Journal of Computer Science 10 (8): 1411-1417, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.1411.1417 Published Online 10 (8) 2014 (http://www.thescipub.com/jcs.toc)

AWARENESS OF EMBEDDING SECURITY FEATURES INTO COMPONENT-BASED SOFTWARE DEVELOPMENT MODEL: A SURVEY

Hasan Kahtan, Nordin Abu Bakar and Rosmawati Nordin

Faculty of Computer and Mathematical Sciences, UniversitiTeknologi MARA Selangor, Malaysia

Received 2014-01-05; Revised 2014-02-23; Accepted 2014-03-20

ABSTRACT

Current applications and systems contain the software components as the basic elements and Component Based Software Development (CBSD) has been successful in building applications and systems. However, the security of CBSD for the software component is still lacking. This study highlights the results of a survey pertaining to the embedding of security features in the CBSD process. The main objective of this survey is to investigate the awareness of embedding security features in the CBSD process in the Malaysian context. For this purpose, experts from industry as well as from the academic community were interviewed. Moreover, an online survey was formulated and e-mailed to the experts and potential candidates. The results show that the embedding of security features in the software lifecycle is crucial because the incorporation of security activities in CBSD will minimize vulnerabilities in the software system, thus reducing system cost.

Keywords: Component-Based Software Development, Software Security, Reusable Software

1. INTRODUCTION

Component-Based Software Development (CBSD) is an emerging technology that focuses on building systems by integrating existing software components. The idea is to assemble software applications from reusable software codes, thereby simplifying software development in terms of time and budget constraints. CBSD offers a range of benefits, such as managing increasing complexity (Chen *et al*., 2011; Kumari and Bhasin, 2011; Sozer *et al*., 2011), improving the capability to reuse components (Lin, 2007; Fredriksson, 2008; Carvalho *et al*., 2009; Li *et al*., 2011), improving efficiency (Salmi and Ioualalen, 2012; Yang *et al*., 2012; Chen *et al*., 2012), decreasing the time and effort needed to develop software (Alhazbi and Jantan, 2007; Kaur *et al*., 2007; Ahmed *et al*., 2012), improving the quality of the system (Capretz, 2005; Mohanty *et al*., 2011), reducing production costs through software reuse (Chen *et al*., 2012; Barnawi *et al*., 2012), reducing maintenance costs (Li *et al*., 2011; Sommerville, 2011), increasing development productivity (Barnawi *et al*., 2012; Aris and Salim, 2007; Shang *et al*., 2011), ensuring a greater degree of consistency (Crnkovic, 2003; Ganguly and Bhattacharyya, 2011; Brada, 2011), providing a wider range of usability (Selvi *et al*., 2008; Jun *et al*., 2012) and supporting the effective use of specialists (Sommerville, 2011; Cann *et al*., 2004; Kaur and Mann, 2010). However, despite the wide adoption of CBSD in the software industry and the tremendous number of publications about it in academic research, CBSD still lacks essential formal foundations for the specification, composition and verification of security requirements. Therefore, current CBSD practices do not provide the essential requirements for developing secure systems. Several studies have reported different challenges involved in the use of CBSD. According to Moradian and Håkansson (2010), the interdependencies among software components create problems at the integration phase. Therefore, security features of software components must be considered and evaluated earlier in the CBSD lifecycle. This study

Corresponding Author: Hasan Kahtan, Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Selangor, Malaysia

highlights this issue by investigating the need to consider security features in the CBSD process.

1.1. Motivation

The existing CBSD processes presented in literature concentrate on the common activities involved in developing component-based software with emphasis on integrating and reusability by acquiring existing component from the repository with unknown security properties. Therefore, the current component-based development process appears to lack the capacity to develop secure systems. Given that security breaches are largely caused by vulnerable software, producing software in a secure manner is important because individuals and organizations mostly depend on software (Jain and Ingle, 2011). Likewise, there is great risk involved in constructing, deploying, operating and using software components that has not been developed with security awareness (Allen *et al*., 2008; Kahtan *et al*., 2012).

Software system design should consist of both functional and non-functional requirements. In component specification, the non-functional requirements refer to security attributes. However, current CBSD lacks non-functional support (Zschaler, 2010). In component or application, the non-functional requirements are equally important as the functional ones (Zschaler, 2010). Furthermore, non-functional requirements should be addressed at the early stage of the CBSD to avoid any costly failures in the future (Zschaler, 2010).

