Journal of Theoretical and Applied Information Technology
20th April 2015. Vol.74 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A RELIABILITY ESTIMATION MODEL USING INTEGRATED TASKS AND RESOURCES

[1]MOHD ADHAM ISA and [2]DAYANG NORHAYATI ABANG JAWAWI,
[1,2]Department of Software Engineering, Faculty of Computing
[1,2]Universiti Teknologi Malaysia, Malaysia
E-mail: [1]mohdadham@utm.my, [2]dayang@utm.my

ABSTRACT

With the growing size of modern systems, the composition of a number of resources for a system is becoming increasingly more complex. Thus, a reliability analysis for that system is essential, especially during design time. The reliability estimation model is rapidly becoming a crucial part of the system development life cycle, and a new model is needed to enable an early analysis of reliability estimation, especially for the system under study. However, the existing approach neglects to address the correlation between resource and system task for estimation of system reliability. This subsequently restricts the accuracy of estimation results and thus, could misguide the reliability analysis in general. This paper proposes a reliability estimation model that enables the computation of the system reliability as a product of resource and system task. The system task reliability is treated as a transition probability that the resource may execute for subsequent resources. To validate the model, one real case study is used and the accuracy of the estimation result is compared with the actual reliability values. The result shows the estimation accuracy is considered at an acceptable level and some of the scenarios have recorded higher accuracy values than previous models. To evaluate our model, the result is compared with that of the existing model and shows our model providing a more accurate estimation for a more complex scenario

Keywords: System reliability estimation, graph-theory, white-box test

1. INTRODUCTION

Software systems in use today are vastly more complex than in previous years and work silently behind the operation of daily life. Supporting the quality of these systems poses a new challenge and a demand for reliable software systems is becoming much more important than ever before. Unreliable systems can cause major problems of inconvenience and even fatal accidents. Therefore, adequate estimation for reliability analysis concerning systems under study would prevent unwanted problems and could avoid serious mishaps in system designs. Prior to reliability estimation, results could also help developers to foresee an expected anomaly in system behavior with respect to the system task and resource [1], [2]. Thus, a suitable estimation reliability model is needed at the design phase so as to accommodate such a circumstance.

The importance of reliability analysis has been established throughout the software engineering community. The standard software engineering practice called Software Reliability Engineering (SRE) [3] generally provides a systematic method on how to facilitate the reliability analysis with the system throughout the development phase. Due to some drawbacks and issues in conducting a reliability analysis during runtime [4], many approaches have appeared to analyze the system reliability during design time [4], [5]. This approach provides more advantages and has become indispensable in the past two decades.

In particular, the assessment of system reliability has been treated as an assembly of components and its reliability has been estimated based on components and transition reliability between components [6]. In this case, two important criteria are further elaborated upon, namely; system structure and failure behavior [7].

Concerning the first criteria, the structure of the system is a connected component, representing a control flow of the components to fulfill system requirements [8]–[10]. The representation is usually in the form of a program graph [11], which is able to translate the flow of the components. Typically,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the prevalent system structure for reliability analysis is divided into two categories, namely: state-based and path-based approaches. Detailed explanations of these models can be found in [12]. In most cases, the system is modeled using a prominent modeling language called UML. This language provides a several standard specification and profile for various domains of systems. Current works can be found in [13], [14] explaining how the UML-based approach for component modeling has been used for designing the system, as well as estimating reliability at design time.

Regarding the second criteria, the failure behavior for components and its transition is identified as being a subtle requirement for reliability estimation. Works in [12] measure the components' failure behavior based on time-dependent failure intensity (which is its timing requirement) and which can be obtained from a usage profile. With the presence of timing requirements in system models, the components' reliability can also can be estimated by manipulating the execution time spent with components before being transferred to a following component [15]. In [16] for instance, the timing is treated as a first class artifact, that is, to assess the level of risk for the system that may harm the system reliability in general.

Therefore, the scope in this paper treated an assembly component as a resource. Instead of treating a single component as a static plug-and-play component, we further elaborated upon it as a resource. Each resource is associated with a system task that is responsible for determining the reliability of transition for the next resource execution. A reliability of transition between resources is determined by task execution. This scenario could be determined by time execution for completion of a system task before being delivered to an alternate system task at another resource.

This paper proposes an estimation reliability model, which is based on a graph theory that enables developers to estimate reliability of the system by analyzing the interaction of system resources via system task. The proposed model is called Resource Dependency Graph (RDG), extended from our previous works in [17]. We generally focus on reliability estimation for systems and assume that the reliability of each component, task and its transition is available. The applicability of the model is validated by one real case study and the result is compared with actual reliability values. The model is further evaluated in conjunction with the existing model so as to compare the accuracy of reliability estimation. A few suggestions are made to discern a difference between both models.

2. RELATED WORKS

Many previous works have neglected the influence of an execution environment, such as *Fault Tree analysis* (FTA) on the reliability of a system. This investigates the fault propagation in architecture and application models [18]. For instance, in [19], the performance aspects at architectural level were discovered based on workload distribution (input parameters). The importance of correlation between system task and resource usage in scenario executions for reliability analysis can be found in [1], [20], [21]. Other works were only addressed for component interaction aspects [22], [23] and resource usage [24] respectively.

