Journal of Computer Science 10 (7): 1298-1306, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.1298.1306 Published Online 10 (7) 2014 (http://www.thescipub.com/jcs.toc)

DEPENDABILITY ATTRIBUTES FOR INCREASED SECURITY IN COMPONENT-BASED SOFTWARE DEVELOPMENT

Hasan Kahtan, Nordin Abu Bakar and Rosmawati Nordin

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Selangor, Malaysia

Received 2014-01-05; Revised 2014-02-25; Accepted 2014-03-01

ABSTRACT

Existing software applications become increasingly distributed as their continuity and lifetimes are lengthened; consequently, the users' dependence on these applications is increased. The security of these applications has become a primary concern in their design, construction and evolution. Thus, these applications give rise to major concerns on the capability of the current development approach to develop secure systems. Component-Based Software Development (CBSD) is a software engineering approach. CBSD has been successfully applied in many domains. However, the CBSD capability to develop secure software applications is lacking to date. This study is an extension of the previous study on the challenges of the security features in CBSD models. Therefore, this study proposes a solution to the lack of security in CBSD models by highlighting the attributes that must be embedded into the CBSD process. A thorough analysis of existing studies is conducted to investigate the related software security attributes. The outcome analysis is beneficial for industries, such as software development companies, as well as for academic institutions. The analysis also serves as a baseline reference for companies that develop component-based software.

Keywords: Component-Based Software Development, Software Security Attributes, Dependability Attributes, Availability, Reliability, Integrity, Confidentiality, Safety, Maintainability

1. INTRODUCTION

Existing software applications become increasingly distributed as their continuity and lifetimes are lengthened; consequently, the users' dependence on these applications is increased. The dependability of these applications has become a primary concern in their design, construction and evolution. Kahtan *et al.* (2012) reported that pervasive computing gives rise to major concerns on the capability of current development models to develop dependable systems. Component-Based Software Development (CBSD) is a software engineering approach (Sommerville, 2011); its capability to develop dependable software applications is unknown to date.

CBSD is a technique that focuses on the use of existing software codes to develop software applications and thus avoids the need to develop from scratch (Alhazbi and Jantan, 2007; Lin, 2007). CBSD shifts the development emphasis from programming software to composing software systems (Gill and Tomar, 2010) by integrating existing software components based on the assumption that certain parts of a large software system reappear regularly and that common parts must be written once and then reused several times rather than written over and over again (Ahmed *et al*., 2012).

Several studies have reported the different challenges in using CBSD in software development in terms of component security trust. According to Moradian and Håkansson (2010), the interdependencies of software components create security issues during the integration phase of their development. Talib *et al*. (2010) stated that the lack of a suitable guide during the CBSD life cycle leads to faults in software requirements, design, or codes and thus results in major security threats. According to

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

Carvalho *et al*. (2009), selecting a component with unknown security properties is unacceptable because it may produce catastrophic results.

Therefore, dependability attributes must be verified and validated throughout the CBSD process to guarantee the dependability of software applications (Kahtan *et al*., 2012). The original definition of dependability underscores the justification of the trust. Dependability is defined as the level of reliability with which a property of a computer system can deliver the service it is supposed to provide (Laprie, 1992). Avizienis *et al*. (2004) defined dependability as the ability of a system to avoid service failures that are more frequent and severe than expected.

This study is organized as follows. Section 1 provides the introduction. Section 2 explains software security and CBSD in detail. Section 3 defines and reviews related studies on software security. Section 4 highlights software security attributes. Dependability attributes are defined in Section 5. Finally, the conclusion is provided in section 6.

