 doi:10.20944/preprints201608.0155.v1

Article

2DCBS: A Model for Developing Dependable Component-Based Software

Hasan Kahtan [1,2,*], Nordin Abu Bakar [3], Rosmawati Nordin[3] and Mansoor Abdullateef Abdulgabber [1]

[1] Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, 26300 Gambang Kuantan Pahang, Malaysia ; hakmansoor@ump.edu.my
[2] Institute of Visual Informatics, Universiti Kebangsaan Malaysia 43600 Bangi Selangor, Malaysia
[3] Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450, Shah Alam, Selangor, Malaysia ; nordin@tmsk.uitm.edu.my; roswati@tmsk.uitm.edu.my
* Correspondence: hasankahtan@ump.edu.my; Tel.: +609-5492436

Abstract: The software industry has adopted component-based software development (CBSD) to rapidly build and deploy large and complex software systems with significant savings at minimal engineering effort, cost, and time. However, CBSD encounters issues on security trust, mainly with respect to dependability attributes. A system is considered dependable when it can produce the outputs for which it was designed with no adverse effect on its intended environment. Dependability consists of several attributes that imply availability, confidentiality, integrity, reliability, safety, and maintainability. Dependability attributes must be embedded in a CBSD model to develop dependable component software. Motivated by the importance of these attributes, this paper pursues two objectives: to design a model for developing a dependable system that mitigates the vulnerabilities of software components, and to evaluate the proposed model. The model proposed in this study is labelled as developing dependable component-based software (2DCBS). To develop this model, the CBSD architectural phases and processes must be framed and the six dependability attributes embedded according to the best practice method. The expert opinion approach was applied to evaluate 2DCBS framing. In addition, the 2DCBS model was applied to the development of an information communication technology (ICT) portal through an empirical study method. Vulnerability assessment tools (VATs) were employed to verify the dependability attributes of the developed ICT portal. Results show that the 2DCBS model can be adopted to develop web application systems and to mitigate the vulnerabilities of the developed systems. This study contributes to CBSD and facilitates the specification and evaluation of dependability attributes throughout model development. Furthermore, the reliability of the dependable model can increase confidence in the use of CBSD for industries.

Keywords: component-based software development; dependability attributes; availability; reliability; integrity; confidentiality; safety; maintainability

1. Introduction

Component-based software development (CBSD) is an emergent technology that focuses on system construction by integrating existing software components. CBSD shifts the development emphasis from programming software to composing software systems by incorporating existing software components based on assumptions that certain parts of a large software system reappear regularly. Moreover, common parts may be written once and then reused many times rather than be written over and over again [1]. At the same time, CBSD offers a range of benefits, from enhancing an individual programmer's productivity to analyzing the costs of the developed software effectively [2-5]. The widespread and systematic reuse of software can meet the demands for accelerated delivery, reduced software production and maintenance costs, and improved quality. Hence, the main objective of CBSD is to lower the overall cost of software development [6,7]. In other words, the cost of software production and maintenance must be reduced. CBSD also allows for the faster

delivery of a software product [8,9]. Therefore, the software industry has adopted CBSD to rapidly build and deploy large and complex software systems with significant savings at minimal engineering effort, cost, and time.

Software must meet a market window set by competitive organizations. High-quality software that meets the requirements of the process to be served with minimal failure and maximum security [10-14]. Nonetheless, several studies have reported different challenges in implementing CBSD. According to Moradian and Håkansson [15], the interdependencies among software components induce problems during the integration phase of software development. Therefore, the dependability attributes of software components must be considered and evaluated in the early stages of the CBSD cycle. In addition, pervasive computing raises major concerns regarding the capability of current development models to develop dependable systems [16]. CBSD is an approach to software engineering [16]; however, its capability to develop dependable software applications remains unknown.

Moreover, a component may be unable to fulfill the application requirements because components and applications follow different requirements and cycles [17]. First, changes (e.g., modifying a few components or updating new versions) in the application level may induce system failure [18]. Second, reusing defective components may undermine trust in the entire software system. Therefore, critical systems such as the military system must adopt an exceptional software development process in place of the conventional approach [19]. The main objective of this process is to ensure the accuracy of system functionality and design to validate the consistency of system implementation with the set requirements.

In conclusion, CBSD still lacks essential formal foundations for the specification, composition, and verification of nonfunctional requirements despite the wide adoption of this development in the software industry and the significant number of academic studies conducted on this topic. As a result, current CBSD practices do not provide the essential requirements for developing dependable systems. To develop a dependable and secure system, dependability attributes must be embedded in the CBSD process. A system is considered dependable when it can produce the outputs for which it was designed with no adverse effect on the intended environment. Dependability comprises several attributes, namely, availability, confidentiality, integrity, reliability, safety, and maintainability.

Nonetheless, the task of embedding dependability attributes into the software component development process is less challenging than the task of evaluating the dependability of these attributes [20] because the requirements of dependability attributes must be specified during the early stages of software component development, along with the complex nature of the operational environment itself. A well-established scale that can measure the dependability of a software component remains difficult to establish in the research community [21]. Dependability attributes such as reliability, safety, and integrity are traditionally treated as afterthoughts against which protection mechanisms are employed following software component development [22,23]. Hence, component-based software must be evaluated based on dependability attributes because this assessment is critical in determining the dependability of a system.

The current study designs a model for developing dependable, component-based software that mitigates the vulnerabilities of web application systems. The model proposed in this paper is known as developing dependable component-based software (2DCBS). To develop this model, the CBSD architectural phases and processes must be framed and the six dependability attributes embedded. The developed 2DCBS model is then applied to the development of web application systems. This study also evaluates the developed 2DCBS model on the basis of dependability attributes.

The remainder of this paper is organized as follows: Section 2 reviews related studies on component-based models. Section 3 explains the motivation for this study. Section 4 describes the methodology. Section 5 elaborates on the model design process. Section 6 presents the 2DCBS model. Section 7 highlights the empirical study conducted on the proposed model. Section 8 discusses the evaluation of this model. Section 9 presents an overall discussion. Finally, Section 10 provides the conclusion.

2. Related Work

This section reviews relevant literature on the basis of two aspects: CBSD approach and CBSD models.

2.1 CBSD Approach

The CBSD approach emerged in the late 1990s, when reusable components were incorporated into development processes. The component base is the basic element of the current software system [24], and CBSD is a technique that uses existing software codes [25]. With this technique, software applications need not be developed from scratch. This technique also facilitates the assembly of software applications using reusable software codes, thereby improving time and budget constraints on software development. CBSD is widely used by middleware platforms and tools and has become the mainstream for current software development. CBSD design for distributed networks (including the Internet) has promoted e-commerce and may expand business markets considerably. Furthermore, CBSD utilizes software components that are easier to produce and more pervasive than ever before. Table 1 presents sample CBSD technologies and standards.

Table 1: CBSD Technologies and Models

CBSD Technologies and Models	Description
AUTOSAR [26]	This model utilizes standardized architecture to provide a method, by which to improve flexibility, scalability and quality of vehicular embedded systems as well as to improve the management of such complicated systems.
BIP [27]	This model utilizes a framework that incorporates heterogeneous real-time components (untimed or timed and synchronous or asynchronous).
COM [28]	This platform supports the communications between components. Developers employ COM to address specific areas that include controls, compound documents, data transfer, automation and storage.
SaveCCM [29]	This model is used in embedded control applications found in vehicular systems. Such applications include safety-critical subsystems (i.e., steering, brakes and power-train), which are responsible for controlling vehicle dynamics.
ProCom [30]	This model is designed to include the entire development process in the vehicular automation and telecommunication domains. It is utilized for distributed embedded systems that are control-intensive.
Koala [31]	This model has a specialized architectural description language and component that particularly target embedded software such as consumer electronics. By using and reusing software components within a specified type of software architecture, Koala is able to manage the complicated embedded software at an increased production speed.
EJB [32]	This model is a server-side component that covers the business logic of an application, including interoperability, concurrency, security, persistence and transactions.
CORBA [33]	This component model is utilized as middleware to minimize the effort required to initiate CORBA application development and deployment.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

Unlike traditional software development approaches, CBSD offers a range of benefits and manages complex systems [34,35]. Software systems consist of a series of components into which software functions and nonfunctional properties are implemented separately. CBSD promotes the employment of effective specialists who can develop reusable components within the scope of their expertise instead of application specialists who perform the same type of work on different projects [36,37]. The approach also reduces the time and effort needed to develop software [38,39]. Moreover, CBSD facilitates the development of components that are independent of specific applications and improves the reusability of components [34,40]. Software system developers can thus maximize existing structures and components, thereby improving the efficiency of software development [41,42]. CBSD also generates a repository of components that supports software system development by providing reusable and tested components.

CBSD likewise increases the productivity of programmers [5,11]. Constantly rewriting codes is an inefficient process because programmers can write and document only limited lines of code per day. With CBSD, programmers can utilize the interactive development environment (IDE) to assemble components in the desired program. Therefore, many lines of code can be written each day and productivity is enhanced [10,43]. Owing to the significant number of economic benefits gained, CBSD is an ideal approach to building software systems. Table 2 presents the application of CBSD in different domains.

Table 2: CBSD Applied to Different Domains

Domains	Description
Cloud Computing [44]	This domain is an adaptable component-based middleware introduced by the authors. Cloud computing offers an extensive solution to executing non-trivial communication applications in multi-domain platforms. It also allows the transition of non-trivial applications in traditional grids to hybrid grid-cloud platforms.
Embedded Systems Software at Run-Time [45]	CBSD is used as an architecture-base. Here, CBSD was used to effectively master deal system heterogeneity and software complexity as well as to manage evolution planning and execution.
Embedded System in Train Control Management System [46]	CBSD is utilized to improve a train control management system (TCMS) supplier organization through the real-time identification of reusable software from existing systems.
Safety-Critical System in the Automotive [47]	CBSD is used to elaborate on the effects of functional safety-critical systems (e.g., electronic systems in the automotive domain) as well as to develop product variants that are critical to ensuring safety.
Safety-Critical System and Dynamic Adaptation [48]	CBSD modeling is used to overcome the adaptation complexity of system design, which is brought about by numerous possible system configurations of a composition of reconfigurable components. This approach reduces the system design complexity at each hierarchical level by grouping the component compositions into a hierarchical component.
E-Business, Knowledge Management and E-Commerce [49]	The CBSD e-business model is applied together with knowledge management for application to the deterministic and semi-dynamic e-commerce environment. CBSD is utilized in the application of e-business through the subdomain of knowledge management.
Enterprise Distributed Real-Time and	The advantages of using component-based middleware are maximized in order to meet the quality-of-service (QoS) requirements of distributed real-time and embedded (DRE) systems as well as to support their

Embedded Systems [50]	implementation. Using component technologies, scalable and efficient standard-based deployment for component-based enterprise DRE systems can be achieved.
Medical Training Systems [51]	A 3D visual CBSD is applied in medical training systems to support haptic devices, such as phantoms. Developing medical training systems is easier and more efficient using the CBSD approach.
Hospital Pharmacy [52]	The CBSD approach is adopted in the health care domain. Hospital pharmacists found success in using this approach to model generic activities.

2.2 CBSD Models

Several authors have reviewed the current state of CBSD models. IrshadKhan, *et al.* [53], stated that different CBSD models have been developed for the industry and for the academe. In this study, the authors discussed and described development activities in five specific papers. Ahmed, *et al.* [25] compared the main drawbacks of selected models in terms of the CBSD process, whereas Pandeya and Tripathi [54] compared their proposed process model with several CBSD models with respect to development features. Chhillar and Kajla [55] and Kaur and Singh [56] studied and compared the most common models of CBSD, which emphasize reuse-based and feedback processes in each phase of software development. To support the development phase of the model proposed in the study, Sharp and Ryan [57] cited selected previous studies in the CBSD field. In addition, Aris and Salim [10] compared the development stages of seven existing CBSD models. Olsen and Loe [58] also investigated and described the advantages and disadvantages of six existing CBSD models.

The general concepts and integration efforts associated with the existing models have been described sufficiently by previous studies. However, these studies considered only a limited number of the models derived from literature. For instance, most authors manually selected the papers for review and narrowed the scope to less than eight models. In fact, the majority of the selected models was not discussed comprehensively. Furthermore, these studies also disregarded the issue pertaining to details regarding the phases and stages of the development process.

