Developing Technology for Usable Pervasive Computing Systems, Kogan Page, London, volume, issue, pages.

[14] Brézillon, P. and R. Araujo (2005), "Reinforcing shared context to improve collaboration". Revue des Sciences et Technologies de l'Information-Série RIA: Revue d'Intelligence Artificielle, volume 19, issue 3, pages 537-556.

[15] Klemke, R. (2000), "Context Framework-an Open Approach to Enhance Organisational Memory Systems with Context Modelling Techniques". in PAKM.

[16] Brézillon, P. (2002), "Modeling and using context: Past, present and future", Rapport de recherche interne LIP6: Paris. pages.

[17] Jafari, M., et al. (2008), "Exploring the contextual dimensions of organization from knowledge management perspective". VINE, volume 38, issue 1, pages 53-71.

[18] Brézillon, P. and J.-C. Pomerol (1999), "Contextual knowledge sharing and cooperation in intelligent assistant systems". Le Travail Humain, volume, issue, pages 223-246.

[19] Goldkuhl, G. and E. Braf (2001), "Contextual knowledge analysis-understanding knowledge and its relations to action and communication". in Second European Conference on Knowledge Management Proceedings.

[20] Rosa, M.G., M.R. Borges, and F.M. Santoro (2003), "A conceptual framework for analyzing the use of context in groupware", in Groupware: Design, Implementation, and Use, Springer. pages 300-313.

[21] Perin, D. (2011), "Facilitating student learning through contextualization: A review of evidence". Community College Review, volume 39, issue 3, pages 268-295.

[22] Dolmans, D.H., et al. (2005), "Problem-based learning: Future challenges for educational practice and research". Medical education, volume 39, issue 7, pages 732-741.

[23] Ko, A.J. and B.A. Myers (2008), "Debugging reinvented: asking and answering why and why not questions about program behavior". in Proceedings of the 30th international conference on Software engineering. ACM.

[24] Apvrille, A. and M. Pourzandi (2005), "Secure software development by example". IEEE Security & Privacy, volume 3, issue 4, pages 10-17.

[25] Curtis, B., H. Krasner, and N. Iscoe (1988), "A field study of the software design process for large systems". Communications of the ACM, volume 31, issue 11, pages 1268-1287.
[26] Henninger, S. (1997), "Case-based knowledge management tools for software development". Automated Software Engineering, volume 4, issue 3, pages 319-340.
[27] Gruber, T.R. (1993), "A translation approach to portable ontology specifications". Knowledge acquisition, volume 5, issue 2, pages 199-220.
[28] Wand, Y., V.C. Storey, and R. Weber (1999), "An ontological analysis of the relationship construct in conceptual modeling". ACM Transactions on Database Systems (TODS), volume 24, issue 4, pages 494-528.
[29] Gruber, T.R. (1995), "Toward principles for the design of ontologies used for knowledge sharing?". International journal of human-computer studies, volume 43, issue 5-6, pages 907-928.
[30] Uschold, M. and M. Gruninger (1996), "Ontologies: Principles, methods and applications". The knowledge engineering review, volume 11, issue 2, pages 93-136.
[31] Noy, N.F. and D.L. McGuinness (2001), "Ontology development 101: A guide to creating your first ontology", Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, Stanford, CA. pages.
[32] Wang, X., et al. (2004), "Semantic space: An infrastructure for smart spaces". IEEE Pervasive computing, volume 3, issue 3, pages 32-39.
[33] Gruninger, M. (2002), "Ontology: applications and design". Commun. ACM, volume 45, issue 2, pages.
[34] Salini, P. and S. Kanmani (2013), "Ontology-based representation of reusable security requirements for developing secure web applications". International Journal of Internet Technology and Secured Transactions, volume 5, issue 1, pages 63-83.
[35] Busch, M. and M. Wirsing (2015), "An Ontology for Secure Web Applications". Int. J. Software and Informatics, volume 9, issue 2, pages 233-258.
[36] Gyrard, A., C. Bonnet, and K. Boudaoud (2013), "The stac (security toolbox: attacks & countermeasures) ontology". in Proceedings of the 22nd International Conference on World Wide Web. ACM.
[37] Kang, W. and Y. Liang (2013), "A security ontology with MDA for software development". in Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on. IEEE.
[38] Marques, M. and C.G. Ralha (2014), "An ontological approach to mitigate risk in web applications". the Proceedings of SBSeg, volume, issue, pages.
[39] Guo, M. and J.A. Wang (2009), "An ontology-based approach to model common vulnerabilities and exposures in information security". in ASEE Southest Section Conference.
[40] Khairkar, A.D., D.D. Kshirsagar, and S. Kumar (2013), "Ontology for detection of web attacks". in Communication Systems and Network Technologies (CSNT), 2013 International Conference on. IEEE.
[41] Razzaq, A., et al. (2014), "Ontology for attack detection: An intelligent approach to web application security". computers & security, volume 45, issue, pages 124-146.
[42] Tudorache, T., et al. (2013), "WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web". Semantic web, volume 4, issue 1, pages 89-99.
[43] Welty, C., D.L. McGuinness, and M.K. Smith (2004), "Owl web ontology language guide". W3C recommendation, W3C (February 2004) http://www. w3. org/TR/2004/REC-owl-guide-20040210, volume, issue, pages.
[44] Powers, S. (2003), "Practical RDF: solving problems with the resource description framework". volume: " O'Reilly Media, Inc.".
[45] Shema, M. (2012), "Hacking web apps: detecting and preventing web application security problems". volume: Newnes.
[46] OWASP, "OWASP Top 10 Application Security Risks - 2017"; Available from: https://www.owasp.org/index.php/Top_10-2017_Top_10.
[47] O'donnell, A.M., D.F. Dansereau, and R.H. Hall (2002), "Knowledge maps as scaffolds for cognitive processing". Educational psychology review, volume 14, issue 1, pages 71-86.