2. SOFTWARE SECURITY AND CBSD

Current applications and systems contain software components as basic elements and CBSD is utilized to build applications and systems (Ahmed *et al*., 2012; Kouroshfar *et al*., 2009; Lau *et al*., 2011). However, CBSD lacks security in terms of software components (Hutchinson *et al*., 2004; Karen *et al*., 2011). Thus, the software can be considered a defective source. For example, malicious hackers and intruders exploit software defects (coding bugs such as buffer overflows and design flaws such as inconsistent error handling) to obtain unauthorized access and launch attacks. This problem emphasizes the need for designers to build a secure software (Al-Azzani and Bahsoon, 2010). In a recent example of hacking reported by Gibbs (2013), the Malaysian site of Google was hacked by a Pakistani group. The search service was replaced with a splash screen crediting the group before the site was taken completely offline. Researchers have acknowledged the importance of software security in the software development lifecycle. Due to poor software development practices, which include improper testing, failure to control common programming errors and poor understanding of the interactions between different software components that will lead to system failure (Kahtan *et al*., 2012). Examples of such opinions are discussed briefly below.

McGraw (2004) stated that in the fight for better software, treating the disease itself (poorly designed and implemented software) is better than taking an aspirin to stop the symptoms. The deep integration of software security into the development process and the engineering lessons from software practitioners have no substitute. Hadžiosmanović *et al*. (2012) discovered that vulnerabilities in Industrial Control Systems (ICSs), which include nuclear power plants and oil and gas extraction and distribution facilities, have increased mainly because of poor software development cycles used by several vendors and the “security by obscurity” paradigm used to “protect” legacy devices. Dobariya and Gajjar (2012) noted that one of the challenges in VoIP is that poor software development can lead to various security problems. Bygren *et al*. (2012) stated that poor software development in early-and mid-stage acquisition can result in failure to provide the desired results. This failure ranges from unwanted or missing features to cost and schedule overruns as well as critical flaws in system security or reliability. Sudhakar and Dava (2012) reported that poor software development practices occur because of marketing pressure and could lead to the release of unsecure server applications. These problems have caused an increase in traffic monitoring approaches. Thomas (2012) claimed that vulnerabilities can cause broad-range system failure, which is often compounded by poor software development. The majority of companies have discovered that they do not possess the engineering discipline to recreate binaries that are currently running their businesses.

Many technologists and commercial vendors have demonstrated that unsecure software affects people’s lives in many ways. For instance, the Symantec Security Response team (Security, 2012) highlighted in the 2011 symantec internet security threat report that more than 5.5 billion malicious attacks were blocked in 2011; this value is more than 81% higher than that in 2010. The Verizon Enterprise Solutions team (Verizon, 2012) revealed in The 2012 Data Breach Incident Report that 855 malicious incidents occurred in 2012; 174 million records were compromised. The sophos team (Sophos, 2012) reported in security threat report 2012 that Google removed more than 100 malicious applications from the Android Market in 2011 when hackers exploited vulnerabilities in the system. Secunia (2011) stated in secunia yearly report 2011 that the

vulnerabilities that affect typical end points have tripled to more than 800 and that the majority of these vulnerabilities (79%) are found in third-party programs. Technologies (2011) team highlighted in that organizations collectively lose 545 man hours a year on the average owing to IT downtime and data recovery.

In conclusion, protecting a continuously evolving network is difficult even when the software is not updated every five minutes. If a software has a defensively designed self-protection mechanism, is properly tested from a security perspective, or has few vulnerabilities, operating a secure network will be easy and cost effective. Progress should be attained on both fronts. However, in the long term, codes that are easy to defend must be developed (McGraw, 2011). Researchers and technologists need to understand how to manage poor software development practices. The software security approach aims to provide such help by exploring the development of best practices. Software security involves helping CBSD designers remove the burden of security problems from end users.