Concerning works for reliability estimation of a system based on components and its transition, it is worth describing the works in [15], [25]–[27]. The study in [15] proposed the Component Dependency Graph (CDG) for analyzing component interaction behavior and its transition probability. However, this approach is not explicitly associated with the triggered *system task* that is used by the resources, and may cause inconvenience for misuse of resources. This problem may suffer drawbacks that increase the probability of failure of the resources. Another work by Lo et al., [25] considers the estimation of system reliability based on multiple input and output values. The estimation model is called Lo component-based reliability model (LCBRM) and the output of the result is determined by sensitivity analysis so as to determine the most sensitive parameters in relation to the estimation accuracy. Works in Hsu and Huang [26] adapted a path-based model called Adaptive reliability analysis. This model uses path testing (ARAUPT) and estimates the reliability of system based on mixture structure (branch, loop and sequence) in order to determine the best scenario to be tested for reliability analysis. Another estimation model worthy of mention is Cheung's user-oriented reliability (CUORM) [27] which estimates the reliability of the system with respect to user module reliability and covers heterogonous system structures such as pipeline, call-return and fault tolerance.

Aggregating this together, the influence of execution environment (resource and task) is worth considering for estimating system reliability. Existing works mostly include only component and transition reliability as an ingredient for reliability estimation. In fact, the influence of system task in

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

each component silently affects the transition from one component to another. On the other hand, the influence of scenario execution should be considered for reliability estimation. This is because the estimation for every possible scenario could help to identify which scenario is the most reliable and hence foresee the effects of scenario complexity with reliability level. Therefore, the criteria of estimation result should be driven by the following requirements, namely: the complexity of system scenario and the transition between components through task transition.

3. THE PROPOSED MODEL

The proposed reliability estimation model is called a Resource Dependency Graph (RDG) model. The model is constructed based on a modular system and is represented as a control flow graph. The behavior in a model assembly of the resource consists of a number of system tasks contained in the resources. The formal definition for RDG can be described as follows:

Definition 1: A resource dependency graph (RDG) is a 4-tuple $\boldsymbol{RDG} = \langle \boldsymbol{RN}, \boldsymbol{E}, \boldsymbol{start}, \boldsymbol{end} \rangle$ *where*:

1. RN is a finite set of resource nodes in the graph where $\boldsymbol{RN} = \{\boldsymbol{rn_i}\}, \boldsymbol{i} = \boldsymbol{1, 2, 3, \dots, |RN|}$
2. E is a finite set of directed edges where $\boldsymbol{E} = \{\boldsymbol{e}\}, \boldsymbol{i} = \boldsymbol{1, 2, 3, \dots, |E|}$
3. $start$ is a start node.
4. end is a termination node

Definition 2: A *resource node* "r": $r \in RN$ is a system resource and is defined by a tuple $< \boldsymbol{rn_i}, \boldsymbol{st_i}, \boldsymbol{rv_i}, \boldsymbol{Rr_i} >$ where: rn_i is the name of the resource; $\boldsymbol{st_i}$, is the name of system task attached to the resource node r_i; $\boldsymbol{rv_i}$, concerns the number of visits to the resource node r_i; and $\boldsymbol{Rr_i}$ is the reliability of resource node r_i.

Definition 3: *System task* of a resource $\boldsymbol{st_i}$ is the list of the task of systems required to execute correctly upon system execution where $\boldsymbol{st_1, st_2, st_3} \dots \boldsymbol{st_i} \xrightarrow{\boldsymbol{requires}} r_i$. Additionally, each system task $\boldsymbol{st_i}$ **is** possessed of its own reliability $\boldsymbol{Rst_i}$ where the definition is defined as the probability of system task $\boldsymbol{st_i}$ to be executed in an error-free manner.

Definition 4: *Resource reliability* $\boldsymbol{Rr_i}$ is the probability that the resource $\boldsymbol{r_i}$ executes correctly in a specific time t and condition upon system execution.

Definition 5: A *directed edges* "e": $e \in E$ is a workflow transaction from one resource to another. The transaction is invoked by system task $\boldsymbol{st_i}$ and represented by a control flow of resources. Its definition is defined by a tuple $< \boldsymbol{tr_{ij}}, \boldsymbol{Rtr_{ij}} >$ where: $\boldsymbol{tr_{ij}}$ is the transition name from system task $\boldsymbol{r_i(st_i)}$ to $\boldsymbol{r_j(st_j)}$ and $\boldsymbol{Rtr_{ij}}$ is the reliability of the transition.

Definition 6: *Transition reliability* $\boldsymbol{Rtr_{ij}}$ is the error-free transaction from one resource to another via system task control flow. The information sent includes the probability of possible system task failure.