A robust systematic literature review (SLR) was presented in our previous work for identifying all relevant research in [59]. Table 3 shows a comparison of existing models with respect to the SLR-based development process. We introduce 26 studies on existing CBSD models and compare them in terms of three main CBSD development phases, namely, the system requirement and qualification, component development, and system development. These phases are subdivided into 31 development stages. Models that support the indicated development stage are checked (√). Process models that do not support the indicated development stage are not checked.

Many CBSD models have been proposed, as shown in Table 3. However, such models do not address the security domain because the security features of existing CBSD models are commonly considered post-development.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

Table 3: Gap Analysis in the Existing CBSD Models

Phases	Development Stages / CBSD Models	Brown et al. [60]	Aoyama [61]	Tran [62]	Lee, et al. [63]	Yau and Dong [64]	Cheesman, et al. [65]	Paul [66]	Crnković [67]	Hutchinson, et al.[68]	Capretz [69]	Mei [70]	Capretz [71]	Crnkovic, *et al.* [72]	Aris and Salim [10]	Qureshi et al. [73]	Kouroshfar, et al. [74]	Sharp and Ryan [57]	Gill and Tomar [1]	Bose [75]	Chhillar et al. [55]	Lau, et al. [76]	Pandeya et al. [54]	Shang et al. [5]	Sommerville [16]	Ahmed et al. [25]	IrshadKhan et al. [53]
System Requirement & Qualification	Domain engineering												√		√				√	√		√		√			√
	Domain analysis and specifications		√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Defining component qualifications required						√	√		√					√		√		√		√				√	√	√
	Obtaining qualified components from(repository / 3rd party vendor)	√	√	√			√	√	√	√	√		√	√	√	√	√	√	√	√	√		√	√	√	√	√
	Security consideration (requirement analysis & component selection)																										
	Determining qualified component development processes			√		√	√						√	√	√		√	√									
Component Development	For reuse (from scratch)																										
	Component requirements analysis				√	√				√	√	√	√	√	√	√	√	√	√	√	√		√	√	√	√	√
	Component design				√	√	√	√		√	√	√	√	√	√	√	√	√	√	√	√	√	√	√		√	√
	Design, architecture, implementing and testing based on security features																										
	Component implementation							√							√		√	√		√	√	√	√	√		√	√
	Component testing				√	√							√	√	√	√	√	√	√	√	√	√	√	√		√	√
	Component maintenance										√		√	√	√	√	√	√	√								√
	Component coding, wrapping and archiving				√	√		√					√				√	√	√		√	√					√
	Post-Modification																										
	Domain analysis and specifications												√		√				√								
	Neglecting the incompatible components	√	√	√					√	√	√	√		√	√										√		
	Evaluate & adaptability components according to architectural style		√							√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	
	Component coding, wrapping and archiving	√				√					√		√		√		√	√	√					√		√	
	Implementing and testing component-based on security features																										
	Component testing												√	√	√		√		√		√			√		√	√
	Without Modification																										
	Domain analysis and specifications							√					√						√		√						
	Obtaining qualified components compatible with system analysis, specification and design									√							√		√	√	√		√	√		√	
	Modifying application design to be suitable with existing components		√												√								√				
	Implementing and testing component-based on security features																										
	Component testing									√			√	√			√		√					√		√	
System Development	Assemble	√					√	√					√	√			√	√	√	√	√	√	√	√	√	√	√
	Testing components configured		√			√	√	√					√	√	√		√	√	√			√	√	√		√	√
	Testing the system after integrated all the components	√	√	√				√	√				√	√	√		√	√	√	√	√	√		√		√	√
	Implementation and deployment						√		√		√	√	√	√			√	√	√		√					√	√
	System maintenance								√				√	√			√	√	√	√	√						√
	Testing system based on security features																										
	Updating components after (system deployment / user feedback)	√									√		√	√	√		√	√	√		√			√		√	

Likewise, security issues are either neglected, included as an afterthought, or minimized because of the cost or efficiency conditions in the software development life cycle. The comparative analysis performed in [59] serves as the basis for the development of 2DCBS model in the current study. Furthermore, we thoroughly analyze existing research in [77] to investigate related software security attributes. Six dependability attributes are identified to address the lack of security issues in CBSD, namely, availability, reliability, confidentiality, integrity, safety and maintainability. When these dependability attributes are considered, the CBSD product is cured of security threats, abnormal behavior, and untrustworthy issues. Moreover, embedding dependability attributes can help CBSD developers unburden end users of security problems.

3. Motivation

The design of the 2DCBS model is motivated by our previous works: the analysis of CBSD gap in [59], the awareness survey conducted in [78], and the vulnerability assessment of selected web applications in [79].

The CBSD gap was analyzed presented in [59], in which we reviewed and summarized existing evidence concerning the challenges involved in applying CBSD, as well as the existing CBSD models in combination with their strengths and weaknesses. We thoroughly reviewed the literature based on an SLR. Several CBSD models have been proposed in literature as per this analysis. However, these models neglect the security features in the CBSD process.

We also presented the results of a survey of experts from the industry and from the academic community to determine the awareness of embedding security features into the CBSD process in [78]. CBSD was important in software production. However, numerous organizations did not fully consider the formal CBSD process for developing software systems. Moreover, the survey results indicated that security features were neglected during the life cycle process of industries. Therefore, a secure component must be developed for the CBSD process. Indeed, incorporating security activities into the software development life cycle is crucial to minimizing the number of security flaws and, consequently, reducing cost.

In Hasan Kahtan, *et al.* [79], we pre-assessed the vulnerability of selected web applications to motivate 2DCBS model development. This vulnerability assessment was conducted to investigate the effect of software development without considering dependability attributes. The vulnerability assessment results indicated that existing web application systems are highly vulnerable. Moreover, the vulnerabilities target the dependability attributes. Hence, disregarding dependability attributes can cause system functionality failure, suspension, or denial of service that result in poor system performance or system crash. Thus, dependability attributes must be verified and validated throughout the software development process to guarantee the dependability of web applications.

4. Methodology

The methodology of 2DCBS development is divided into three phases, namely, phase 1: identification of the problem and gap analysis, phase 2: model development, and phase 3: model evaluation. The flowchart of the development methodology, which is composed of associated activities and deliverables, is presented in Figure 1.

Phase 1 presents the preliminary study that investigates three main domains: CBSD, software security attributes, and evaluation methods. This investigation identified current CBSD issues, essential CBSD elements, CBSD phases and stages, common security features, dependability attributes, vulnerability lists, evaluation methods, and vulnerability assessments tools. Furthermore, the gap in CBSD models pertaining to security features was analyzed in current literature.

Information on the CBSD approach, software security attributes, and evaluation methods is gathered by obtaining related materials from books, theses, journals, and articles published in proceedings or in technical reports. To identify all relevant research papers, the literature is thoroughly searched using the SLR method. The information obtained facilitates a comprehensive

understanding of the concepts and models pertaining to the CBSD approach, software security attributes, and evaluation methods.

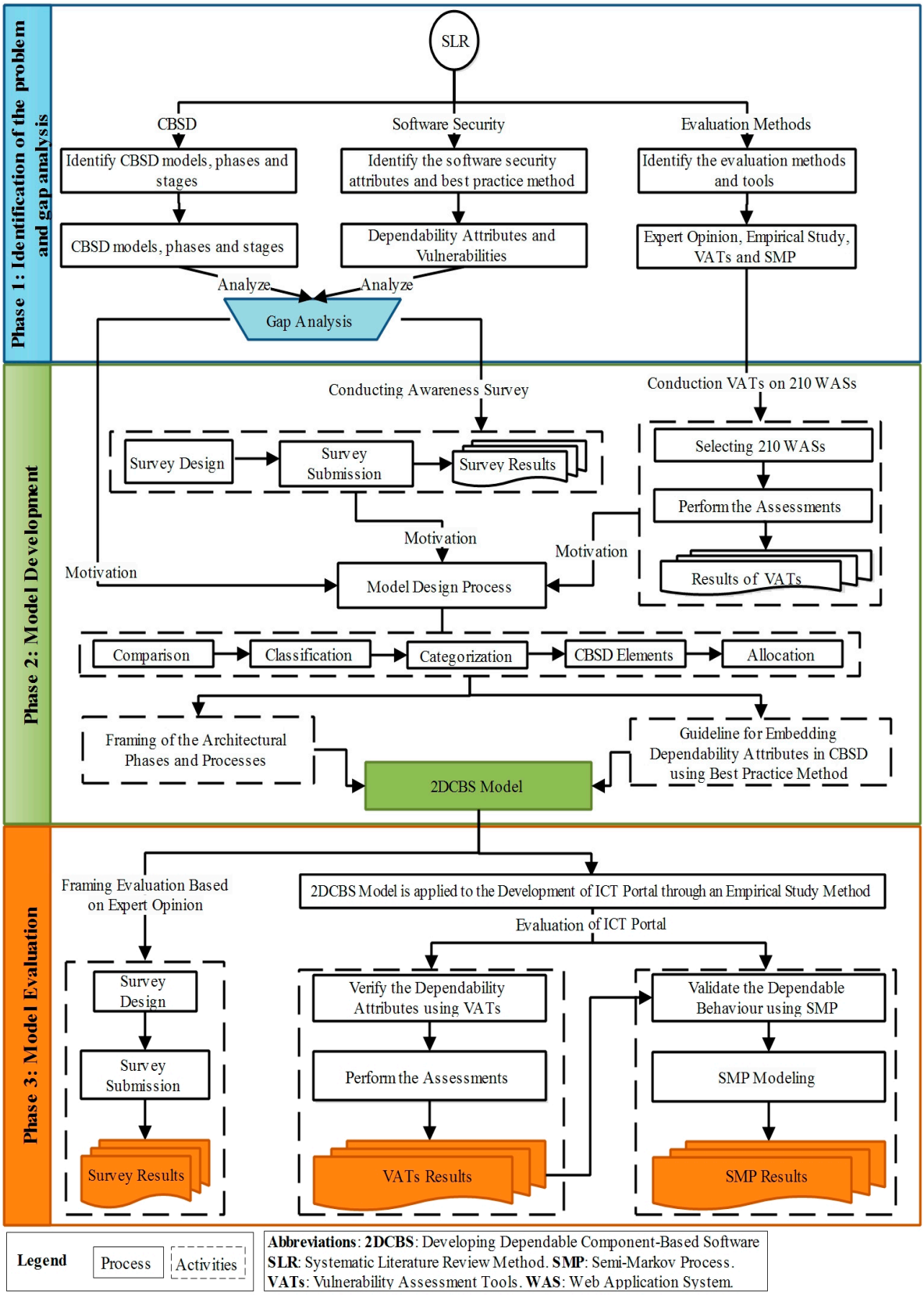

Figure 1: 2DCSB Development Methodology

To investigate the CBSD domain further, the existing CBSD models and issues are studied. The short-listed literature is then reviewed comprehensively to obtain the required information. The aims of the research are to understand the terminologies used in the CBSD realm, as well as the relationships of these terms with one another; to resolve confusion arising from the differences in terminologies; and to gain intensive knowledge of current CBSD development. Furthermore, the existing CBSD process models are studied comparatively to identify the features of each model. Each model is analyzed to determine its strengths and weaknesses. In addition, the gap in the existing CBSD models is analyzed. Strengths are retained and weaknesses are removed to derive the features to be incorporated into the 2DCBS process model. The results of the analysis are presented in our previous work [59].

In the software security domain, existing studies are analyzed thoroughly to investigate the related software security attributes that must be incorporated into the CBSD process to overcome security issues. Several attributes were used interchangeably to describe the properties of software security. The analysis findings show that dependability attributes are the solutions to security threats, abnormal behavior, and untrustworthy issues in a software system. Hence, the dependability attributes that can overcome the lack of security in CBSD are identified as follows: availability, reliability, confidentiality, integrity, safety, and maintainability. When these dependability attributes are considered, the CBSD product is protected against security vulnerabilities and threats. Moreover, embedding dependability attributes helps CBSD developers relieve end users of the burden of security problems. The results of the analyses are detailed in our previous work [77].

In the evaluation method domain, vulnerability assessment tools (VATs) are identified and utilized in the evaluation process to measure the dependability attributes. VATs systematically evaluate the networks used to recognize defiance of security and determine appropriate security measures. Such tools come in the form of security scanners that protect network security. The objective of VATs is to deliver efficient, thorough, and automated identification for detecting known vulnerabilities in the configuration of a specific operating system.