3. SOFTWARE SECURITY

The term "software security" has been defined by many authors in literature. McGraw (2004) stated that software security is engineering software that can continue to function correctly under malicious attacks. Cogo (2013) defined software security as a function focused on vulnerability extraction and system protection. Ghebremedhin (2012) defined software security as a measure that prevents the danger or risk of attacks. The Department of Homeland Security (Karen *et al*., 2006) defined secure software as software that cannot be forced to perform unintended functions. Software security also refers to the process of creating software that is considered secure (Boampong and Wahsheh, 2012). This process includes a robust design that deters software attacks through the identification and expulsion of problems in the software itself and through the provision of proper information to software developers, architects and users with regard to the development of secure applications.

One important goal of software security is to ensure justifiable confidence in the following basic features: The software under consideration functions in the intended manner and does not compromise the security or any other required properties of the software; the software environment or managed information is reliable and can continue to operate under all anticipated circumstances; and the software does not have vulnerabilities (McGraw, 2011). Such vulnerabilities may refer to an anomalous and hostile environment and to other utilization conditions (Karen, 2009). For this reason, software developers must anticipate these conditions and then design and implement the software to manage these challenges efficiently. Such conditions may include any of the following: Exposure of the operational software to accidental events threatening its basic security; exposed to intentional choices or actions that threaten its security during its development, deployment, operation, or continuity; and the presence of unintentional faults in the software and its environment.

Many studies have stated that the only means to solve software vulnerability is to consider software security development (Khaled and Han, 2006; Kim, 2004; McGraw, 2004; Mir and Quadri, 2012; Simpson, 2012). However, efforts to measure and improve software security remain under investigation (Alberts *et al*., 2012; Colombo *et al*., 2012; Karen *et al*., 2006; Lai, 2012; Steward *et al*., 2012). To address the research problem and the gaps identified in existing literature, the following issues are investigated:

- What are the common software security attributes addressed in the literature
- What significant attributes should be considered to solve the shortcomings of CBSD

4. SOFTWARE SECURITY ATTRIBUTES

In literature, the terms "dependability," "trustworthiness," and "survivability" are used interchangeably to describe the properties of software security (Karen, 2009).

4.1. Dependability

Dependable software executes predictably and operates correctly under all conditions even when the conditions are hostile, the software is under attack, or the software operates on a malicious host.

4.2. Trustworthiness

Trustworthy software contains few vulnerabilities or weaknesses that can be intentionally exploited to subvert

or sabotage the reliability of the software. To be considered trustworthy, the software must contain no logic that causes it to behave in a malicious manner.

4.3. Survivability

Survivability (also referred to as "resilience"): Software is considered resilient when it can (a) resist (i.e., protect itself against) or tolerate (i.e., continue operating dependably in spite of most known attacks, including as many novel attacks as possible) attacks and (b) recover quickly with as little damage as possible from attacks that the software can neither resist nor tolerate.

In several studies, the trustworthiness attribute is required in CBSD to achieve a secure software component (Muhammad and Zulkernine, 2011; Yan and Prehofer, 2011; Zhou, 2010). Other researchers believe that survivability (or resilience) is necessary to have secure software components (Gama *et al*., 2012; Zhiwen *et al*., 2010). The dependability attribute is a widely accepted concept through which to achieve better security in most software components (Crnkovic and Grunske, 2007; Xu *et al*., 2006; Yi and Li, 2011). In Security in The Software Lifecycle (Karen *et al*., 2006), the Department of Homeland Security defined "dependability" as the degree to which the software is operable and capable of performing its functionality or delivering a service that can justifiably be relied upon (i.e., trusted) as correct. To achieve dependability, the software must have the ability to avoid service failures that are frequent, severe, or have longer duration and are considered acceptable to users. Survivability and trustworthiness are closely related to dependability (Karen *et al*., 2006).