As mentioned above, the method employed to acquire the parameter values for system task $\boldsymbol{Rst_i}$, resource $\boldsymbol{Rr_i}$ and transition $\boldsymbol{Rtr_{ij}}$ will be explained in detail in Section 4.1. However, to show the applicability of the proposed RDG model, a benchmark case study will be used, where the values for those parameters are already available.

3.1 Diagram Compositional

The representation of RDG model is based on the defined definitions given in a previous section. As shown in Figure 1, the RDG model consists of a finite set of resource nodes: $r \in RN$ accommodates with start node $start$ and ends with node end. The elements of resource node r for system task $\boldsymbol{st}$ can be one or more, $\boldsymbol{st_i\ where\ i = 1, 2, 3, \dots, |N|}$ and represented as $\{\boldsymbol{st_1, st_2, st_3, \dots, st_n}\}$. These system tasks determine how the resource node behaves for a certain execution scenario, depending on its system architecture.

In order to cope with the system structure complexity, the RDG model can be contained with a mixture structure; *sequence* and *branch*. The sequence structure is a call sequence with direct transition from resource r_i to resource r_j without introducing parallel execution, in contrast to the branch structure. Whilst the RDG model is founded on a scenario-based approach, in a logical sense there is no loop structure, which has been introduced. For example, a decision statement for one call sequence to complete one scenario is based on system task transition. Consequently, the composition of loop structure is omitted due to the uncertainty of loop numbers that could lead to incorrect call sequence termination during a scenario execution decision. Thus, the system reliability based on RDG is the combination of

Journal of Theoretical and Applied Information Technology
20th April 2015. Vol.74 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

sequence and branch structure and will be explained in detail in Section 3.3.

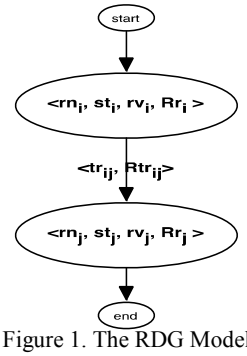

Figure 1. The RDG Model

3.2 Parameter Estimations

Initial input values are essential to initiate an execution scenario at the system level as these values can determine how the system may behave. The values could also influence the control flow of resources during an execution scenario that is highly dependent on input stimuli. For reliability estimation at design level, the input values can be obtained through various techniques such as usage profile, fault injection and testing coverage.

In our RDG model, we defined three parameters for calculating system reliability based on an execution scenario. These parameters can stimulate the estimation of system reliability as well as depicting the relationship via the aforementioned values.

A. System task reliability "$\boldsymbol{Rst_i}$".

In this paper, we assume that the reliability of individual system task is available and has been calculated by domain experts. Several techniques involving reliability estimations for system task have been proposed including testing, code coverage and so on. We use this parameter as groundwork for estimating each of the system resources.

B. Resource reliability "$\boldsymbol{Rr_i}$".
The parameter for individual resource reliability $\boldsymbol{Rr_i}$ is calculated based on each attached system task at the resource $\boldsymbol{r_i}$ for a specific scenario. The system scenario is built up depending on the system task interaction within each resource to another. It is calculated using the following equation:

$$Rr_i = \prod_{i=1}^{n} Rst_{i\ in\ S_x} \quad (1)$$

where n is the number of system task $\boldsymbol{st_i}$ in resource $\boldsymbol{r_i}$ and S_x represents the current execution scenario in the execution time of $\boldsymbol{Rst_i}$ and $\boldsymbol{Rr_i}$.

C. Transition reliability "$\boldsymbol{Rtr_{ij}}$".

This parameter represents the reliability transition from st_i to st_j. It is the probability by which the system task is correctly delivered from $\boldsymbol{r_i}$ to $\boldsymbol{r_j}$ to analyze the effect of the system reliability through resource interaction via system task transition. Based on our RDG model, the resource interaction is executed through an internal system task that is attached to each resource. The time for each system task is completed before being transferred to a next task and is assumed to be a transition probability. Therefore, the transition reliability $\boldsymbol{Rtr_{ij}}$ can be calculated as:

$$Rtr_{i,j} = Rr_i * Rst_i | i = 1,2,3,\dots,|RN|_{S_x} \quad (2)$$

where S_x represents the current execution scenario in the transition path of resource.

3.3 Method for reliability estimation

The methods used to calculate the system reliability consist of two structures, namely, *sequence* and *branch*. These structures indicate the execution scenario of the systems, which is, representing the interaction of system resources via the control flow of system task.

3.3.1 Sequential structure

Execution scenario is supposedly in a sequential order, where the transaction of resources extends from $\boldsymbol{tr_{1,2}}$ to $\boldsymbol{tr_{n-1,n}}$. Assuming the RDG model consists of n resource nodes, the reliability of the system scenario RS_x can be calculated as:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$RS_x = Rr_n \times \prod_{i=1}^{n} (Rr_i^{Rtr_{i,i+1}})^{\prod_{j=1}^{i}(Rr_{i-j})^{Rtr_{i-j,i-j+1}}} \quad (3)$$

where S_x is the current execution scenario of systems and each resource reliability is assumed to have failed independently for a subsequent system task transition.