Phase 2 presents the development of the 2DCBS model. This phase aims to establish a model that can mitigate vulnerabilities in software components. A survey is conducted on awareness regarding the embedding of security features in the CBSD process on the basis of gap analysis and identification by VATs in phase 1. Vulnerability assessments are performed on 210 selected web application systems (WASs) to determine the record of WAS vulnerabilities. The main processes in the survey on awareness are survey design, survey submission, and survey analysis. Three processes are involved in the investigation of VATs in relation to the 210 selected WASs, namely, the selection of the WASs and the assessment and analysis of the results. These processes are presented in Figure 1. The outcomes of the awareness survey and VATs, as well as those of the gap analysis, motivate the design of the 2DCBS model in this study. In the 2DCBS design process, the following five sub-activities are considered: 1) comparison, 2) classification, 3) categorization, 4) CBSD elements, and 5) allocation. The activities are discussed in further detail in subsection 5.1 of this paper.

The final step in phase 2 is the development of 2DCBS model, which is facilitated based on: 1) the framing of the 2DCBS architectural phases and processes and 2) the embedding of dependability attributes in the CBSD process stages. The model design process is detailed further in section 5 of this paper.

Phase 3 discusses the evaluation of the 2DCBS model. The framing of 2DCBS is assessed on the basis of the interviews and the survey conducted with industrial experts. As shown in Figure 1, three activities are considered in the framing evaluations, namely, survey design, survey submission, and the analysis of survey result. The survey methodology, details, and the criteria used to select the expert participants are provided in our previous work [78]. A similar survey is conducted with industrial experts to determine awareness regarding the embedding of security features in the CBSD process. A similar method is adopted for the survey that evaluates the framing of the 2DCBS model. Twenty-five industrial experts served as the respondents. These participants are experts from in-

house software development company. The details and results of the survey are highlighted in subsection 8.1 of this paper.

An empirical study is conducted using a real-system test bed in consideration of industrial practicality to evaluate the model. This study applies the 2DCBS model to the development of an ICT portal. The empirical study is discussed in further detail in section 7 of this paper. Three activities are considered in model evaluation to verify the dependability attributes of the developed system; these activities are VAT configuration, VAT application, and the analysis of the VAT results. In Hasan Kahtan *et al.* [79], we also assessed vulnerability in a similar manner. The vulnerabilities in the developed WASs are evaluated in a similar manner. The assessments aim to identify the dependability attributes of the developed WASs and to verify the capability of the 2DCBS model to mitigate the vulnerabilities in the developed WASs. The results of these evaluations are presented in subsection 8.2 of this paper. In addition, the semi-Markov process (SMP) is considered as well in the evaluation of the dependable behavior of the developed ICT portal.

5. Model Design Process

The first step in deriving the 2DCBS model is to determine the elements and processes involved. As indicated in Figure 1, the process of deriving the 2DCBS model includes three main activities: 1) model design; 2) framing of the 2DCBS architectural phases and processes; and 3) implementation of guidelines for embedding dependability attributes in the CBSD process stages using best practice method. The following five sub-activities are considered in the model design process:

1. Comparison: Existing CBSD process models are compared by identifying the elements and processes of each model and by investigating their application processes. Each model is analyzed further to determine its strengths and weaknesses. These strengths and weaknesses serve as the foundation for framing the 2DCBS architectural phases and processes. Strengths are retained, and weaknesses are resolved. This study is summarized in a table for clarity. In this stage, the elements and the processes to be incorporated into the proposed 2DCBS process model are derived.
2. Classification: The processes of existing models are classified by the compartmentalization method for the CBSD phases. Three fundamental CBSD phases were considered in this procedure, namely, system requirement and qualification, component development, and system development.
3. Categorization: These processes are categorized further according to their descriptions. Despite being labelled differently, some processes describe the same activities. Therefore, these differences and similarities must be resolved before processes can be identified. Processes with similar sets of activities are noted, and the names of different processes are replaced with a label that reflects the set of activities.
4. CBSD Elements: The following elements are defined based on the analyses conducted in the second and third sub-activities:

    - Stages that comprise the CBSD processes and reusability features;
    - Compartmentalization method of the architectural phases; and
    - The method of iterative and incremental integration.

    The architectural phases and processes of the proposed model are thus framed.

5. Allocation: As per the finalized framing of the architectural phases and processes, the dependability attributes are embedded in the proposed model process in the following steps:

- Emphasis on dependability attributes in requirement analysis and component selection;
- Design, architecture, implementation, and testing based on dependability attributes;
- Component implementation and testing based on dependability attributes; and
- System testing based on dependability attributes.

The aforementioned sub-activities comprise the process of embedding the dependability attributes in the CBSD process. Hence, two main elements are combined to design the 2DCBS model: the framing of the CBSD architectural phases and processes and the embedding of the dependability attributes.

5.1 Framing of the Architectural Phases and Processes

The 2DCBS architectural phases and processes for framing are mainly drawn from existing CBSD process models; strengths are retained, whereas weaknesses are improved. The first step taken in framing the 2DCBS model is to identify the architectural phases and processes should be included in the model. Figure 2 presents the framing of these phases and processes. Processes in the existing models are grouped into architectural phase compartmentalization, iterative and incremental integration, and reusability preservation, as described in the legend box in Figure 2.

5.1.1 Architectural Phase Compartmentalization

All traditional software life cycle models are sequential. Thus, each phase must be completed before the next begins. To apply CBSD successfully, the architectural phases must be compartmentalized in the model to solve the sequential structure issue. Compartmentalization facilitates the parallel performance of several activities without requiring the stringent completion of all the requirements of one activity before another activity can be initiated. Thus, architectural phase compartmentalization helps developers reduce development time and resources. This process is divided into three phases: system requirement and qualification, component development, and system development phases.

1. *System Requirement and Qualification Phase:* A set of software components that can be applied to existing and future software system domains is identified, constructed, cataloged, and disseminated. To identify common areas and methods of describing the system by applying requirement standards, system requirements and qualifications are considered in the analysis of the system domain. Therefore, this phase should be initiated at the start of software specification if system reusability is considered. This phase enables software engineers to share and reuse software components while working on new and existing systems.
2. *Component Development Phase:* This phase is further compartmentalized into three subphases. Once the system requirements and qualifications are established, the component development team decides on the required component to be used (e.g., components that already exist in the organization's repository or components that can be bought) and the components that must be developed from scratch for possible reuse. These subphases are subject to mini life cycles. Therefore, three teams work in parallel on the associated subphases.

    2.1 Development for Reuse: The development process of a new component commences with the definition of the component interface. This definition represents a fixed mechanism among components. Components can be designed and implemented when the interface is developed and the objectives of the methods are set

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

2.2 Development without Modification: To redeploy previous software development projects, a component can be reused or migrated into a specialized subclass of an existing component that was created by a programmer.

2.3 Post-Modification Development: Building a new module from scratch is always avoided in component-based development. Existing components may require either minor or major modifications to adapt to other components. For example, the component interface may not conform to requirements or some methods may require modifications. Such modifications can be achieved by adaptation, which involves appending the component with a thin layer of code that implements the required changes.

3. *System Development Phase:* In this phase, the components developed by the component development team are integrated into an architectural style and interconnected with an appropriate infrastructure to facilitate the effective coordination and management of the component.

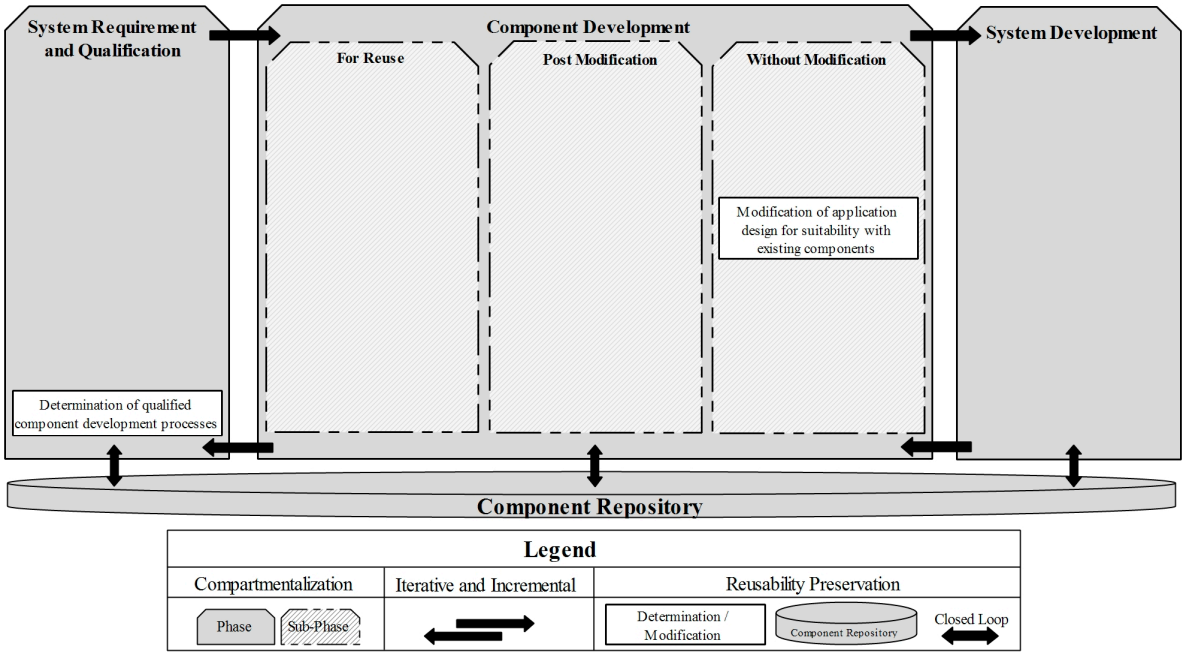

Figure 2: Framing of 2DCBS Architectural Phases and Processes

5.1.2 Iterative and Incremental Integration

Software component development often experiences uncertainties with respect to requirements and implementation approaches. The inclusion of different parties before these uncertainties are resolved complicates the development process further. In these situations, parties receive unclear specifications at the start of the development process. Thus, collaboration between parties is required throughout the course of the project. Glitches occur because practices and processes require long distance collaboration. The use of iterative and incremental development (IID) as a process model is necessary in software development that is fraught with uncertainties and unpredictable changes. IID is a system developed through iterations and the incremental addition of new features. This system is suitable for distributed development and reduces distribution problems through its rapid reaction to changes.

5.1.3 Reusability Preservation

A problem that discourages component reuse is the lack of specifications that allow programmers to anticipate component reuse in the development process. This problem can be

overcome by providing a specific location in the model that forces developers to consider the reusability feature of a component in the development process. The following features of the 2DCBS model preserve reusability:

1. *Determination of qualified component development processes:* The component development phase begins once the system requirement and qualification phase is complete. The team (development without modification) executes the task if the component is either available in-house or is acquired from a third party and can be applied. If the chosen component requires modification, the task is assigned to another team (post-modification development). If no component is available, the other team (development for reuse) executes this task.
2. *Modification of application design for suitability with existing components:* In certain situations, modifying the application design is necessary to suit the available components. The intent of such modification is to reduce the cost of developing components from scratch. Modification includes the customization of application design based on the components acquired when tailoring or modifying the components is costly or time consuming.
3. *Reusable Library (Repository):* To apply CBSD successfully, the deposition of components into the component repository must be demonstrated explicitly. To improve component based software productivity, reusable components should be selected. The repository stores and manages reusable components. The main benefits of working on reusable components with a repository include classification, searching, modification, testing, implementation, version control, change control, and current and consistent documentation.
4. *Closed Loop:* In the closed-loop model, components from the previous development cycle are explicitly fed back to the model to populate the repository. CBSD emphasizes the reuse of components from previous development life cycles.
5. *Traceability:* Each stage in each phase should be demonstrated in a clear and sequential process to help developers. Moreover, each process should include detailed explanations and adequate examples to guide developers in applying this model. Unimportant processes are omitted to avoid confusion.