According to Talib *et al*. (2010), the lack of a suitable set of guides on the CBSD life cycle will lead to faults in the requirements, design, or codes of the software, which, in turn, will result in major security vulnerabilities. Meanwhile, Karen (2007) mentioned that a component, in its entire lifetime, may use different applications running in many types of environments and perform several tasks. The security provided by outside vendor software components usually does not provide the security features of all possible software system styles in all environment executions. A component might verify security in single application in a specific operating system, but that component might not do so in a completely different application. Moreover, according to Atan *et al*. (2007), the capability to deliver secure, high quality software applications within the allotted time and budget remains a challenge to most software development companies. Any faults in the software or delay in its delivery will cause problems for many individuals involved.

2. MATERIALS AND METHODS

A survey was conducted to investigate the awareness of the embedding of security features in the CBSD process in the Malaysian context. The interviews were conducted with experts from different industries and the academic community to obtain insights into the awareness of the embedding of security features in the CBSD process. The interviewees were also allowed to comment on other aspects of the research depending on their knowledge or interests. In addition to the interviews, an online survey was also formulated and e-mailed to potential candidates. A total of 360 candidates were involved in the survey and mainly included industry experts (i.e., managers, software developers, engineers), academic staff in the software engineering department of local universities and PhD and Master's degree students who are conducting research in related fields. The following criteria were considered in the selection of expert participants:

- Must work as a software architecture engineer or a software system designer or developer with a minimum of five years of experience
- Must have experience and expertise in using state-of-the-art CBSD model and technologies
- Must have experience and expertise in using state-of-the-art software security
- Must be willing to act as neutral assessor
- Willingness to provide valuable analysis and interpretation based on his/her experience

The questionnaire contains questions based on the CBSD processes and software security features. Respondents were required to mark their expert opinion on the given statements in the form of questions. The survey consisted of 23 questions of various types, including multiple-choice, short answer and ratings. The questionnaire is thus divided into three sections:

- Section A-Profile
- Section B-Desired features on Component-Based Software Development (CBSD)
- Section C-Software Security

A Likert scale of 1 to 5, as shown in **Table 1**, was used to obtain participant preferences or degree of agreement. The online survey developed can be referred to at ttp://inforec.uitm.edu.my/perseus/se.ashx?s=0B7FD90F2EAAFB80.

Table 1. Likert scale

5	Strongly agree
4	Agree
3	Fair
2	Disagree
1	Absolutely disagree

3. RESULTS

Figure 1 shows that 53% of the respondents strongly agree that the availability of a component repository is beneficial for them to store all information about the developed components, followed by 35% who agree and 12% are fair. By contrast, none of the respondents disagree that the availability of the component repository is beneficial for them. **Figure 2** shows the pie chart related to the importance of CBSD. The chart indicates that 50% of the respondents agreed that CBSD is important because it promotes reusability to higher levels of abstraction. However, only 4% of the respondents disagree on the importance of CBSD.

According to **Figure 3**, 96.7% of the respondents do not use a formal CBSD process for developing software system. Meanwhile, 3.3% of the respondents considered the use of a formal CBSD process for developing software systems. **Figure 4** presents that 65% of the respondents totally agreed (combination of strongly agree and agree) that security features are neglected during the lifecycle process in the industries. However, only 23% disagree with this issue. **Figure 5** indicates that 41% of the respondents strongly agree that component designers lack a security background, followed by 33% who agree, 9% for fair and 17% who disagree. Based on **Figure 6**, 56% of the respondents strongly agree that dealing with an external component is risky because such component is ambiguous in terms of security context. By contrast, only 1% of the respondents disagree about the risk of dealing with an external component in terms of security.

Moreover, 63% of the respondents agree on the concept of embedding security activities to the software development life cycle to minimize the number of security flaws. By contrast, only 9% of respondents disagree on embedding the security activities in the life cycle of software development. These results are represented by **Figure 7**. In addition, the benefits of incorporating the security activities to CBSD is reflected in **Figure 8**. The benefits include minimizing the vulnerabilities and threats in the software and reducing the system cost by finding faults during upfront analysis.