Proof: The scenario is executed in a sequential order as shown in Figure 2, and the probability of failure of the delivered information from resource r_1 to r_2 is influenced by the transition $Rtr_{1,2}$. The next transition follows Markov process rules, where a transition to the following resource is independent of past resources and depends only on the current resource. Therefore, the expected probability value RP_i to reach subsequent resource nodes can be calculated as:

$$RP_i = (Rr_1^{Rtr_{1,2}}) \times (Rr_2^{Rtr_{2,3}}) \times \ldots \times (Rr_{i-1}^{Rtr_{i-1,i}}) = \prod_{j=2}^{i} Rr_{j-1}^{Rtr_{j-1,j}} \quad (4)$$

All the resources along the path will form a series of call sequences, which represent the views of reliability. The reliability of the call sequence is computed as a product of resource. The current resource reliability is dependent upon the resource along the path that the current resource may travel. Therefore, the reliability of call sequence can be denoted as:

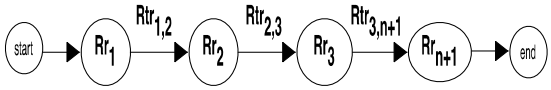

Figure 2. Sequential structure

$$RS_x = Rr_n \times \left(Rr_1^{Rtr_{1,2}}\right) \times \left(Rr_2^{Rtr_{2,3}}\right)^{Rr_1^{Rtr_{1,2}}} \times \left(Rr_3^{Rtr_{3,4}}\right)^{Rr_1^{Rtr_{1,2}} \times Rr_2^{Rtr_{2,3}}} \times \times (R_n^{Rtr_{n,n+1}})^{Rr_1^{Rtr_{1,2}} \times Rr_2^{Rtr_{2,3}} \times \ldots \times Rr_{n-1}^{Rtr_{n-1,n}}} = Rr_n \times \prod_{i=2}^{n} \left(Rr_i^{Rtr_{i,i+1}}\right)^{\prod_{j=1}^{i} Rr_{i-j}^{Rtr_{i-j,i-j+1}}} \quad (5)$$

3.3.2 *Branch structue*

A branch structure could be performed in an execution scenario if there is any parallel transaction from one resource to another. The transaction is realized with a different connecting system task. Supposing that a system has n resource node consisting of $n-2$ branches as shown in Figure 3, a reliability of a system scenario RS_x can be denoted as:

$$RS_x = \prod_{i=1}^{n} Rr_1^{Rtr_{1,i}} \times \prod_{i=2}^{n} (Rr_i^{Rtr_{i,n}})^{\prod_{j=1}^{i} Rr_1^{Rtr_{1,j+1}}} \quad (6)$$

Proof: A branch structure poses a similar structure to a sequence structure. The difference between both structures is that there is a possibility that one resource may execute a parallel system task to different resources. As shown in Figure 3, we adapted a similar calculation for sequence structure for branching formulation depicted as:

$$RS_x = 1 \times \left[(Rr_1^{Rtr_{1,2}}) \times (Rr_1^{Rtr_{1,3}}) \times (Rr_1^{Rtr_{1,n-1}})\right] \times \left[(Rr_2^{Rtr_{2,n}})^{Rr_1^{Rtr_{1,2}}} \times (Rr_3^{Rtr_{3,n}})^{Rr_1^{Rtr_{1,3}}} \times (Rr_{n-1}^{Rtr_{n-1,n}})^{Rr_1^{Rtr_{1,n-1}}}\right] = \prod_{i=1}^{n} Rr_1^{Rtr_{1,i}} \times \prod_{i=2}^{n} (Rr_i^{Rtr_{i,n}})^{\prod_{j=1}^{i} Rr_1^{Rtr_{1,j+1}}} \quad (7)$$

In summary, the proposed model provides an approach for estimating reliability of the system using assembled resources and tasks. Early reliability values for resources, tasks and transition need to be estimated first so as to be input values for system estimation purposes. The process to acquire such parameters is outside the scope of this paper. The applicability of the models is further validated and evaluated in detail in a later section.

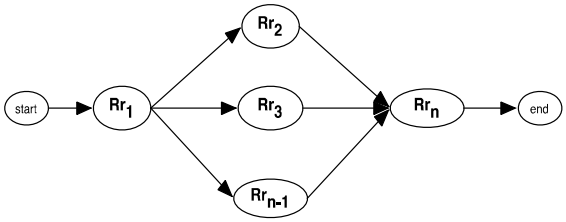

Figure 3. Branch structure

4. EVALUATION

Journal of Theoretical and Applied Information Technology
20th April 2015. Vol.74 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

To evaluate our RDG model, we used a real case study from [27] and compared the produced results with the existing techniques, which have already been reported in [26]. The case study is a complex system developed by Bell Labs for switching systems. The structure of the system as shown in Figure 4 was developed based on control flow between components and validated by [28], [29]. The availability of component and transition reliability has already been reported in [27] and outlined in Table 1. Hence, we assume the reliability values for each system task in each resource that consequently build up the reliability values for components and transition are as described in Section 3.2.