5.2 Guideline for Embedding Dependability Attributes

On the basis of our previous work in [77], we conclude that dependability attributes should be considered to overcome the security problems caused by poor software development in current WASs. Dependability attributes must be embedded in the CBSD process to address the drawbacks of current CBSD practices. Furthermore, the vulnerabilities associated with dependability attributes should be identified to construct dependable software components. Thus, we introduce a guideline for embedding dependability attributes in the CBSD process using the best practice method [80]. This method trains employees to ensure that they fully understand their responsibilities in implementing security rules. The guideline is designed with the assistance of expert software developers and security consultants from a local company in Malaysia. This guideline consists of a set of best practices designed to embed dependability attributes in the CBSD process. The objective of the guideline is to demonstrate the embedding of such attributes in the four phases of the CBSD process, namely, requirements, design, implementation, and testing. Figure 3 presents the dependability attributes embedded in the CBSD process. The guideline is detailed in [80].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

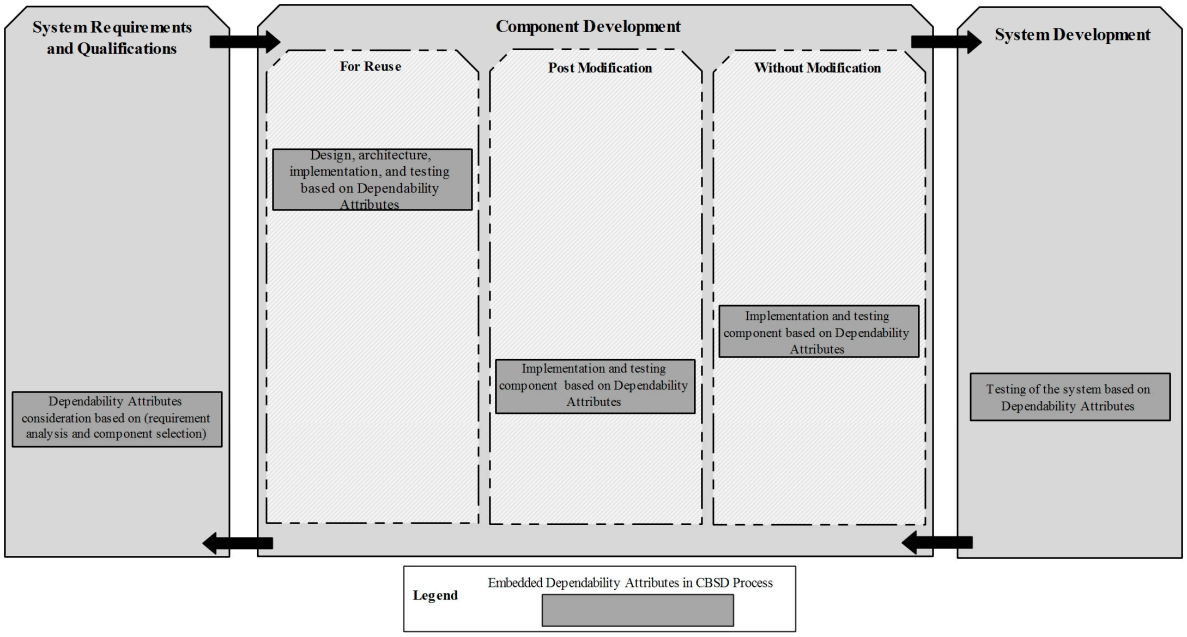

Figure 3: Embedding Dependability Attributes in CBSD Process

6. 2DCBS Model

As a result of the findings of model design process, the 2DCBS model is proposed as shown in Figure 4. The proposed model is divided into three phases: system requirement and qualification, component development, and system development.

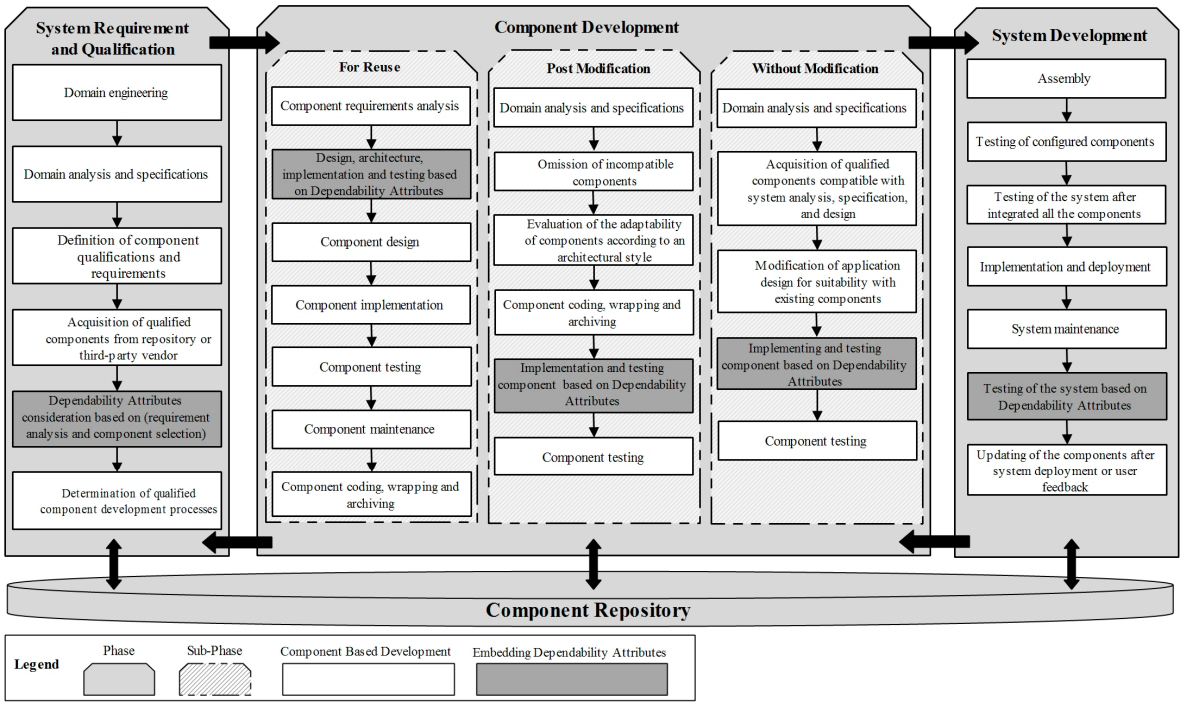

Figure 4: 2DCBS Model

In the system requirement and qualification phase, the system domain is analyzed by identifying common areas and methods to describe the system. Six stages are outlined, and the analysis of the dependability attributes in requirement and component selection is highlighted as an essential stage of this phase. The component development phase is also compartmentalized into three subphases, namely, the development for reuse, post-modification development, and development without

modification phases. Each of these three subphases consists of several stages. Design, architecture, implementation, and testing based on dependability attributes are integrated into these three subphases. Several stages are outlined in the system development phase. In this phase, the components developed by the component development team are integrated into an architectural style and interconnected with an appropriate infrastructure that facilitates the effective coordination and management of the component. The testing system based on the dependability attributes is incorporated.

The arrows in Figure 4 indicate that the three main phases are simultaneously connected to one another and to the model repository, where IID and reusability preservation are incorporated. In reusability preservation, components are deposited into the component repository.

7. Empirical Study

Empirical study is suited for many types of software engineering research because the objects of study are contemporary phenomena. These phenomena are difficult to study in isolation [81]. Theoretical and conceptual studies differ from analytical and controlled empirical studies and have therefore been criticized by researchers as being less valuable, being impossible to generalize, and being biased, among other reasons. These criticisms can be avoided by adopting proper research methodology practices and by realizing that knowledge should not be limited to statistical significance [81].

The rigorous implementation of a CBSD model requires application despite the actual demands of real software applications. Ideally, a CBSD model is applied to numerous systems; however, this ideal situation is not feasible. To address these problems, a CBSD model should be applied in an empirical study [81,82]. This study aims to construct an industrially feasible software application system using the CBSD approach. The model implementation process highlights industrial practicality to ensure that the dependability attributes of the software components are applied in an experimental context. Thus, developing a WAS using the CBSD approach is possible. The question is whether or not a model can significantly contribute to the resolution of the lack of security trust in a WAS using the CBSD approach.

Therefore, an empirical study is conducted on the 2DCBS model according to industrial practicality. This empirical study applies the 2DCBS model to develop an ICT portal. The ICT portal development that follows the 2DCBS model can ensure the proper integration of the dependability attributes and generalization of the results.

We collaborated with a local company in Malaysia for ICT portal development. Owing to the competition among software development companies, the company name is kept confidential for commercial reasons. We refer to the company as Software Development Company (SDC). The ICT portal was developed by a software development team that consists of six members currently working at SDC. SDC is a leader in ICT innovations in Malaysia and has facilitated new market creation for partners through patentable technologies for economic growth. With over 25 years of experience, SDC contributes its core technological competencies to the industry to raise Malaysia's local, regional, and international market competitiveness. The developed online portal provides various applications and related information to help users improve their lives in social community. Moreover, the online portal is also equipped with intelligent service delivery platform (ISDP) application to provide the community with access to useful information on science, technology, and innovation. The empirical study is detailed in our previous work in [83].

8. Evaluation

Evaluating the acceptability, usability, and reliability of a software development model requires the model to be applied and tested in an actual software development environment and thus demands much time and budget. Several CBSD models have been introduced in the literature; however, most of these models have not been evaluated because of limitations in time and budget. Moreover, the development of a well-established scale that can measure the dependability of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

software components remains a great challenge for the research community. Despite such challenges, the 2DCBS model is evaluated in this paper to verify whether or not the 2DCBS model can mitigate the vulnerabilities in the developed system. Two elements of the 2DCBS model are evaluated as follows:

1. The framing of 2DCBS architecture is evaluated by conducting an expert evaluation.
2. The dependability attributes embedded in the 2DCBS process are evaluated by verifying the dependability attributes of the developed ICT portal.

8.1 Expert Evaluation

The results obtained are based on the supplemental documents and explanations given by experts on the 2DCBS model during an interview. Figures 5 to 14 show the survey results of the 2DCBS framing evaluation. As per Figure 5, 44% of the experts strongly agreed and 50% agreed that the processes included in the 2DCBS model are essential to CBSD. This result provides a total of 94% validation for the model framing. Only 6% of the experts rated the 2DCBS framing as fair. Experts agreed that the processes included in the 2DCBS model are essential to CBSD for several reasons. First, the architectural phases of the 2DCBS model have been compartmentalized to solve the sequential structure issue. Second, including IID as a process in the 2DCBS model is necessary to ensure collaboration between parties during the project. Third, the 2DCBS model provides a specific location that forces developers to consider the reusability feature in the development process.

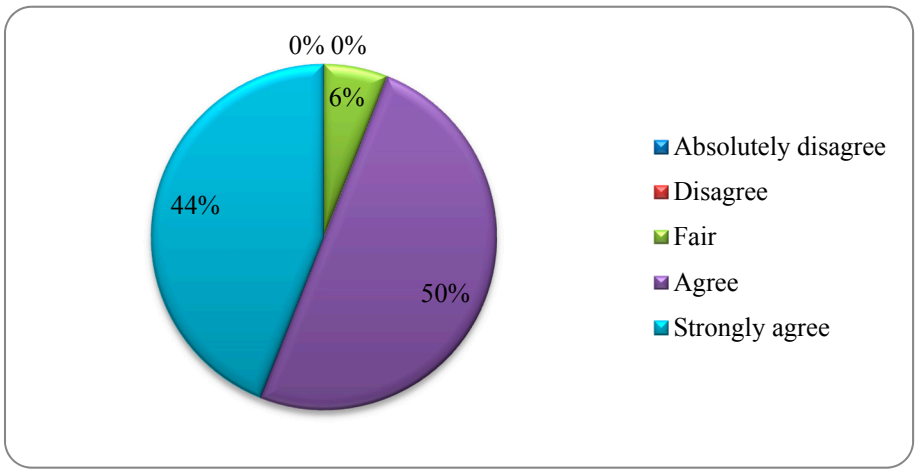

Figure 5: The 2DCBS Process are All Essential for CBSD Model

According to Figure 6, 42% of the experts strongly agreed with the statement that 2DCBS solves the sequential structure issue, 54% agreed with this statement, and 4% rated the statement as fair. Experts agreed that the 2DCBS solves the sequential structure issue because compartmentalization allows several activities to be performed in parallel without having to stringently complete the requirements of one activity before starting another activity. Figure 7 verifies that the 2DCBS model is easy to understand and simple to apply as proven that when 73% of the experts strongly agreed and 27% agreed with the statement. Experts agreed that the 2DCBS is easily understood and applied because of the provided traceability. Each stage in each phase is demonstrated in a clear and sequential process. Moreover, each process includes detailed explanations to guide developers in applying this model.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