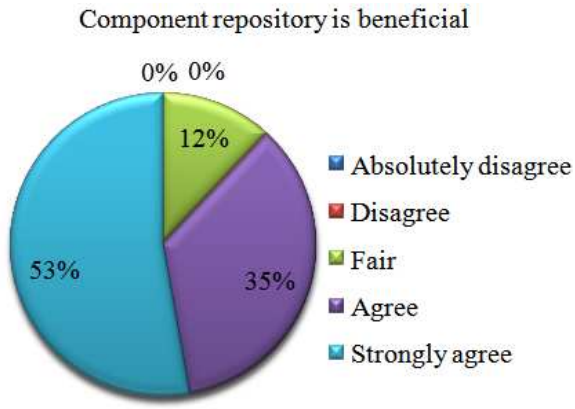

Fig. 1. Benefit of component repository

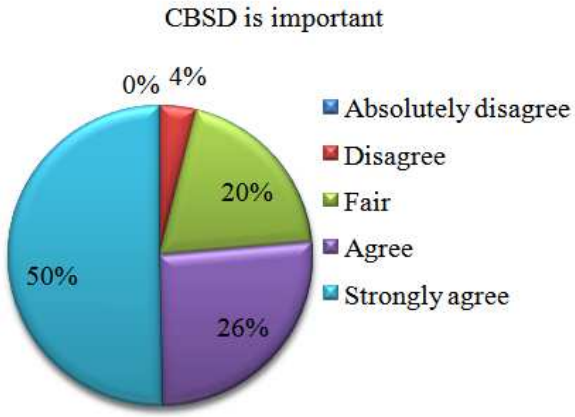

Fig. 2. Importance of CBSD

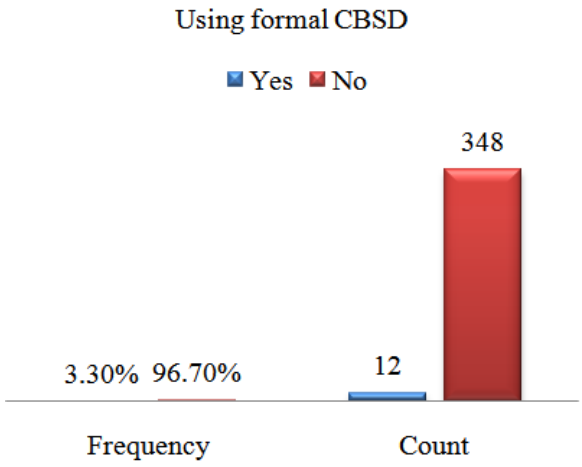

Fig. 3. Use of formal CBSD

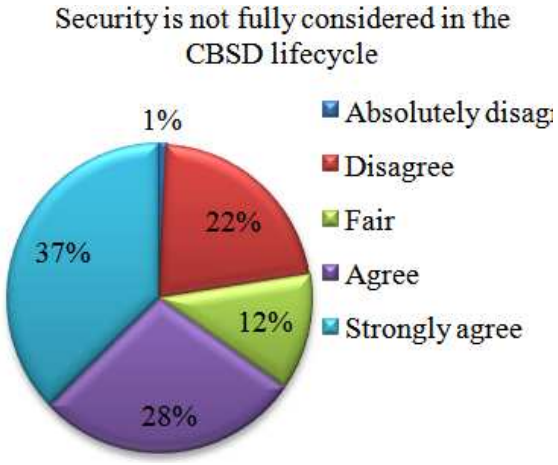

Fig. 4. Consideration on security

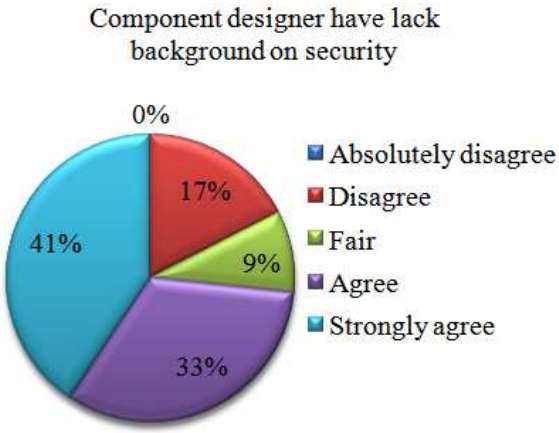

Fig. 5. Background of designer in terms of security

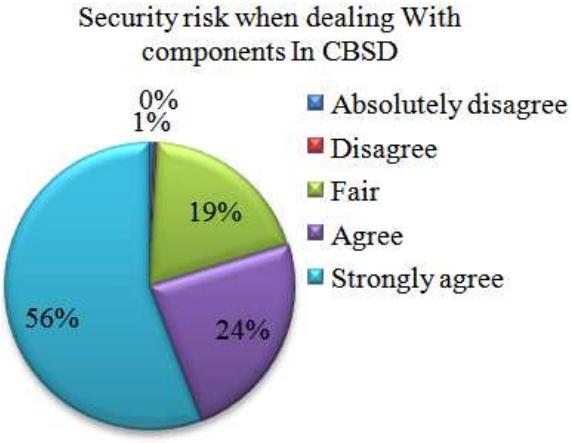

Fig. 6. Security risk of components in CBSD

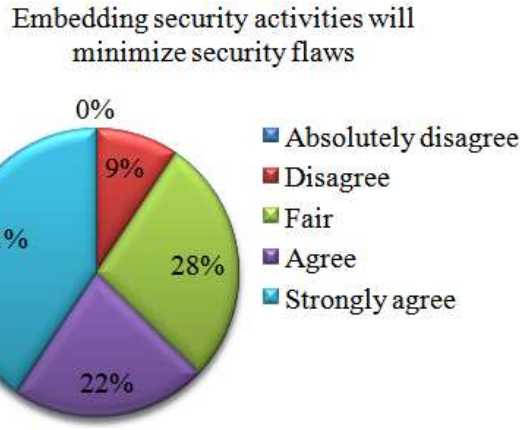

Fig. 7. Embedding security activities in CBSD

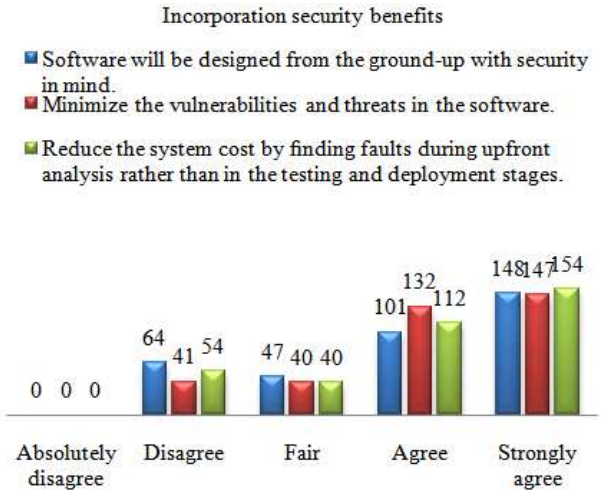

Fig. 8. Security benefits in CBSD process

4. DISCUSSION

Based on the survey results, CBSD promotes the effective use of specialists who can focus on developing reusable components within the scope of their knowledge instead of application specialists conducting the same work on different projects. Thus, software system developers can take advantage of existing structures and components, which improve the efficiency of software development. At the same time, CBSD creates a repository of components, which supports software system development by providing reusable and tested components. Therefore, CBSD is essential for achieving successful software reuse.

However, the survey results revealed that there is a risk when dealing with Commercial Off The Shelf

(COST). The development and deployment of components in binary form are outside the control of component users. The risk associated with the component selection with indefinite security properties is not acceptable. When such component is selected, the results might be catastrophic. Thus, the software component security assessment has become an increasingly important activity to guarantee the reliability of reusing software components.

Aside from that, the survey results highlighted that the current development of software is not supported by a formal model for CBSD. This is due to, programmers are highly educated people and prefer to develop software from scratch. They may feel that incorporating the work of others will limit their creativity. However, sharing is the key to successful CBSD and failure to do so will totally eliminate the opportunity for CBSD.

Moreover, the survey results discovered that the managements tend to focus more on risk from external vulnerabilities. However, many internal problems emerge from the ignorance of the developer instead of external vulnerabilities, which is equally risky because accidental failures could have a large impact. In addition, software developers who are unfamiliar with security features create software designs with no security consideration. Software developers lack information on how to develop software security. However, they are often asked to certify that their components are of trusted quality. Likewise, security issues are either neglected, added as an afterthought, or minimized due to cost or efficiency conditions in the development life cycle of software.

Therefore, the survey results confirmed that the security features must be presented and assessed at the earliest phases of the CBSD life cycle in order to achieve a predictable, repeatable process for engineering high-quality software components. Moreover, embedding security features to the software development lifecycle will mitigate the vulnerabilities and reduce the cost of maintenance.