To conduct the experiment relative to the case study with our model, a few assumptions have been defined. The assumption does not change the paradigm of RDG model and still remains in line with its definition. From the case study, we assume that the internal system task reliability Rst_i is known as path reliability, as shown in Table 1. According to RDG model, the resource transaction is always triggered by a system task transition that is contained in the resources. Therefore, we assume all outgoing paths from one resource to another are triggered by system task transition. Thus, the transition reliability for RDG model $Rtr_{i,j}$ can be calculated as mentioned in Section 3.2 (c). As for

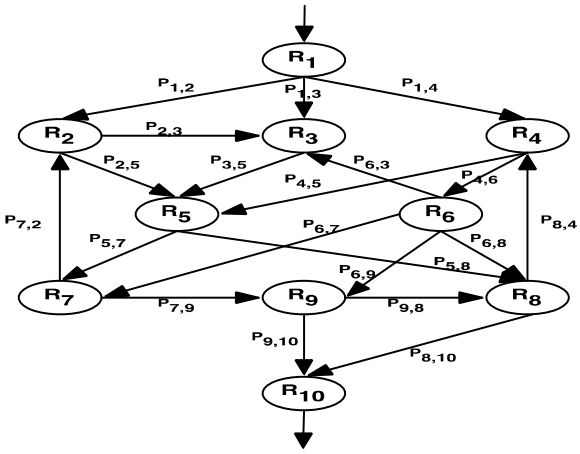

Figure 4. Case study

resource reliability, we derived this directly from components reliability as listed in Table 1.

The purpose of the experiment on the case study was to prove the effectiveness of our proposed model regarding reliability estimation accuracy as against the actual reliability value. A strategy was defined to show in principle how our proposed model is suitable for estimating a complex system. The strategy is outlined as below:

1. To show the acceptable degree of estimation accuracy as compared with actual reliability values, as well as for the previous model.
2. To compare the accuracy of estimation result of our model with that of the most similar model.

To show how our proposed model is capable of dealing with a more complex scenario in relation to another model in terms of estimation of accuracy results.

4.1 Experiment setting

In order to estimate the system reliability based on scenario, we constructed a four different call sequence (CS1, CS2, CS3, CS4) that represents a system scenario based on previous works [26] and which is shown in Figures 5 and 6. The call sequence is re-developed using RDG model and its reliability is calculated based on Eq. (5) and (7). Each call sequence carries a different weight of complexity in order to add a variety of system behaviors through the call sequence. CS1 for instance poses a simple call sequence wherein the sequence structure assembled five resources via task transition (path). Another more complex call sequence is depicted in CS4. The scenario consists of a mixture of structures (sequence and branch) and involves more resource and task transitions than the rest of the call sequence. The variety of the call sequence could indicate the effectiveness of our model when validated and evaluated with actual reliability values and against the existing model respectively.

***Table 1**. Reliability for Component and Path in Case study*

Component	Reliability	Path	Reliability	Path	Reliability
R1	0.999	$P_{1,2}$	0.6	$P_{6,3}$	0.3
R2	0.980	$P_{1,3}$	0.2	$P_{6,7}$	0.3
R3	0.99	$P_{1,4}$	0.2	$P_{6,8}$	0.1
R4	0.970	$P_{2,3}$	0.7	$P_{6,9}$	0.3
R5	0.950	$P_{2,5}$	0.3	$P_{7,2}$	0.5
R6	0.995	$P_{3,5}$	1	$P_{7,9}$	0.5
R7	0.985	$P_{4,5}$	0.4	$P_{8,4}$	0.25
R8	0.950	$P_{4,6}$	0.6	$P_{8,10}$	0.75
R9	0.975	$P_{5,7}$	0.4	$P_{9,8}$	0.1
R10	0.985	$P_{5,8}$	0.6	$P_{9,10}$	0.9

Journal of Theoretical and Applied Information Technology
20th April 2015. Vol.74 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