According to Sommerville (2011), the dependability of a computer system is the property of the system that reflects the user's degree of trust in that system. The most important dimensions of dependability are availability, reliability, safety and security. Moreover, the dependability of a computer system is a property that equates to its trustworthiness. The dependability attributes in CBSD has been considered in many studies (De Andrés *et al*., 2008; Gallina *et al*., 2012; Kharchenko *et al*., 2009; Tambe *et al*., 2010; Yi and Li, 2011). Dependability counters the security vulnerabilities, abnormal behavior and untrustworthy issues in a software system. To consider a system dependable, it must be viewed according to different but complementary properties (or instances of dependability), such as availability, integrity, reliability, safety, maintainability and confidentiality (Avizienis *et al*., 2004; Dai *et al*., 2006; Redwine Jr, 2007).

In conclusion, current literature shows that dependability attributes are the cure for security threats, abnormal behavior and untrustworthy issues in a software system. Therefore, in this study, dependability attributes are considered to overcome the security lacking in component-based software development.

5. DEPENDABILITY ATTRIBUTES

In simple terms, dependability ensures that the software always operates correctly. Several low-level properties can be viewed as attributes of dependability as well. **Table 1** presents the dependability attributes (Avizienis *et al*., 2004; Redwine Jr, 2007).

Two interesting questions must be considered in relation to this topic: How can these properties be satisfied collectively in one CBSD process and can the current state of the CBSD approach collectively address the requirements.

Tremendous research efforts have been exerted in current literature, resulting in many studies on integrating all six dependability attributes into the CBSD process. **Table 2** highlights the dependability attributes integrated into the CBSD process addressed in the current literature.

However, each study on verifying safety and maintainability as well as on estimating reliability, integrity, confidentiality and availability properties in CBSD has progressed independently (Crnkovic and Grunske, 2007). This condition can be attributed to the following reasons: (1) the attributes of safety and maintainability that address challenges must be specified, composed and verified in the software component; (2) the traditional ways to estimate reliability and availability attributes in a system architecture using stochastic methods are based on uncertain and inaccurate parameters; and (3) studies that analyze the vulnerabilities of confidentiality and integrity attributes are inadequate. All of these attributes must be embedded into the CBSD process to develop dependable component-based software systems. If the CBSD approach is enhanced in this manner, specifying and verifying these same attributes would be possible.

Table 1. Dependability attributes

Dependability attributes	Description
Availability	The software should be operational and ready to provide the correct service. It must be easily accessible to its intended and authorized users, referring to both humans and processes. The availability of a system is the probability that it is capable of delivering useful services at any time.
Reliability	The continuity of correct service refers to the probability that the system can aptly the expected services to the user over a given period of time. It also refers to the probability of failure-free operation over a certain period of time for a specific purpose within a particular environment.
Integrity	This attribute refers to the fact that the software must be protected from subversion; improper system alterations must be absent. Unauthorized modifications by unauthorized entities, such as corruption, overwriting, destruction, tampering and insertion of unintended (including malicious) logic or deletion, lead to subversions in the system. In such cases, integrity must be preserved during the development and implementation of the software.
Confidentiality	This attribute refers to the absence of unauthorized disclosure of information as applied to the software rather than the data being handled. Moroever, its refers to the characteristics, existence and/or protected or hidden content of unauthorized entities. Confidentiality often prevents unauthorized entities from learning about the system through reverse engineering or by developing effective attacks against the system.
Safety	This attribute refers to the absence of catastrophic effects on both the environment and the user. It can also refer to the possibility that the system can cause damage to the environment and people.
Maintainability	This attribute refers to the ability to go through repairs and modifications. New requirements emerge as systems are used.Thus, the usefulness of a system must be maintained by modifying it to accommodate such requirements. Maintainable software is software that can be adapted economically to manage new requirements; in such instances, the modifications are unlikely to introduce new errors into the system.