5. CONCLUSION

This study presents the results of a survey on the awareness of embedding the security features in the CBSD process. The results show that CBSD is important in software production. Numerous organizations do not relatively consider the formal CBSD process for developing software system. Thus, a secure component to CBSD process is an urgent need. However, the survey indicates that security features seem to be neglected during the lifecycle process in industries. Indeed, embedding security activities to the software development life cycle is crucial to minimize the number of security flaws, thus reducing the cost as well.

This study has contributed to highlight the needs and the motivations to consider the security features in the CBSD process. Moreover, the survey results have revealed several motivations that contribute to a person's willingness to employ in such research both in the academic field (i.e., lecturers, students and researchers), as well as at the industry field (i.e., managers, software developers, engineers).

This study limited to 360 respondents and the questionnaire contains questions based on the CBSD processes and software security features only. Details questions on the software component requirements, design, implementation and testing throughout the CBSD process were not considered. Moreover, software security attributes are beyond the scope of this paper.

Future studies could start by identifying the security attributes that need to be embedded into the software components to mitigate the vulnerabilities. In addition, proposing a guideline for eliciting, analysing, specifying and composing the security attributes of the component based software development.

6. ACKNOWLEDGEMENT

This research is funded by Research Management Institute, UniversitiTeknologi MARA (UiTM). The authors would like to thank the participants in this survey, especially to Mr. ErrazudinIshak from MIMOS Berhad for his assistance in the process of data collection.

7. REFERENCES

Ahmed, N., M.R. Asim and M.R.J. Qureshi, 2012. A step forward to component-based software cost estimation in object-oriented environment. Pak. J. Sci., 62: 1202-2511.

Alhazbi, S. and A. Jantan, 2007. Dependencies Management in Dynamically Updateable Component-Based Systems. J. Comput. Sci., 3: 499-505 DOI : 10.3844/jcssp.2007.499.505.

Allen, J., S. Barnum, R. Ellison, G. McGraw and N. Mead, 2008. Software Security Engineering: A Guide for Project Managers. 1st Edn., Addison-Wesley Professional, ISBN-10: 0132702452, pp: 368.

Aris, H. and S. Salim, 2007. The development of a simplified process model for CBSD. Int. Arab J. Inform. Technol., 4: 89-96.

Atan, R., A.A.A. Ghani, M.H. Selamat and R. Mahmod, 2007. Automating measurement for software process models using attribute grammar rules. Int. J. Eng., 1: 24-24.

Barnawi, A., M.R.J. Qureshi and A.I. Khan, 2012. A framework for next generation mobile and wireless networks application development using hybrid component based development model. Int. J. Res. Rev. Next Generat. Netw., 2: 51-58.

Brada, P., 2011. Enhanced type-based component compatibility using deployment context information. Electr. Notes Theoretical Comput. Sci., 279: 17-31. DOI: 10.1016/j.entcs.2011.11.009

Cann, S., A. Rossi and P. Pilgrim, 2004. Frameworks for building component based applications.

Capretz, LF., 2005. Y: A new component-based software life cycle model. J. Comput. Sci., 1: 76-82. DOI: 10.3844/jcssp.2005.76.82.

Carvalho, F., S.R.L. Meira, B. Freitas and J. Eulino, 2009. Embedded software component quality and certification. Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications, Aug. 27-29, IEEE Xplore Press, Patras, pp: 420-427. DOI: 10.1109/SEAA.2009.90

Chen, J., H. Wang, Y. Zhou and S.D. Bruda, 2011. Complexity metrics for component-based software systems. Int. J. Digital Content Technol. Applic., 5: 235-244. DOI: 10.4156/jdcta.vol5.issue3.24

Chen, X., J. Zhao, K. Fu and Y. Chang, 2012. Refactoring of mechanical model simulation software based on component technology. Adv. Materials Res., 466: 1145-1149. DOI: 10.4028/www.scientific.net/AMR.466-467.1145

Crnkovic, I., 2003. Component-based software engineering-new challenges in software development. Software Focus, 2: 127-133. DOI: 10.1002/swf.45

Fredriksson, J., 2008. Improving predictability and resource utilization in component-based embedded real-time systems: School of Innovation. PhD Thesis, Mälardalen University.