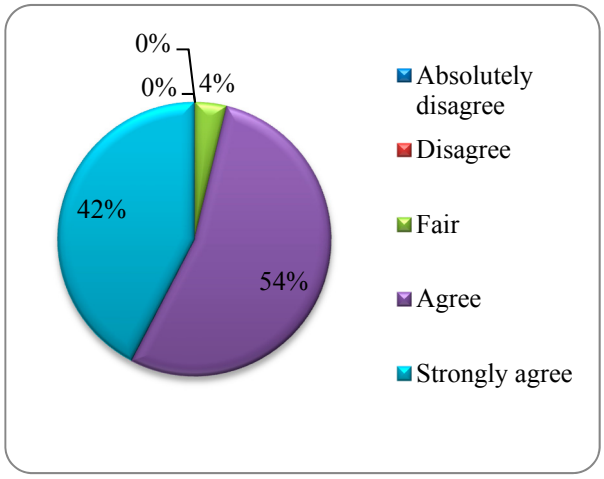

Figure 6: 2DCBS Solves the Sequential Structure Issue

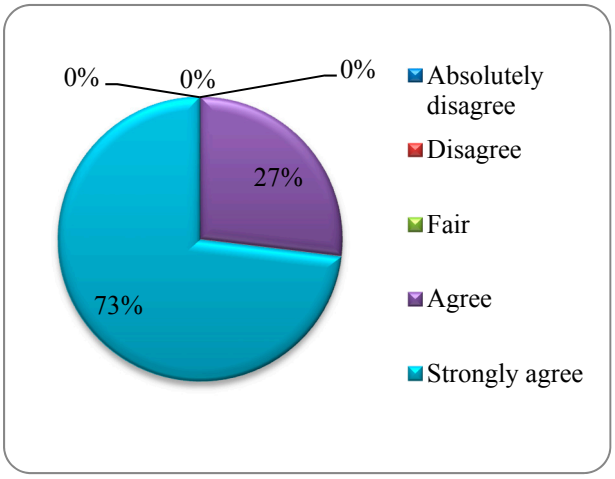

Figure 7: 2DCBS is Easy and Simple to be Applied

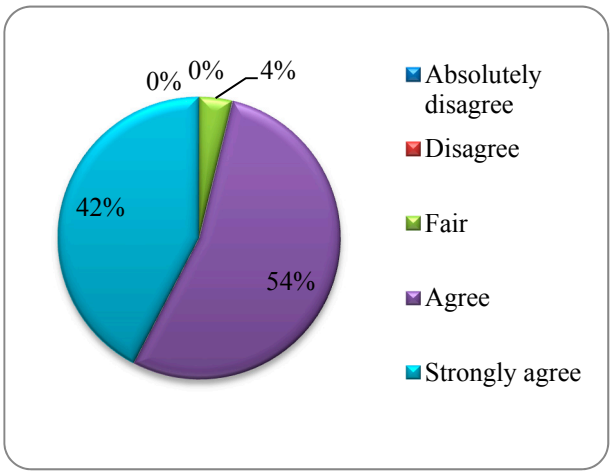

Figure 8: 2DCBS Consider A Place for the Reusable Components

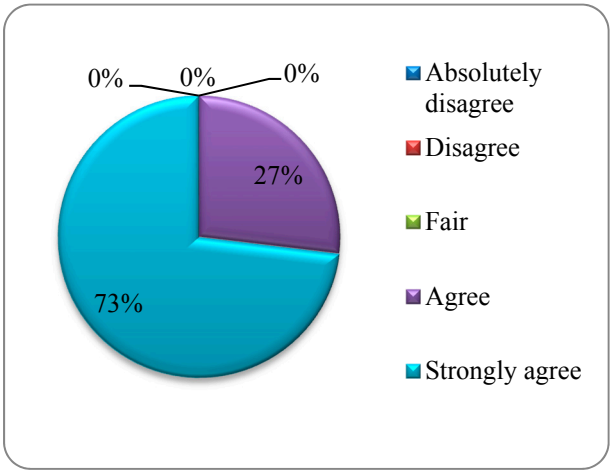

Figure 9: 2DCBS Improves the Component Development Phase

Moreover, 42% of the experts strongly agreed and 54% agreed with the statement that the 2DCBS considers the use of reusable components. Only 4% of the experts rated the statement as fair. This finding is reflected in Figure 8. Experts agreed that reusable components are considered in the 2DCBS because the deposition process of components into the component repository is essential to improve component based software productivity.

Figure 9 shows how the 2DCBS model improves the component development phase by three compartmentalized subphases to help developers reduce development time and resources in software production. Based on the result, 73% of the experts strongly agreed that the 2DCBS model has improved the component development phase, and 27% agreed. Figure 10 illustrate the benefits of architectural phase compartmentalization. Out of 26 experts, 25 experts highly agreed (combination of the experts who strongly agreed and agreed) with the benefits of architectural phase compartmentalization. The benefits of architectural phase compartmentalization are listed in Figure 10.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

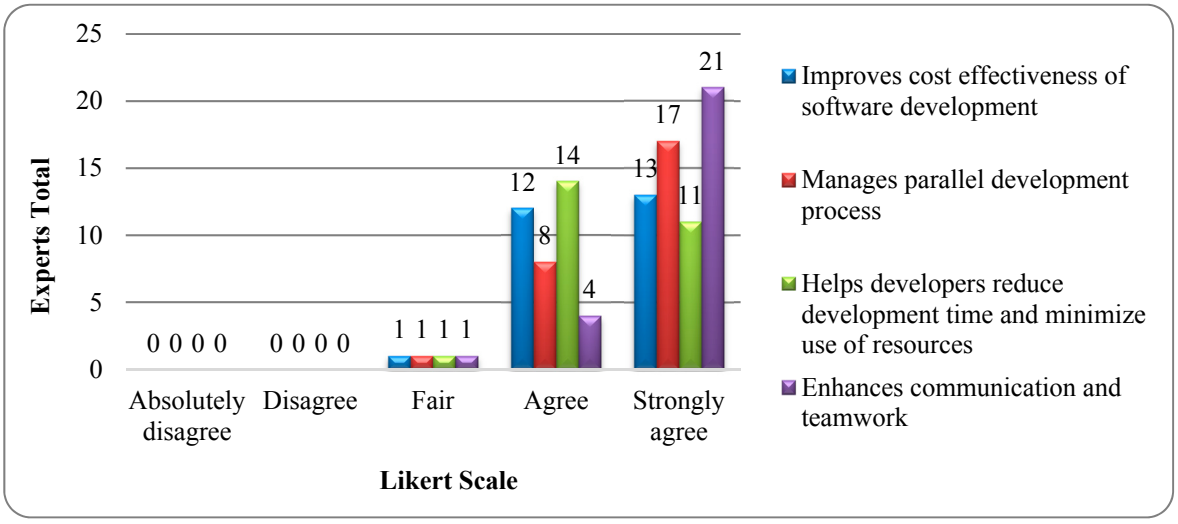

Figure 10: Benefits of Architecture Phase Compartmentalization

Figure 11 reveals the suitability of the use of IID as a process model. Out of the 26 experts, 25 highly agreed with the suitability of IID in the development process.

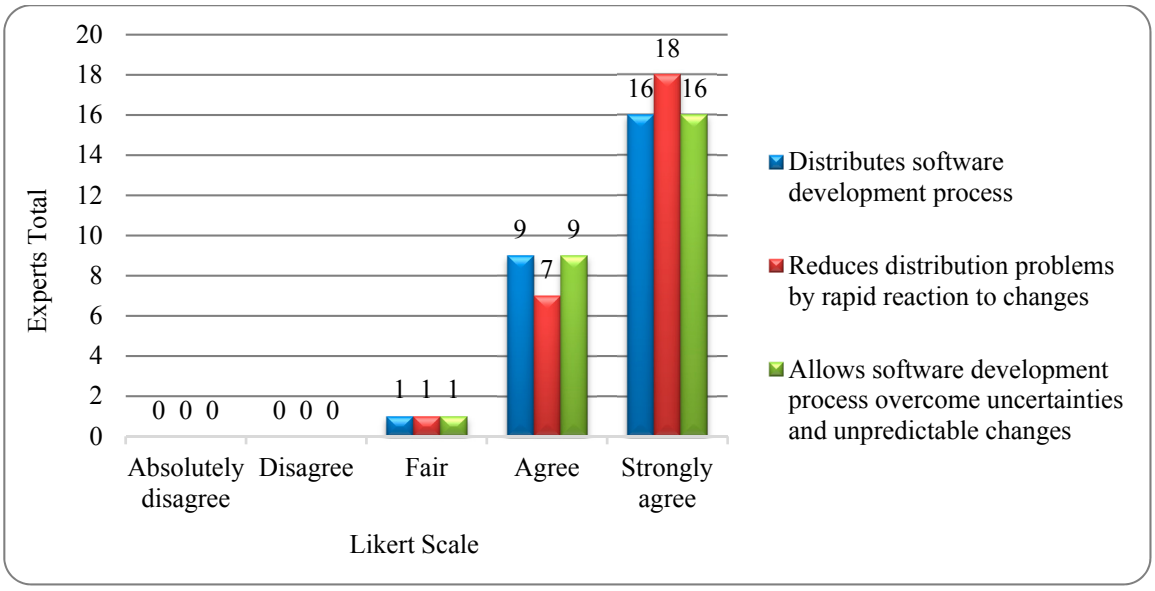

Figure 11: The Used of Iterative and Incremental Development (IID)

Moreover, Figure 12 shows that 58% of the experts strongly agreed that the 2DCBS provides traceability, 38% agreed, and only 4% rated this statement as fair. The survey also showed that 62% of the experts strongly agreed and 15% agreed that customization of application design based on the components helps reduce development cost. However, 23% of the experts rated this statement as fair. These survey results are presented in Figure 13. According to Figure 14, a total of 96% of the experts highly agreed that the closed-loop feature promotes the reuse of components from previous development life cycles. From the survey results, the phases and processes involved in the 2DCBS framing allows for short development time, increased productivity and product quality, and reusability of CBSD, thereby forecasting the successful application of CBSD.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

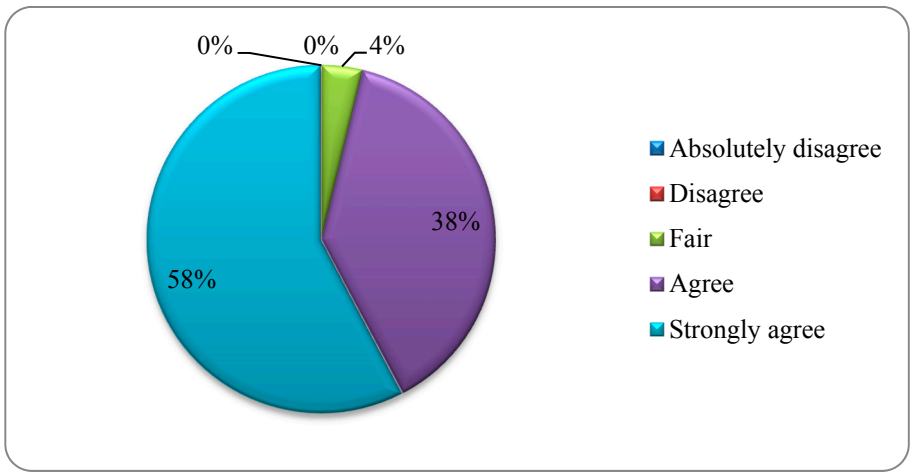

Figure 12: 2DCBS Provides Traceability

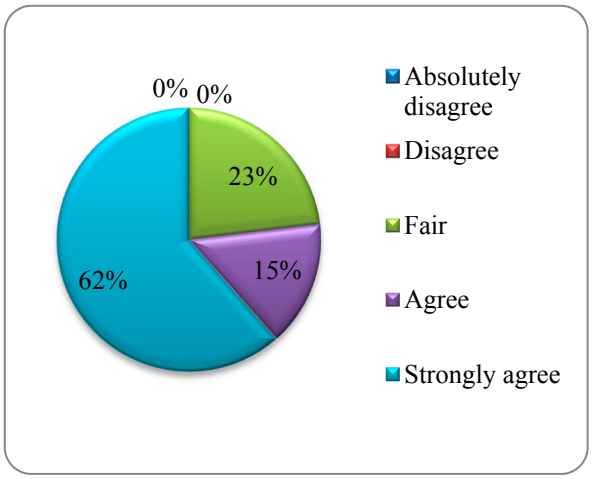

Figure 13: Cost Reduction Based on Customization of Application Design

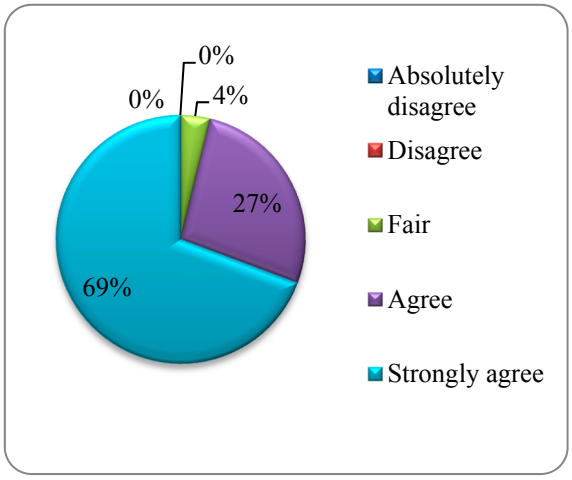

Figure 14: Closed-loop will Promote the Reusability

8.2 Verifying the Dependability Attributes

An empirical study on the proposed 2DCBS model was discussed in our previous work in [83]. This empirical study aimed to apply the proposed 2DCBS model to the development of an ICT portal. An evaluation of the developed ICT portal was carried out using VATs to verify the dependability attributes of the developed ICT portal. Two versions of the ICT portal were developed to evaluate the dependability attributes of the 2DCBS model. The first version was developed with the traditional CBSD model (not embedded with dependability attributes). This version is referred to as "traditional deployment" hereafter. The second version of the ICT portal was developed with the 2DCBS model with embedded dependability attributes. This version is referred to as "2DCBS deployment" hereafter. The evaluation of the ICT portal was performed based on the following two key dimensions:

1. 2DCBS deployment should mitigate failures better than traditional deployment.
2. 2DCBS deployment should mitigate vulnerabilities better than traditional deployment.