Table 2. Dependability attributes in the CBSD proces

Dependability attributes Publications	Availability	Reliability	Integrity	Confidentiality	Safety	Maintainability
Koziolek *et al*. (2013)	√					
Li *et al*. (2012)		√				
Machida *et al*. (2011)	√					
Reussner *et al*. (2003)	√	√				
Matevska and Hasselbring (2007)	√					
Grunske (2007)	√	√			√	
Lanoix *et al*. (2007)	√		√	√	√	
Jha *et al*. (2013)	√	√				√
Mir and Quadri (2012)	√		√	√		
Nicolas *et al*. (2011)			√	√		
Conmy and Bate (2010)					√	
Vidushi and Baliyan (2011)						√
Sharma *et al*. (2009)						√
Agrawal (2012)						√

6. CONCLUSION

A thorough analysis of existing research was conducted in this study to investigate the related software security attributes. Literature indicates that several attributes are used interchangeably to describe the properties of software security. Hence, the dependability attributes utilized to solve the lack of security in the CBSD process were proposed in this study. Six dependability attributes were identified:

Availability, reliability, confidentiality, integrity, safety and maintainability. By considering these dependability attributes, the CBSD product will be cured of security threats, abnormal behavior and untrustworthy issues. Moreover, embedding dependability attributes will help CBSD developers relieve end users of the burden of security problems.

This study has contributed to highlight the importance of embedding the CBSD process into the dependability attributes, the analysis in this study serves as a baseline reference for software development companies in promoting the CBSD. The analysis is beneficial for industries, such as software development companies and can also be used for academic purposes in order to fill the gap in existing CBSD models.

This study limited to address the importance of embedding the CBSD process into the dependability attributes. Discussion of embedding the CBSD phases (i.e., requirement analysis, design, implementation and testing) into dependability attributes is beyond the scope of this paper.

Future studies could start by identifying the vulnerabilities that affect the dependability attributes. In addition, designing a model for developing a dependable component-based software, which can overcome the weaknesses of the existing models while retaining their strengths.

7. ACKNOWLEDGEMENT

This research is funded by the Research Management Institute, Universiti Teknologi MARA (UiTM).

8. REFERENCES

Agrawal, J.R.D.V.K., 2012. Study of perfective maintainability for component-based software systems using aspect-oriented-programming techniques. Proceedings osf the International Conference on Intelligent Computational Systems, Jan. 7-8, Dubai, pp: 62-66.

Ahmed, B., A.H. Al-Talhi, M. Qureshi and A.I. Khan, 2012. Novel component-based development model for SIP-based mobile application. King AbdulAziz Univers.

Al-Azzani, S. and R. Bahsoon, 2010. Using implied scenarios in security testing. Proceedings of the ICSE Workshop on Software Engineering for Secure Systems, May 01-08, ACM New York, NY, USA., pp: 15-21. DOI: 10.1145/1809100.1809103

Alberts, C.J., J.H. Allen and R.W. Stoddard, 2012. Risk-based measurement and analysis: Application to software security. Software Engineering Institute, Carnegie Mellon University.

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.

Avizienis, A., J.C. Laprie, B. Randell and C. Landwehr, 2004. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput., 1: 11-33. DOI: 10.1109/TDSC.2004.2

Boampong, P.A. and L.A. Wahsheh, 2012. Different facets of security in the cloud. Proceedings of the 15th Communications and Networking Simulation Symposium, Mar. 26-29, Society for Computer Simulation International San Diego, CA, USA.

Bygren, S., G. Carrier, T. Maher, P. Maurer and D. Smiley *et al*., 2012. Applying the fundamentals of quality to software acquisition. Proceedings of the IEEE International, Systems Conference, Mar. 19-22, IEEE Xplore Press, Vancouver, BC., pp: 1-6. DOI: 10.1109/SysCon.2012.6189447

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. 17-29, IEEE Xplore Press, Patras, pp: 420-427. DOI: 10.1109/SEAA.2009.90

Cogo, V.V., 2013. Diversity in Automatic Cloud Computing Resource Selection. 1st Edn., LAP LAMBERT Academic Publishing, ISBN-10: 3659433594, pp: 68.