Ganguly, D. and S. Bhattacharyya, 2011. Winning the Industrial Competitiveness with E-commerce Adopting Component-Based Software Architecture. In: Advances in Computer Science, Intelligent System and Environment, Jin, D. and S. Lin (Eds.)., Springer, Berlin, ISBN-10: 3642237533, pp: 69-75.

Jain, S. and M. Ingle, 2011. Software security requirements gathering instrument. Int. J., 2: 116-121.

Jun, G., W. Bo, W. Yunsheng, Z. Bin and W. Jiaojiao, 2012. Research of the software aging regeneration strategy based on components. Proceedings of the International Conference on Informatics, Cybernetics and Computer Engineering, Nov. 19-20, Springer, Melbourne, Australia, pp: 601-608. DOI: 10.1007/978-3-642-25188-7_74

Kahtan, H., N.A. Bakar and N. Rosmawati, 2012. Reviewing the challenges of security features in component based software development models. Proceedings of the IEEE Symposium on E-Learning, E-Management and E-Services, Oct. 21-24, IEEE Xplore Press, Kuala Lumpur, pp: 1-6. DOI: 10.1109/IS3e.2012.6414955

Karen, G., 2007. Software security assurance: A State-Of-the-Art Report (SOAR): DTIC Document.

Kaur, A. and K.S. Mann, 2010. Component based software engineering. Int. J. Comput. Applic., 2: 105-108.

Kaur, K., P. Kaur, J. Bedi and H. Singh, 2007. Towards a suitable and systematic approach for component based software development. World Acad. Sci. Eng. Technol., 27: 190-193.

Kumari, U. and S. Bhasin, 2011. A composite complexity measure for component-based systems. ACM SIGSOFT Soft. Eng. Notes, 36: 1-5. DOI: 10.1145/2047414.2047426

Li, D., X. Li, Z. Liu andV. Stolz, 2011. Interactive transformations from object-oriented models to component-based models. United Nations University, Macao.

Lin, J., 2007. Mapping UML component specifications to JEE implementations. J. Comput. Sci., 3: 780-785. DOI: 10.3844/jcssp.2007.780.785.

Mohanty, S., A.A. Acharya and D.P. Mohapatra, 2011. A model based prioritization technique for component based software retesting using UML state chart diagram. Proceedings of the 3rd International Conference on Electronics Computer Technology, Apr. 8-10, IEEE Xplore Press, Kanyakumari, pp: 364-368. DOI: 10.1109/ICECTECH.2011.5941719

Moradian, E. and A. Håkansson, 2010. Controlling security of software development with multi-agent system. Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, Sept. 8-10, Springer, Cardiff, UK., pp: 98-107. DOI: 10.1007/978-3-642-15384-6_11

Salmi, N. and M. Ioualalen, 2012. Towards efficient component performance analysis in component based architectures. Proceesdings of the Software Quality. Process Automation in Software Development, Jan. 17-19, Springer, Vienna, Austria, pp: 121-142. DOI: 10.1007/978-3-642-27213-4_9

Selvi, R.T., R. Meenakshi, N. Balasubramanian, G. Manohar and Q. Li *et al*., 2008. Framework and architectural style metrics for component based software engineering. Proceedings of the International Conference on Mathematics and Computers in Science and Engineering, (CSE '08), pp: 7-7.

Shang, M., H. Wang and L. Jiang, 2011. The Development process of component-based application software. International Conference Infor. Technology Computer Engineering Management Science, (EMS' 11), pp: 11-14.

Sommerville, I., 2011. Software Engineering. 8nd Edn., Addison Wesley.

Sozer, H., C. Hofmann, B. Tekinerdoğan, M. Aksit and E. Gelenbe *et al*., 2012. Runtime verification of component-based embedded software.

Talib, M.A., A. Khelifi and L. Jololian, 2010. Secure Software Engineering: A new teaching perspective based on the SWEBOK. Interdisciplinary J. Inform. Knowl. Manage., 5: 83-99.

Yang, Z., F. Ju and B. Shao, 2012. Research on integration of spatial data mining and gis based on component technology. Proceedings of the International Conference on Computational Environment Science, Jan. 15-16, Springer, Australia, Melbourne, pp: 161-167. DOI: 10.1007/978-3-642-27957-7_20

Zschaler, S., 2010. Formal specification of non-functional properties of component-based software systems. Software Syst. Model., 9: 161-201. DOI: 10.1007/s10270-009-0115-6