Different sets of experiments were performed to evaluate the ICT portal based on these two dimensions. Traditional deployment served as the reference in the comparison with the 2DCBS results from both cases. The comparison was performed to investigate the effects of the two deployment methods on the level of vulnerabilities in the system. The results obtained during JMeter,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

OpenVAS, and RATS (Rough Auditing Tool for Security) scanning of the ICT portal with different deployment methods are presented in the following subsections.

8.2.1 Apache JMeter Results

JMeter was used to measure the availability and reliability attributes of the ICT portal. The results were collected after implementing all the test plans. Figure 15 shows that before 100 threats, the values of Bar1 and Bar2 throughputs are close because only a few threats were loaded to the test bed at that period. From 150 threats to 200 threats, the average value of Bar1 is approximately 10% higher than that of Bar2. However, increasing the number of threats to more than 200 per second causes a rapid decrease in the value of Bar2, which in turn causes failure in the system services.

The decrease in Bar1 is not obvious, and the services remain active. This finding indicates that the service system with 2DCBS deployment is better than that with traditional deployment, and that the fluctuation in Bar1 is less sharp than that in Bar2. Thus, Bar1 operates more steadily in service systems. Figure 15 provides a summary of the availability and reliability results of 2DCBS deployment and traditional deployment. Increasing the number of requests per second causes a linear change in the observed availability and reliability. However, when the number of requests exceeds the threshold value of 175, a decline in availability and reliability begins. The decline in availability and reliability in the 2DCBS deployment is only 3.34%, demonstrating improved availability and reliability compared with traditional deployment. The results of the assessment show that 2DCBS deployment performs better than traditional deployment when the load of the system is increased gradually. The graph indicates that throughput for Bar1 is stable with 100 users to 200 users but unstable with 250 users, because the ICT portal is designed to support only 200 users concurrently.

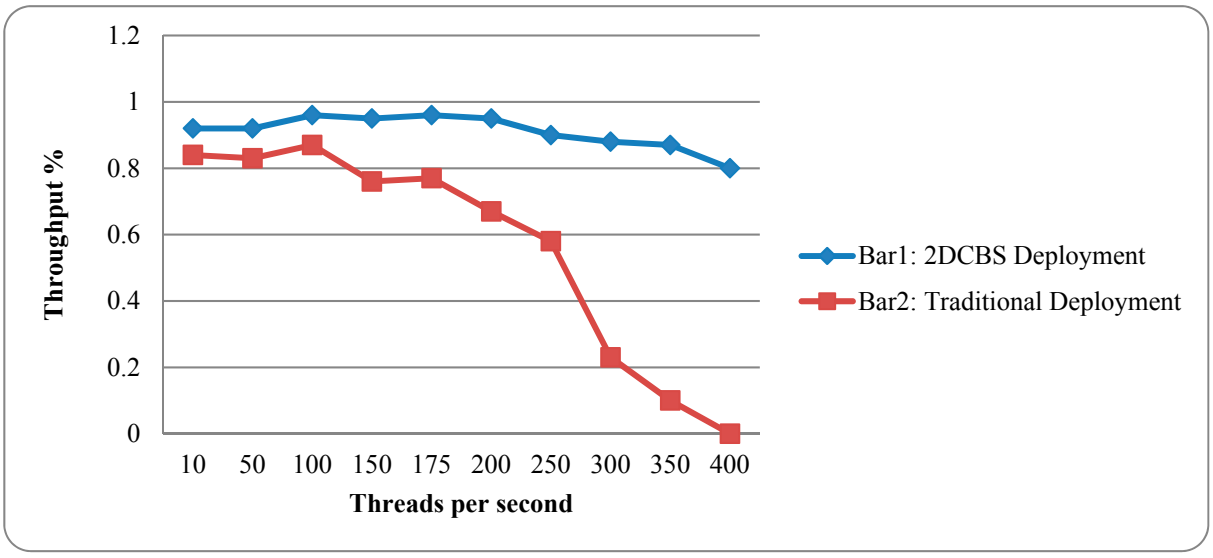

Figure 15: Availability and Reliability Comparison

8.2.2 OpenVAS Results

OpenVAS was used to scan traditional and 2DCBS deployments. Two separate scans were performed using the same configuration. The first scan was performed on traditional deployment, and the second scan was performed on 2DCBS deployment. Two separate reports were generated from the scans. These reports list the vulnerabilities detected in traditional and 2DCBS deployments. Each of Figures 16 to 18 presents the filtered results from each scan. The detected vulnerabilities were included based on their type and were linked to pertinent dependability attributes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

Figure 16 presents the comparison of confidentiality vulnerabilities. The numbers of vulnerabilities detected by OpenVAS under the high-risk factor are 24 for traditional deployment and 5 for 2DCBS deployment. Under the medium-risk factor, 12 vulnerabilities are detected for traditional deployment and 8 for 2DCBS deployment. Under the low-risk factor, 18 vulnerabilities are detected for traditional deployment and 12 for 2DCBS deployment.

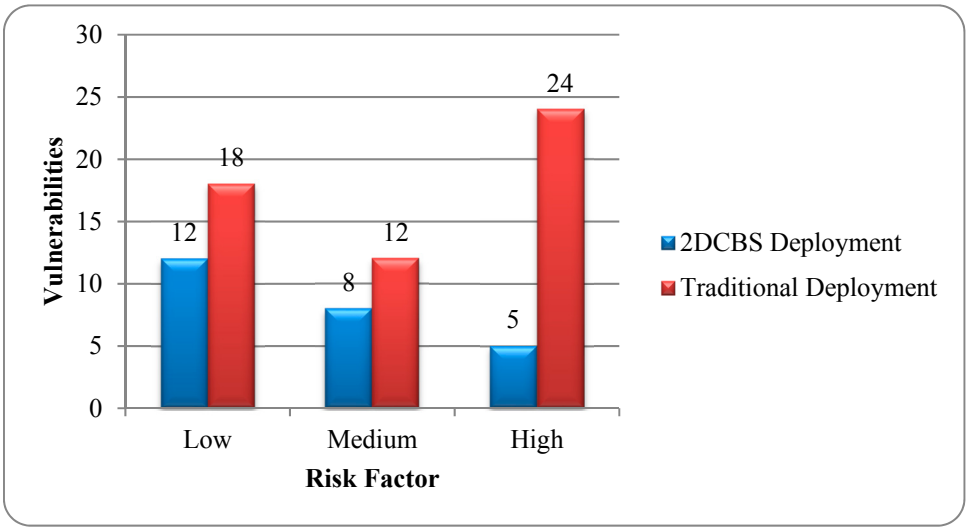

Figure 16: Confidentiality Vulnerabilities Comparison

Figure 17 presents the comparison of integrity vulnerabilities. The numbers of vulnerabilities detected by OpenVAS under the high-risk factor are 29 for traditional deployment and 8 for 2DCBS deployment. Under the medium-risk factor, 14 vulnerabilities are detected for traditional deployment and 12 for 2DCBS deployment. Under the low-risk factor, 26 vulnerabilities are detected for traditional deployment and 18 for 2DCBS deployment.

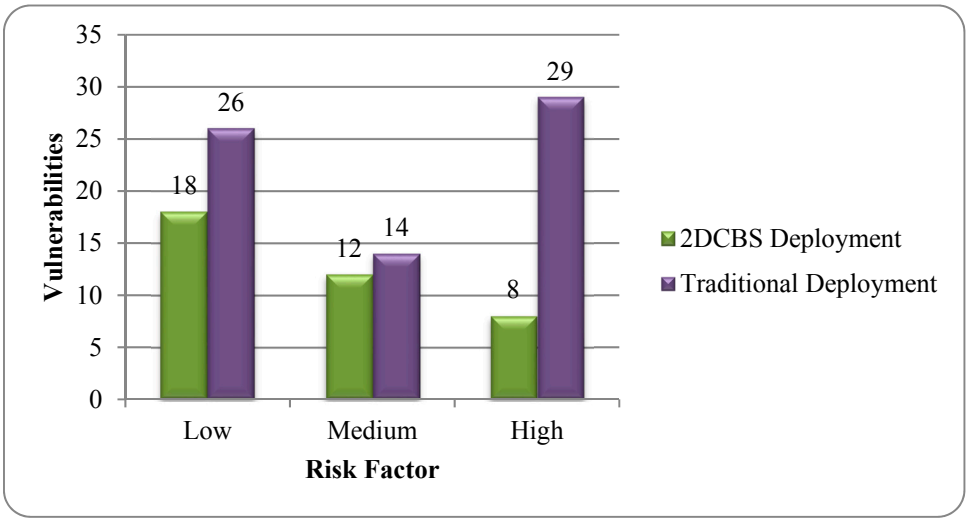

Figure 17: Integrity Vulnerabilities Comparison

Figure 18 presents the comparison of safety vulnerabilities. The numbers of vulnerabilities detected by OpenVAS under the high-risk factor are 34 for traditional deployment and 7 for 2DCBS deployment. Under the medium-risk factor, 26 vulnerabilities are detected for traditional deployment and 18 for 2DCBS deployment. Under the low-risk factor, 22 vulnerabilities are detected for traditional deployment and 14 for 2DCBS deployment.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

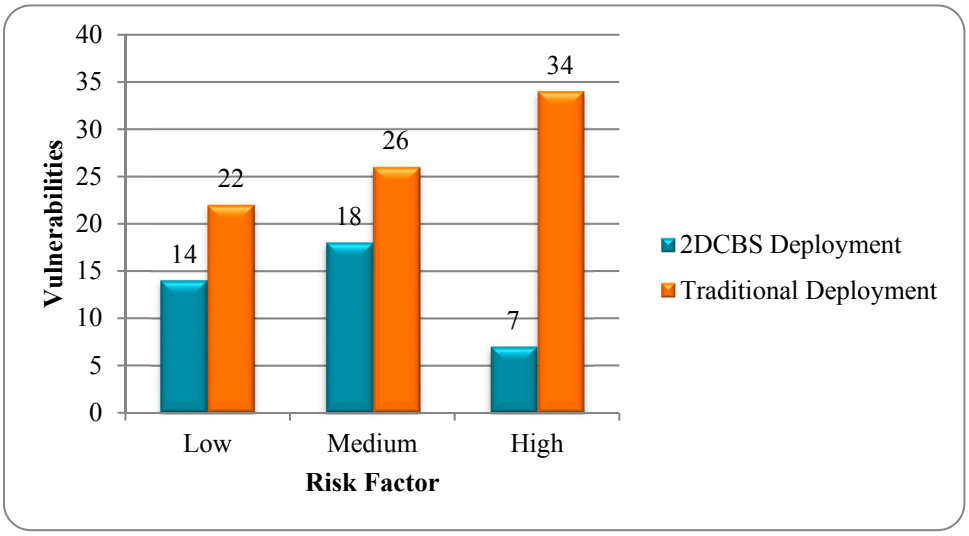

Figure 18: Safety Vulnerabilities Comparison

8.2.3 RATS Results

RATS was used to measure the maintainability attribute of the ICT portal. Figure 19 presents the comparison between traditional and 2DCBS deployments in terms of system maintainability. The comparison results show that 75% of the issues in traditional deployment source code were detected, whereas 25% of the issues in the 2DCBS deployment source code were detected. System maintainability is evidently improved when the 2DCBS model is adopted. Figure 19 also compares the maintainability vulnerabilities. The numbers of vulnerabilities detected by RATS under the high risk factor are 36 for traditional deployment and 8 for 2DCBS deployment. Under the medium-risk factor, 14 vulnerabilities are detected for traditional deployment and 12 for 2DCBS deployment. Under the low-risk factor, 26 vulnerabilities are detected for traditional deployment and 18 for 2DCBS deployment.