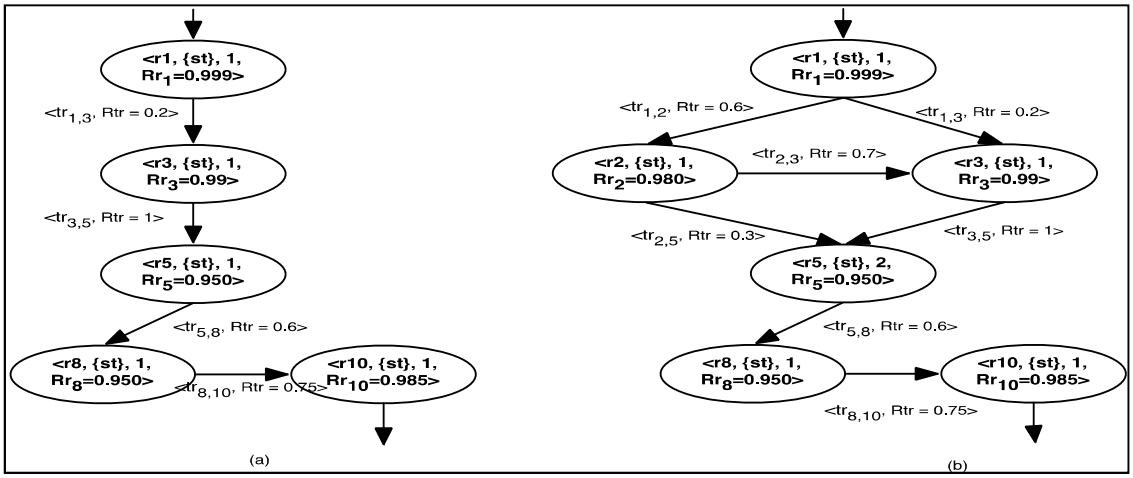

Figure. 5. Scenario for case study using RDG model (a) CS1 and (b) CS2

In order to validate the accuracy of reliability estimation, we used the accuracy (AC) metrics, which are originally derived from Magnitude Relative Error (MRE) metrics from [30]. The justification is to measure the percentage of accuracy as compared with actual reliability values, as well as the existing approach. The AC can be defined as:

$$MRE = \frac{|RS_E - RT|}{RT} \quad (8)$$

where the RT consists of the real values for reliability and RS_E is the estimated reliability values calculated by estimation model. The actual reliability values is obtained from [26] and we validate our model by calculating the AC for each of the respective reliability values as shown in Table 2.

In Table 2, the actual reliability value (0.826) is directly obtained from the case study. In addition, the reliability values for CUORM (0.829), LCBRM (0.827) and ARAUPT (0.8595) are recalculated using the values and system architecture in the model shown in Table 1 and Figure 4 respectively. Each model produced different values of reliability. In the LCBRM for instance, the model is based on a single input and output system. The reliability is calculated based on the product of reliability for each resource. The reliability of transition between resources is assumed based on the number of visits to the next resource before entering an absorbing resource. Another model such as ARAUPT is concerned with the path reliability during resource transition. The reliability of the system is calculated as a resource product where the current resource reliability is dependent upon the path that it may take.

In order to evaluate the estimation accuracy of our model, we compared the produced results with those of the most similar model. The model [26] calculated the system reliability based on aggregating its path reliability and mixture with three different system structures, namely; sequence, branch and loop. This is slightly different to our model, where we aggregated the path and resource reliability together. Using a similar case study, we calculated the AC for both models, made a comparison with actual reliability values (0.826) and then analyzed the result in view of the complexity of call sequence.

The complexity of the system structure could determine the level of its reliability and hence developers could foresee the behavior of the system. By utilizing the complexity value for each call sequence, the correlation between the call sequence complexity and the accuracy of reliability estimation can be analyzed in order to make a significant assumption of how the complexity of the system could affect the accuracy of estimation. Table 3 shows the complexity of call sequence CS1, CS2, CS3 and CS4 based on McAbe's

Journal of Theoretical and Applied Information Technology
20th April 2015. Vol.74 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

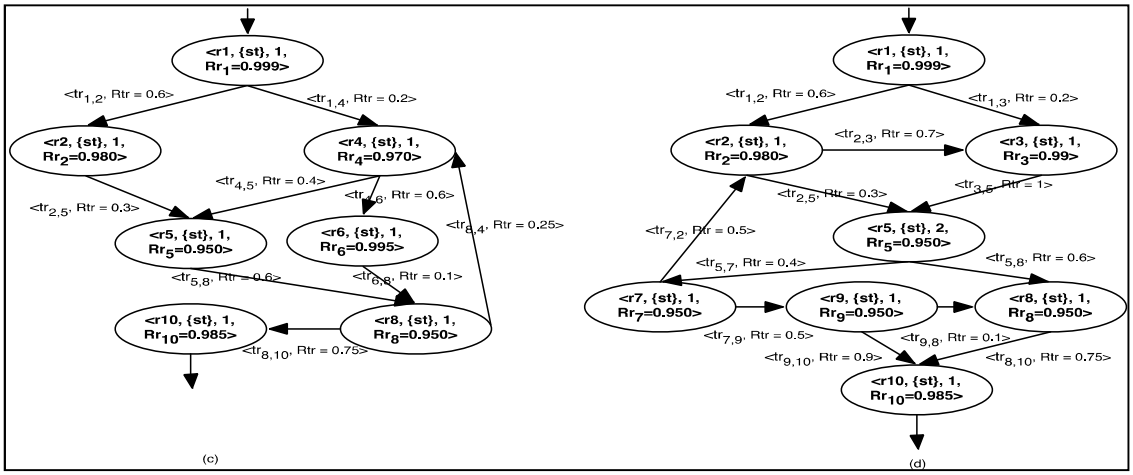

Figure. 6. Scenario for case study using RDG model (a) CS3 and (b) CS4

approach [31]. This approach is called *cyclomatic complexity (CC)* and calculates the complexity of modular structure of the system using a number of edges and nodes on system structures.