Colombo, R.T., M.S. Pessôa, A.C. Guerra and C.C. Gomes, 2012. Prioritization of software security intangible attributes. ACM SIGSOFT Soft. Eng. Notes, 37: 1-7. DOI: 10.1145/2382756.2382781

Conmy, P. and I. Bate, 2010. Component-based safety analysis of FPGAs. IEEE Trans. Indust. Inform., 6: 195-205. DOI: 10.1109/TII.2009.2039938

Crnkovic, I. and L. Grunske, 2007. Evaluating dependability attributes of component-based specifications. Proceedings of the 29th International Conference on Software Engineering-Companion, May 20-26, IEEE Xplore Press, Minneapolis, MN, USA., pp: 157-158. DOI: 10.1109/ICSECOMPANION.2007.36

Dai, Y.S., T. Marshall and X. Guan, 2006. Autonomic and dependable computing: Moving towards a model-driven approach. J. Comput. Sci., 2: 496-504. DOI: 10.3844/jcssp.2006.496.504

De Andrés, D., J.C. Ruiz and P. Gil, 2008. Designing component-based vlsi systems: Core selection using dependability, performance, area and power consumption measures. Universidad Politécnica de Valencia (UPV).

Dobariya, D. and J. Gajjar, 2012. Threats in SIP based voip systems.

Gallina, B., M.A. Javed, F.U. Muram and S. Punnekkat, 2012. A model-driven dependability analysis method for component-based architectures. Proceedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications, Sept. 5-8, IEEE Xplore Press, Cesme, Izmir, pp: 233-240. DOI: 10.1109/SEAA.2012.35

Gama, K., W. Rudametkin and D. Donsez, 2012. Resilience in dynamic component-based applications. Proceedigs of the 26th Brazilian Symposium on Software Engineering, Sept. 23-28, IEEE Xplore Press, Natal, pp: 191-195. DOI: 10.1109/SBES.2012.32

Ghebremedhin, A.G., 2012. Combining static source code analysis and threat assessment modeling for testing open source software security.

Gibbs, S., 2013. Google Malaysia taken offline by Pakistani hackers. The Guardian.

Gill, N.S. and P. Tomar, 2010. Modified development process of component-based software engineering. ACM SIGSOFT Soft. Eng. Notes, 35: 1-6. DOI: 10.1145/1734103.1734120

Grunske, L., 2007. Early quality prediction of component-based systems-a generic framework. J. Syst. Soft., 80: 678-686. DOI: 10.1016/j.jss.2006.08.014

Hadžiosmanović, D., L. Simionato, D. Bolzoni, E. Zambon and S. Etalle, 2012. N-Gram against the machine: On the feasibility of the n-gram network analysis for binary protocols. Res. Attacks, Intrusions Defenses, 354-373. DOI: 10.1007/978-3-642-33338-5_18

Hutchinson, J., G. Kotonya, I. Sommerville and S. Hall, 2004. A service model for component-based development. Proceedings of the 30th Euromicro Conference, Aug. 31-Sept. 3, IEEE Xplore Press, pp: 162-169. DOI: 10.1109/EURMIC.2004.1333368

Jha, P.C., V. Bali, S. Narula and M. Kalra, 2013. Optimal component selection based on cohesion and coupling for component based software system under build-or-buy scheme. J. Comput. Sci. DOI: 10.1016/j.jocs.2013.07.003

Kahtan, H., N.A. Bakar and R. Nordin, 2012. Reviewing the challenges of security features in component based software development models. Proceedings of the IEEE Symposium 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., 2009. Introduction to software security.

Karen, G., T. Winograd and B.A. Hamilton, 2011. Safety and security considerations for component based engineering of software-intensive systems.

Karen, G., T. Winograd, H.L. McKinley, P. Holley and B.A. Hamilton, 2006. Security in the software life cycle: Making software development processes-and the software produced by them-more secure. Department of Homeland Security.