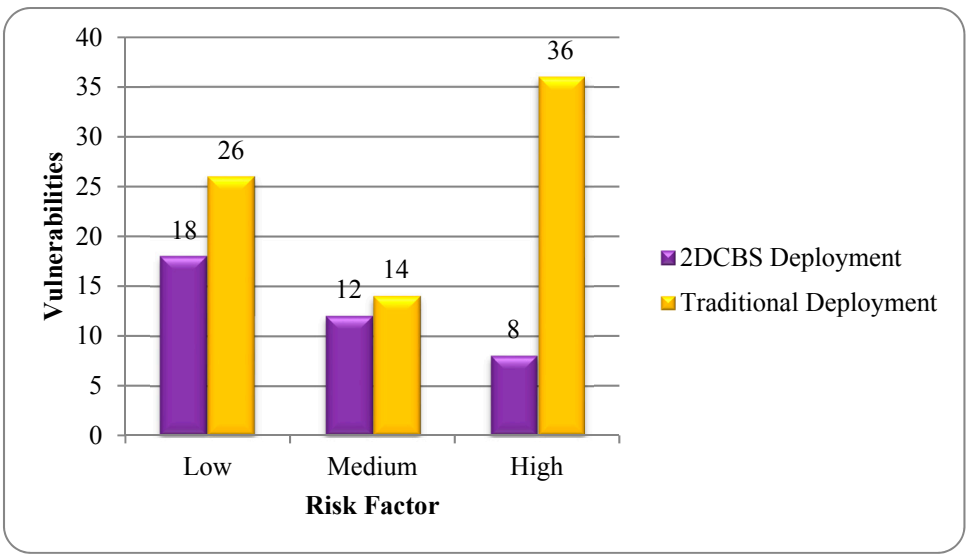

Figure 19: Maintainability Vulnerabilities Comparison

8.2.4 Summary of Verification Process

Traditional deployment served as the reference for comparison with the 2DCBS results. The results of the VATs on the dependability attributes of the ICT portal show that 2DCBS deployment

mitigates failures better than traditional deployment does. Moreover, 2DCBS deployment mitigates vulnerabilities better than traditional deployment does. The degradation in availability and reliability in the 2DCBS deployment is only 3.34%, demonstrating improved availability and reliability compared with traditional deployment. The assessment results show that 2DCBS deployment is more efficient than traditional deployment as system load gradually increases. In addition, the vulnerabilities of the systems are promptly tolerated and risks are reduced to a manageable level in 2DCBS deployment.

9. Discussion

We have surveyed the existing CBSD models from the literature and have indicated that each of the existing models has its own advantages and disadvantages. This work also compares the 2DCBS model with other available models to present the phases, stages, and features of the 2DCBS model. The key features that the 2DCBS model possesses but most other models lack are the following:

1. Demonstration of Embedding Dependability Attributes: The 2DCBS model demonstrates the process of embedding the six dependability attributes in the CBSD process. This model systematically guides software developers, designers, and engineers in building a dependable system. In addition, the model traces dependability attributes, requirements, design, implementation, and testing throughout the CBSD process, thereby enabling the model to assist managers and developers during the development of software systems.
2. Guideline for Embedding Dependability Attributes in CBSD Phases: The proposed model outlines a guideline based on the best practice method. The guideline was designed with the assistance of expert software developers and security consultants from a local company in Malaysia. This guideline consists of a set of best practices designed to embed dependability attributes in the CBSD process. The guideline includes the processes of eliciting and defining the requirements of dependability attributes by employing risk analysis and assessment approach.
3. Empirical Study: An empirical study on the 2DCBS model was carried out based on industrial practicality. This empirical study applies the 2DCBS model to the development of an ICT portal. Demonstrating the ICT portal development following the 2DCBS model can ensure the proper integration of the dependability attributes and generalization of the results.
4. Evaluations: Evaluations of the 2DCBS model were carried out in this study to verify that the 2DCBS model is capable of mitigating the vulnerabilities in the developed system. Two elements of the 2DCBS model were evaluated as follows: a) framing of the 2DCBS architecture was evaluated by conducting an expert evaluation and b) dependability attributes embedded in the 2DCBS process were evaluated by verifying the dependability attributes of the developed ICT portal.
5. Compartmentalization: The 2DCBS model solves the sequential structure issue by compartmentalizing the architectural phases. Compartmentalization allows several activities to be performed in parallel without having to stringently complete the requirements of one activity before starting with another activity. Thus, architectural phase compartmentalization helps developers reduce development time and resources.
6. Reusability: The 2DCBS model promotes component reuse by providing a specific location in the model that forces developers to consider the reusability feature in the development process.
7. IID: The 2DCBS model includes the use of IID method as a process in software development. IID is a system developed by iterations and incremental additions of new features. IID is suitable for distributed development and reduces distribution problems by its rapid reaction to changes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 August 2016 doi:10.20944/preprints201608.0155.v1

Table 4 highlights the comparison between 2DCBS model and the existing CBSD models based on aforementioned key features. Models that support the indicated key features are checked (√). The 2DCBS model is showing that it covers all of the listed key features.

Table 4: Comparison between 2DCBS Model and the Existing CBSD Models

Key Features / CBSD Models	Demonstration of Embedding Dependability Attributes	Guidelines for Composing Dependability Attributes in CBSD Phases	Empirical Study	Expert Evaluation	Dependability/Security Evaluation	Compartmentalization	Reusability	Iterative and Incremental Development (IID)
Brown et al. [60]							√	
Aoyama [61]							√	
Tran [62]							√	√
Lee, et al. [63]							√	√
Yau and Dong [64]							√	√
Cheesman, et al. [65]							√	√
Paul [66]							√	√
Crnković [67]							√	√
Hutchinson, et al.[68]							√	√
Capretz [69]							√	√
Mei [70]							√	√
Capretz [71]							√	√
Crnkovic, *et al.* [72]						√	√	√
Aris and Salim [10]			√	√		√	√	√
Qureshi et al. [73]						√	√	√
Kouroshfar, et al. [74]						√	√	√
Sharp and Ryan [57]						√	√	√
Gill and Tomar [1]						√	√	√
Bose [75]						√	√	√
Chhillar et al. [55]						√	√	√
Lau, et al. [76]						√	√	√
Pandeya et al. [54]						√	√	√
Shang et al. [5]						√	√	√
Sommerville [16]						√	√	√
Ahmed et al. [25]			√			√	√	√
IrshadKhan et al. [53]			√	√		√	√	√
2DCBS Model	**√**	**√**	**√**	**√**	**√**	**√**	**√**	**√**

10. Conclusion

The aims of all software development is to produce an application that is readily available to resolve software issues in an organization, reliable in terms of its operation, can ensure confidentiality at the highest level, protect the safety of sensitive data, uphold the integrity of the system, and require

low-cost maintenance. This paper explored the above-mentioned issues and modeled them in the CBSD process as six dependability attributes, namely, availability, reliability, confidentiality, safety, integrity, and maintainability. These attributes were embedded in the 2DCBS model based on a proposed guideline to develop a dependable system. This guideline can assist software developers in incorporating the dependability attributes into the requirement analysis, design, implementation, and testing process. The vulnerabilities of the systems can be promptly tolerated and risks can be reduced to a manageable level.

The evaluation of 2DCBS model were carried out based on two stages. First, 2DCBS framing was evaluated by conducting a survey and an interview with experts. Second, the dependability attributes of the developed system were verified using VATs. The survey results on 2DCBS framing indicate that experts highly agreed on the following: a) all 2DCBS processes are essential to CBSD, b) 2DCBS solves the sequential structure issue, c) architectural phase compartmentalization is beneficial, d) IID is suitable for distributed development, and e) 2DCBS specifies a location for reusable components. Thus, 2DCBS framing achieves its objective. Aside from that, the vulnerability assessment results show that compared with the traditional development model, the 2DCBS model could mitigate vulnerabilities. The degradation observed in the availability of the 2DCBS model was improved, being only 3.34%. The assessment results show that 2DCBS deployment performed more efficiently than traditional deployment does as the system load gradually increased.

Furthermore, to significantly improve the CBSD process, this paper presented a model that provides traceability and is easy to understand and apply even by those who are new to the CBSD approach. Furthermore, the implementation of the 2DCBS model in developing an ICT portal proves its versatility in software development. The links between the CBSD phases, which are requirement analysis, design, implementation, and testing, and the dependability attributes, which have been neglected in existing studies, are clearly shown in this study. These links can further convince software developers that the CBSD approach is completely reliable. As a result, the confidence of the software industry stakeholders in the reliability of CBSD products can be increased.

Our future work will focus on providing a tool support for developers based on 2DCBS model. This feature can be considered in future investigations to present a visual modelling environment for embedding dependability attributes; allow automatic mapping of analysis results; and design, implement, and assess the dependability attributes. In this paper, the 2DCBS model is applied to develop a web application system. The developed system is evaluated after the system is deployed within a short run time. Therefore, future studies can evaluate the developed system deployed within a long run time (for example, one year) to investigate system tolerance in a real-time, long-running system.

Acknowledgments: The research is partially supported by the Universiti Malaysia Pahang. Project number: (RDU-160372).

Author Contributions: All the authors have contributed to this work. All authors read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gill, N.; Tomar, P. Modified development process of component-based software engineering. *ACM SIGSOFT Software Engineering Notes* **2010**, *35*, 1-6.
2. Brada, P. Enhanced type-based component compatibility using deployment context information. *Electronic Notes in Theoretical Computer Science* **2011**, *279*, 17-31.
3. Ganguly, D.; Bhattacharyya, S. Winning the industrial competitiveness with e-commerce adopting component-based software architecture. *Advances in Computer Science, Intelligent System and Environment* **2011**, 69-75.

4. Jun, G.; Bo, W.; Yunsheng, W.; Bin, Z.; Jiaojiao, W. In *Research of the software aging regeneration strategy based on components*, 2012; Springer: pp 601-608.
5. Shang, M.; Wang, H.; Jiang, L. In *The development process of component-based application software*, 2011; IEEE: pp 11-14.
6. Carvalho, F.; Meira, S.R.L.; Freitas, B.; Eulino, J. In *Embedded software component quality and certification*, 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009. SEAA '09, Patras 2009; IEEE: Patras pp 420-427.
7. Nadeem, A.; Asim, M.R.; Qureshi, M. A step forward to component-based software cost estimation in object-oriented environment. *Pakistan Journal of Science* **2012**, *62*, 250-257.
8. Ahmed, B.; Rizwan, Q.; Asif, I.K. A framework for next generation mobile and wireless networks application development using hybrid component based development model. *International Journal of Research and Reviews in Next Generation Networks (IJRRNGN)* **2012**, *1*, 51-58.
9. Xin, C.; Zhao, J.; Fu, K.; Chang, Y. Refactoring of mechanical model simulation software based on component technology. *Advanced Materials Research* **2012**, *466*, 1145-1149.
10. Aris, H.; Salim, S. The development of a simplified process model for cbsd. *The International Arab Journal of Information Technology* **2007**, *4*, 89-96.
11. Mohanty, S.; Acharya, A.A.; Mohapatra, D.P. In *A model based prioritization technique for component based software retesting using uml state chart diagram*, 2011; IEEE: pp 364-368.
12. Venčkauskas, A.; Štuikys, V.; Toldinas, J.; Jusas, N. A model-driven framework to develop personalized health monitoring. *Symmetry* **2016**, *8*, 65.
13. Conejero, J.M.; Sánchez-Figueroa, F.; Rodríguez-Echeverría, R.; Preciado, J.C. Scpl: A social cooperative programming language to automate cooperative processes in (a) symmetric social networks. *Symmetry* **2016**, *8*, 71.
14. Qin, G.; Wang, L.; Li, Q. Resource symmetric dispatch model for internet of things on advanced logistics. *Symmetry* **2016**, *8*, 20.
15. Moradian, E.; Håkansson, A. Controlling security of software development with multi-agent system. *Knowledge-Based and Intelligent Information and Engineering Systems* **2010**, 98-107.
16. Sommerville, I. *Software engineering: Ninth edition*. Pearson-Addison Wesley: Boston Massachusetts United States, 2011.
17. Karen, G. *Software security assurance: A state-of-the-art report (soar)*; DTIC Document: 2007.
18. Karen, G.; Winograd, T.; McKinley, H.L.; Holley, P.; Hamilton, B. Security in the software life cycle: Making software development processes–and the software produced by them–more secure, draft 1.1. *Department of Homeland Security* **2006**, 46.
19. Redwine Jr, S. Software assurance: A curriculum guide to the common body of knowledge to produce, acquire, and sustain secure software. *H. Security* **2007**.
20. Mir, I.A.; Quadri, S. Analysis and evaluating security of component-based software development: A security metrics framework. *International Journal of Computer Network and Information Security (IJCNIS)* **2012**, *4*, 21.
21. Cinque, M.; Cotroneo, D.; Pecchia, A. Enabling effective dependability evaluation of complex systems via a rule-based logging framework. *International Journal On Advances in Software* **2010**, *2*, 323-336.