Table 2. *Actual and estimated reliability values from existing works*

Approach	Reliability
Actual reliability	0.8260
CUORM	0.8290
LCBRM	0.8270
ARAUPT	0.8595

Table 3*. Cyclomatic Complexity for Call Sequence*

Call Sequence	CC
CS1	1
CS2	3
CS3	4
CS4	6

4.2 Result on acceptable degree of estimation

From Table 4, the reliability for call sequence CS1, CS2, CS3 and CS4 are calculated using our model. Each call sequence poses a different structure that consequently leverages different complexity values. As depicted in Table 3, the complexity of each call sequence poses a different weight of complexity due to different resource structures. The complexity influences the reliability of each call sequence as shown in Table 4. As mentioned in Section 3.3, the reliability transition is in accordance with the Markov process rule, where the reliability of the next transition is solely dependent on the present resource. However, the call sequence reliability is the product of resource reliability, where the next resource reliability depends upon the current resource and is also dependent upon past resource reliability as well. Therefore, the more complex one call sequence is, the lower the reliability values that will be produced.

From the results in Table 5, the estimation accuracy AC for our proposed model is calculated against reliability values from case study and existing models as depicted in Table 2. This experiment seeks to discover the acceptable degree of estimation accuracy as compared with actual reliability, as well as reliability values produced by the existing model. For the result with *actual reliability,* the AC for CS3 (99.24%) and CS4 (97.66%) is higher than the AC for CS1 (89.66%) and CS2 (91.77%) respectively. As for comparing with the other model, the accuracy of the RDG model remains acceptable as depicted in Table 5.4. Interestingly, the result recorded with the RDG model is much closer as compared with the recent model (ARAUPT). This was essentially caused by the similar reliability method in RDG and ARAUPT, but the RDG model considers the past resource reliability rather than only path-reliability in ARAUPT. The inclusion of the past resource reliability for calculating the current resource reliability proved that the estimation accuracy had increased.

The AC for CS3 and CS4 is much closer to the actual reliability as compared with CS1 and CS2. The AC is also acceptable in comparison with CUORM, LCBRM and ARAUPT. This result

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

indicates the estimation accuracy is closely related to the complexity of the call sequence itself. This is due to the fact that the complexity of call sequence for CS1 and CS2 is lower than CS3 and CS4, as shown in Table 2. The greater the complexity the call sequence, the more accurate is the reliability estimation that can be produced. This is due to the fact that the complexity of call sequence poses a different kind of resource structure (branch, sequence). The number of resources and tasks involved also justifies the effects of system complexity affecting the reliability (which is in line with Software Growth Reliability Model (SGRM) [25] [32]). Therefore, this result shows that the complexity of call sequence should be considered as a justification of how it will significantly affect the accuracy of reliability estimations for reliability analysis.

***Table 4**. Reliability values for call sequence using RDG model*

Call Sequence	Call sequence reliability (RDG)
CS1	0.9114
CS2	0.8940
CS3	0.8323
CS4	0.8453

4.3 Result on accuracy degree of estimation

In order to evaluate the accuracy of the estimation, the RDG model is compared with the ARAUPT model. This model is quite similar to the RDG model and focuses on a path-based approach for estimating the reliability of the system. Although the ARAUPT model uses the path-based approach, the calculation of each resource, however, only considers the aggregation of path reliability along the resource execution. Similarly with the RDG model, however, this model makes an inclusion of the previous resource reliability, together with its path reliability, to calculate the reliability of the current resource reliability. Each resource depends upon the path along which the resource was executed. It differentiates from the proposed model in that the resource reliability is dependent upon other resources along the path that the current resource travels. With the selection of ARAUPT for the comparative model, the purpose is to show how the RDG model can handle a more accurate result of reliability estimation as against a more complex call sequence. By using the same case study, the AC of both models is compared against the actual reliability value.

***Table 5**. The comparison of estimation accuracy with RDG model*

Call Sequence	Reliability (RDG)	% Accuracy (AC)			
		Actual	CUORM	LCBRM	ARAUPT
CS1	0.9114	89.66	90.06	89.79	93.96
CS2	0.8940	91.77	92.16	91.90	95.99
CS3	0.8323	99.24	99.60	99.36	96.84
CS4	0.8453	97.66	98.03	97.79	98.35

From the result in Table 5.5, the AC of ARAUPT for CS1 and CS2, which are 93.62% and 94.68%, shows the closest result compared with the RDG model, 89.66% for CS1 and 91.77% for CS2 respectively. However, for CS3 and CS4, the RDG model recorded a more accurate estimation, especially for CS3, which is 99.24% as compared with ARAUPT, 97.88%. This is due to the fact that the call sequence for CS3 poses a high complexity value (Table 3) in accordance with the greater number of resources and transitions. In particular, the ARAUPT model estimates the system reliability based on the path of resources with mixture structure (loop, branch, sequence). Considering the inability of the ARAUPT model to deal with looping structure effectively, the estimation may conclude with an inaccurate result, thus leading to an infinite consequence. In contrast with a more direct structure (branch, sequence), the RDG model can directly calculate the reliability along the path of resource execution without intervention from uncertainty loops.