Khaled, K.M. and J. Han, 2006. Assessing security properties of software components: A software engineer's perspective. Prcoeedings of theAustralian Software Engineering Conference, Apr. 18-21, IEEE Xplore Press, Sydney, NSW., pp: 10-10. DOI: 10.1109/ASWEC.2006.13

Kharchenko, V., V. Sklyar and A. Siora, 2009. Dependability of safety-critical computer systems through component-based evolution. Proceedigns of the 4th International Conference on Dependability of Computer Systems, Jun. 30-Jul. 2, IEEE Xplore Press, Brunow, pp: 42-49. DOI: 10.1109/DepCoS-RELCOMEX.2009.22

Kim, H., 2004. A framework for security assurance in component based development. Comput. Sci. Applic. DOI: 10.1007/978-3-540-24707-4_70

Kouroshfar, E, Y.H. Shahir and R. Ramsin, 2009. Process patterns for component-based software development. Component-Based Soft. Eng. DOI: 10.1007/978-3-642-02414-6_4

Koziolek, A., D. Ardagna and R. Mirandola, 2013. Hybrid multi-attribute QoS optimization in component based software systems. J. Syst. Software, 86: 2542-2558. DOI: 10.1016/j.jss.2013.03.081

Lai, S.T., 2012. An analyzer-based security measurement model for increasing software security. Int. J. Comput. Sci., 4: 81-91.

Lanoix, A., H. Denis, M. Heisel and J. Souquieres, 2007. Enhancing dependability of component-based systems. Proceedings of the 12th Ada-Europe International Conference on Reliable Software Technologies, Jun. 25-29, Springer Berlin Heidelberg, Geneva, Switzerland, pp: 41-54. DOI: 10.1007/978-3-540-73230-3_4

Laprie, J.C., 1992. Dependability: Basic concepts and terminology of dependable computing and fault-tolerant systems.

Lau, K.K., F.M. Taweel and C.M. Tran, 2011. The W model for component-based software development. Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications, Aug. 30-Sept. 2, IEEE Xplore Press, Oulu, pp: 47-50. DOI: 10.1109/SEAA.2011.17

Li, Q., H. Yang and H. Wang, 2012. Component-based software system reliability allocation and assessment based on ANP and petri. Proceedings of the International Conference on Reliability, Risk, Maintenance and Safety Engineering, Jun. 15-18, IEEE Xplore Press, Chengdu, pp: 227-231. DOI: 10.1109/ICQR2MSE.2012.6246225

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

Machida, F., E. Andrade, D.S. Kim and K.S. Trivedi, 2011. Candy: Component-based availability modeling framework for cloud service management using sysml. Proceedings of the 30th IEEE Symposium on Reliable Distributed Systems, Oct. 4-7, IEEE Xplore Press, Madrid, pp: 209-218. DOI: 10.1109/SRDS.2011.33

Matevska, J. and W. Hasselbring, 2007. A scenario-based approach to increasing service availability at runtime reconfiguration of component-based systems. Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, Aug. 28-31, IEEE Xplore Press, Lubeck, pp: 137-148. DOI: 10.1109/EUROMICRO.2007.10

McGraw, G., 2004. Software security. Security Privacy, IEEE, 2: 80-83. DOI: 10.1109/MSECP.2004.1281254

McGraw, G., 2011. Software security and the building security in maturity model.

Mir, I.A. and S.M.K. Quadri, 2012. Analysis and evaluating security of component-based software development: A security metrics framework. Int. J. Comput. Netw. Inform. Security, 4: 21-21. DOI: 10.5815/ijcnis.2012.11.03

Moradian, E. and A. Håkansson, 2010. Controlling security of software development with multi-agent system. Knowledge-Based Intell. Inform. Eng. Syst., DOI: 10.1007/978-3-642-15384-6_11