22. Goertzel, K.M. Introduction to software security. In *Build Security In*, Department of Homeland Security and Department of Defense Data and Analysis Center for Software: 2009.
23. McGraw, G. Software security. *Datenschutz und Datensicherheit-DuD* **2012**, *36*, 662-665.
24. Yi, S.; Li, D. In *The research of component-based dependable encapsulation*, Proceedings of the 13th International Conference on Mathematical Methods in Electrical Engineering and Computer Science, Angers, France, 2011; World Scientific and Engineering Academy and Society (WSEAS): Angers, France, pp 27-30.
25. Ahmed, B.; Abdurrahman, A.-T.; Rizwan, Q.; Asif, I.K. Novel component based development model for sip-based mobile application. *International Journal of Software Engineering & Applications (IJSEA)* **2012**, *3*, 85-99.
26. AUTOSAR, G.R. Autosar–technical overview v2. 0.1. June: 2006.
27. Basu, A.; Bozga, M.; Sifakis, J. In *Modeling heterogeneous real-time components in bip*, Fourth IEEE International Conference on Software Engineering and Formal Methods, 2006. SEFM 2006., 2006; Ieee: pp 3-12.
28. Box, D. Essential com. Object technology series. Addison-Wesley: Reading, England, 1997.
29. Åkerholm, M.; Carlson, J.; Fredriksson, J.; Hansson, H.; Håkansson, J.; Möller, A.; Pettersson, P.; Tivoli, M. The save approach to component-based development of vehicular systems. *Journal of Systems and Software* **2007**, *80*, 655-667.
30. Sentilles, S.; Vulgarakis, A.; Bureš, T.; Carlson, J.; Crnković, I. A component model for control-intensive distributed embedded systems. *Component-Based Software Engineering* **2008**, 310-317.
31. Van Ommering, R.; Van Der Linden, F.; Kramer, J.; Magee, J. The koala component model for consumer electronics software. *Computer* **2000**, *33*, 78-85.
32. DeMichiel, L.; Keith, M. Jsr 220: Enterprise javabeans, version 3.0: Ejb core contracts and requirements. *Final release, Sun Microsystems* **2006**.
33. Emmerich, W. In *An overview of omg/corba*, Distributed Objects-Technology and Application (Digest No: 1997/332), IEE Colloquium on, 1997; IET: pp 1/1-1/6.
34. Chen, J.; Wang, H.; Zhou, Y.; Bruda, S.D. Complexity metrics for component-based software systems. *International Journal of Digital Content Technology and its Applications* **2011**, *5*, 235-244.
35. Kumari, U.; Bhasin, S. A composite complexity measure for component-based systems. *ACM SIGSOFT Software Engineering Notes* **2011**, *36*, 1-5.
36. Cann, S.; Rossi, A.; Pilgrim, P. Frameworks for building component based applications. http://www.jcorporate.com/econtent/Content.do?state=resource&resource=702: 2004.
37. Arvinder, K.; Mann, K. Component based software engineering. *International Journal of Computer Applications IJCA* **2010**, *2*, 105-108.
38. Fredriksson, J. *Improving predictability and resource utilization in component-based embedded real-time systems*. School of Innovation, Design and Engineering, Mälardalen University: 2008.
39. Kaur, K.; Kaur, P.; Bedi, J.; Singh, H. In *Towards a suitable and systematic approach for component based software development*, 2007; Citeseer: pp 190-193.

40. Sözer, H.; Hofmann, C.; Tekinerdoğan, B.; Aksit, M.; Gelenbe, E.; Lent, R.; Sakellari, G. Runtime verification of component-based embedded software. **2012**.
41. Salmi, N.; Ioualalen, M. Towards efficient component performance analysis in component based architectures. *Software Quality. Process Automation in Software Development* **2012**, 121-142.
42. Yang, Z.; Ju, F.; Shao, B. Research on integration of spatial data mining and gis based on component technology. *Advances in Computational Environment Science* **2012**, 161-167.
43. Tekumalla, B. Status of empirical research in component based software engineering-a systematic literature review of empirical studies. **2012**.
44. Mathias, E.; Baude, F. In *Multi-domain grid/cloud computing through a hierarchical component-based middleware*, Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, New York, USA, 2010; ACM: New York, USA, p 2.
45. Navas, J.F.; Babau, J.P.; Pulou, J. Reconciling run-time evolution and resource-constrained embedded systems through a component-based development framework. *Science of Computer Programming* **2012**.
46. Riaz, S. Moving towards component based software engineering in train control applications. Linköping University, 2012.
47. Baumgart, S.; Froberg, J.; Punnekkat, S. In *Towards efficient functional safety certification of construction machinery using a component-based approach*, 3rd International Workshop on Product Line Approaches in Software Engineering (PLEASE), 2012, 2012; IEEE: pp 1-4.
48. Adler, R.; Schaefer, I.; Trapp, M.; Poetzsch-Heffter, A. Component-based modeling and verification of dynamic adaptation in safety-critical embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)* **2010**, *10*, 20.
49. Neelam, S. A component-based model for e-business, integrating knowledge management and e-commerce. *Jour-nal of Information and Operations Management ISSN* **2012**, 0976-7754.
50. Otte, W.R.; Gokhale, A.; Schmidt, D.C. Efficient and deterministic application deployment in component-based enterprise distributed real-time and embedded systems. *Information and Software Technology* **2012**.
51. Kosuki, Y.; Okada, Y. In *3d visual component based development system for medical training systems supporting haptic devices and their collaborative environments*, Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), 2012, 2012; IEEE: pp 687-692.
52. Baacke, L.; Mettler, T.; Rohner, P. Component-based process modelling in health care. In *17th European Conference on Information Systems, ECIS 2009*, Verona, Italy, 2009; pp 430–441.
53. IrshadKhan, A.; Alam, M.; Noor-ul-Qayyum, N.-u.-Q.; Ali Khan, U. Validation of component based software development model using formal b-method. *International Journal of Computer Applications* **2013**, *67*, 24-35.
54. Pandeya, S.S.; Tripathi, A.K. Testing component-based software: What it has to do with design and component selection. *Journal of Software Engineering and Applications* **2011**, *4*.
55. Chhillar, R.S.; Kajla, P. A new-knot model for component based software development. *International Journal of Computer Science* **2011**, *8*.
56. Kaur, K.; Singh, H. Candidate process models for component based software development. *Journal of Software Engineering* **2010**, *4*, 16-29.

57. Sharp, J.; Ryan, S. A theoretical framework of component-based software development phases. *ACM SIGMIS Database* **2010**, *41*, 56-75.
58. Olsen, S.L.N.; Loe, K. The coexsel tool. Norwegian University of Science and Technology, NTNU. Department of Computer and Information Science, IDI, Norwegian, 2005.
59. Hasan Kahtan ; Nordin Abu Bakar ; Rosmawati Nordin. In *Reviewing the challenges of security features in component based software development models*, IEEE Symposium on E-Learning, E-Management and E-Services (IS3e), 2012, Kuala Lumpur 2012; IEEE Kuala Lumpur pp 1 -6.
60. Brown, A.W.; Wallnan, K.C. In *Engineering of component-based systems*, 1996; Published by the IEEE Computer Society: p 414.
61. Aoyama, M. In *Process and economic model of component-based software development: A study from software cals next generation software engineering program*, 1997; IEEE: pp 100-103.
62. Tran, V. In *Component-based integrated systems development: A model for the emerging procurement-centric approach to software development*, 1998; IEEE: pp 128-135.
63. Lee, S.; Yang, Y.; Cho, E.; Kim, S.; Rhew, S. In *Como: A uml-based component development methodology*, 1999; IEEE Computer Society: p 54.
64. Yau, S.; Dong, N. Integration in component-based software development using design patterns. *COMPSAC-NEW YORK-* **2000**, 369-376.
65. Cheesman, J.; Daniels, J.; Szyperski, C. *Uml components: A simple process for specifying component-based software*. Addison-Wesley Reading, MA: 2001; Vol. 2.
66. Paul, A. Ebiz components. *Objective View, no. 6* **2003**, pp. 12-20.
67. Crnković, I. Component-based software engineering-new challenges in software development. In *Journal of computing and information technology CIT*, IEEE: Minneapolis, MN, USA, 2003; Vol. 11, pp 157 - 158.
68. Hutchinson, J.; Kotonya, G.; Sommerville, I.; Hall, S. In *A service model for component-based development*, 2004; IEEE: pp 162-169.
69. Capretz, L. A software process model for component-based development. *Information Technology Journal* **2004**, *3*, 176-183.
70. Mei, H. Abc: Supporting software architectures in the whole lifecycle. In *The Second International Conference on Software Engineering and Formal Methods SEFM 2004.*, IEEE Computer Society: Beijing, China, 2004; pp 342 - 343.
71. Capretz, L. Y: A new component-based software life cycle model. *Journal of Computer Science* **2005**, *1*, 76-82.
72. Crnkovic, I.; Chaudron, M.; Larsson, S. In *Component-based development process and component lifecycle*, International Conference on Software Engineering Advances, Tahiti 2006; Tahiti p44.
73. Qureshi, M.; Hussain, S. A reusable software component-based development process model. *Advances in Engineering Software* **2008**, *39*, 88-94.
74. Kouroshfar, E.; Yaghoubi Shahir, H.; Ramsin, R. Process patterns for component-based software development. *Component-Based Software Engineering* **2009**, 54-68.
75. Bose, D. Component based development. Cornell University Department of Computer Science, 2010.

76. Lau, K.K.; Taweel, F.M.; Tran, C.M. In *The w model for component-based software development*, 2011; IEEE: pp 47-50.
77. Hasan Kahtan; Nordin Abu Bakar; Rosmawati Nordin. Dependability attributes for increased security in component-based software development. *Journal of Computer Science* **2014**, *10*, 1298-1306.
78. Hasan Kahtan; Nordin Abu Bakar; Rosmawati Nordin. Awareness of embedding security features into component-based software development model: A survey. *Journal of Computer Science* **2014**, *10*, 1411-1417.
79. Hasan Kahtan ; Nordin Abu Bakar ; Rosmawati Nordin; Mansoor Abdullateef Abdulgabber. Evaluation dependability attributes of web application using vulnerability assessments tools. *Information Technology Journal* **2014**, *13*, 2240-2249.
80. Hasan Kahtan; Nordin Abu Bakar ; Rosmawati Nordin. Embedding dependability attributes into component-based software development using the best practice method: A guideline. *Journal of Applied Security Research* **2014**, *9*, 348 - 371.
81. Runeson, P.; Host, M.; Rainer, A.; Regnell, B. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons: Hoboken, New Jersey, 2012.
82. Coppit, D.; Sullivan, K.J. In *Multiple mass-market applications as components*, International Conference on Software Engineering Limerick, Ireland, 2000; IEEE: Limerick, Ireland, pp 273-282.
83. Hasan Kahtan ; Nordin Abu Bakar ; Rosmawati Nordin; Mansoor Abdullateef Abdulgabber. Embedding dependability attributes into component-based software development: A guideline. *Computer Fraud & Security - Elsevier* **2014**.