Muhammad, K. and M. Zulkernine, 2011. Building components with embedded security monitors. Proceedings of the Joint ACM SIGSOFT Conference-QoSA and ACM SIGSOFT Symposium-ISARCS on Quality of Software Architectures-QoSA and Architecting Critical Systems-ISARCS, Jun. 20-24, ACM New York, NY, USA., pp: 133-142. DOI: 10.1145/2000259.2000282

Nicolas, N., K. Boudaoud, C. Delettre and M. Riveill, 2011. A component-based approach to security protocol design. Proceedigns of the IEEE Workshops of International Conference on Advanced Information Networking and Applications, Mar. 22-25, IEEE Xplore Press, Biopolis, pp: 279-284. DOI: 10.1109/WAINA.2011.34

Redwine Jr, S.T., 2007. Software assurance: A curriculum guide to the common body of knowledge to produce. Acquire and Sustain Secure Software. H. Security.

Reussner, R.H., H.W. Schmidt and I.H. Poernomo, 2003. Reliability prediction for component-based software architectures. J. Syst. Softw., 66: 241-252. DOI: 10.1016/S0164-1212(02)00080-8

Secunia, 2011. Secunia yearly report.

Security, S.E., 2012. Symantec internet security threat report-2011 Trends.

Sharma, A., P.S. Grover and R. Kumar, 2009. Predicting maintainability of component-based systems by using fuzzy logic. Contemporary Comput., 40: 581-591. DOI: 10.1007/978-3-642-03547-0_55

Simpson, S., 2012. Sharing lessons learned: “Practiced” practices for software security. Datenschutz und Datensicherheit-DuD, 36: 641-644. DOI: 10.1007/s11623-012-0218-z

Sommerville, I., 2011. Software Engineering and Pearson.

Sophos, 2012. Security threat report.

Steward, C., L.A. Wahsheh, A. Ahmad, J.M. Graham and C.V. Hinds *et al*., 2012. Software security: The dangerous afterthought. Proceedings of the 9th International Conference on Information Technology: New Generations, Apr. 16-18, IEEE Xplore Press, Las Vegas, NV., pp: 815-818. DOI: 10.1109/ITNG.2012.60

Sudhakar, A. and U.D. Dava, 2012. An approach to detect and prevent denial of service attacks and worms using distributed denial-of-service detection mechanism. Int. J. Eng.,

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

Tambe, S., A. Dabholkar, J. Balasubramanian, A. Gokhale and D.C. Schmidt, 2010. Model-driven tools for dependability management in component-based distributed systems. Vanderbilt University.

Technologies, C.A., 2011. Research report: The avoidable cost of downtime.

Thomas, M., 2012. Accidental systems, hidden assumptions and safety assurance. Achieving Syst. Safety. DOI: 10.1007/978-1-4471-2494-8_1

Verizon, 2012. Data breach investigations report.

Vidushi, S. and P. Baliyan, 2011. Maintainability analysis of component based systems. Int. J. Softw. Eng. Applic., 5: 107-118.

Xu, L., H. Ziv, T.A. Alspaugh and D.J. Richardson, 2006. An architectural pattern for non-functional dependability requirements. J. Syst. Softw., 79: 1370-1378. DOI: 10.1016/j.jss.2006.02.061

Yan, Z. and C. Prehofer, 2011. Autonomic trust management for a component-based software system. IEEE Trans. Dependable Secure Comput., 8: 810-823. DOI: 10.1109/TDSC.2010.47

Yi, S. and D. Li, 2011. The research of component-based dependable encapsulation. Proceedings of the 13th IASME/WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering Conference on Applied Computing, (CAC' 11), World Scientific and Engineering Academy and Society (WSEAS) Stevens Point, Wisconsin, USA., pp: 27-30.

Zhiwen, W., L. Ke, Z. Huafeng and X. Qin, 2010. Component availability based survivability recovery in information system. Proceedings of the 5th International Conference on Frontier of Computer Science and Technology (CST' 10).

Zhou, Y., 2010. A visual modeling tool for the development of trustworthy component-based systems. Concordia University.