Contrary to the RDG model, the ability to treat a system as a call sequence gives a slight advantage when dealing with looping structure. The RDG model assumed the system is executed in a sequence and branch structure only and omitted the loops structure in order to avoid infinite consequence. The justification is supported by aggregating together not only path reliability but resource reliability as well. By doing this, the RDG model is proven to be more suitable for estimating the complex structure, rather than a simple structure like CS1 and CS2. For example, the complexity of resource transition will influence the structure of call sequence and thus, the number of transition reliability and resource reliability factors can cause a significant deterioration in the reliability of the overall system. Therefore, the greater the complexity of the call sequence, the more accurate

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

will be the reliability estimation percentage by considering the in relation between resource reliability and task reliability in an estimation model.

Table 6. *Comparison with path-based approach*

Approach	% Accuracy (AC_{actual})			
	CS1 **CC=1**	CS2 **CC=3**	CS3 **CC=4**	CS4 **CC=6**
RDG	89.66	91.77	99.24	97.66
ARAUPT	93.62	94.68	97.88	97.26

5. DISCUSSION

Based on our strategy, the experiment was conducted with the goal of showing how the system complexity influenced the reliability level of the system. To achieve the goal, we divided the experiment into two strategies, namely: 1) acceptable degree of estimation accuracy and, 2) influence of scenario complexity against reliability.

As for the first strategy, the results in Table 5 show an acceptable level of estimation accuracy as compared with actual reliability model from the selected case study. The proposed model was then tested with the four different call sequence as shown in Figures 6 and 7. The result shows that the accuracy metrics (AC) carry a different weight of values. In this case, the AC values are still acceptable compared to actual reliability, as well as with other models. The argument is that the calculation of the subsequent resource reliability is correlated with previous resources that have been visited along the path of the call sequence. Besides considering only path reliability, our model has proved that the correlation with resource reliability (assuming this is through task transition) can increase the accuracy of reliability estimation. This occurs since the reliability of the latter is dependent upon the former.

In relation to the second strategy, the motivation is to compare the proposed estimation model with the most similar model (ARAUPT). The result obtained was concerned more with the level of estimation accuracy as compared with actual reliability value. From the result in Table 6, the RDG model recorded higher accuracy for CS3 and CS4 respectively, as compared with ARAUPT model. However, the AC for CS1 and CS2 for ARAUPT model are higher than for the RDG model. The justification for this is that the calculation of the current resource reliability is correlated with previous resources that have been visited along the path of the call sequence. Besides considering only path reliability, the RDG model has proved that the correlation with resource reliability (assuming this is through task transition) can increase the accuracy of reliability estimation. This occurs since the reliability of the current resource is dependent upon the previous resource, along the call sequence.

However, an interesting fact has been discovered during the experiment in terms of call sequence complexity. In relation to the investigation of the influence of call sequence complexity against the estimation accuracy, it seems that the RDG model produced a more accurate AC for call sequence CS3 and CS4 as compared to the ARAUPT model. Accordingly, the complexity of the system can influence its reliability. In this case, the complexity for each call sequence is calculated. Based on results shown in Table 6 for the RDG model, the AC values for CS1 and CS2 were recorded as slightly lower (89.66% and 91.77%) than those of the ARAUPT model, being 93.62% and 94.68% respectively. By contrast, the CS3 and CS4 produced by the RDG model obtained a much higher AC value, 99.24% and 97.66% as compared with that of the ARAUPT model, 97.88% and 97.26% respectively. This is due to the fact that the RDG model omitted the looping structure so as to avoid infinite behaviour that could lead to uncertainty of reliability estimation. Therefore, it is suggested that the higher complexity value for call sequence can be made as a reference for approximation of reliability estimation. The greater the complexity of the call sequence, the more accurate will be the estimation results, which are produced by the RDG model.

Aggregating this together, the RDG model has shown its ability to estimate the reliability of the system in line with actual reliability values and other models in terms of the acceptability of the estimation accuracy. On the other hand, by comparing the most similar model, the RDG model was shown to be able to handle a more complex scenario due to its ability to omit the looping structure and aggregation of transition and resource reliability for estimation formulation. Finally, the RDG model engaged with the system task in each resource. It is more logically sound if the system task parameter is visible at each resource. With this

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

parameter, in the future, the sensitivity analysis can be conducted so as to determine which task or resource could affect the level of reliability of the system. In addition, identification pertaining to what will affect system reliability through sensitivity analysis is also needed. Therefore, it is reasonable to determine why a system task is engaged with sufficient resources so as to easily perform sensitivity analysis in order to locate which system task, resources and task transition will contribute to the level of system reliability.

6. CONCLUSION

This paper has delivered one significant contribution, namely, a reliability estimation model. The reliability estimation model is called the RDG model. This model was designed using a graph-based approach and provides estimation for system reliability based on three parameters, specifically: resource, task and task transition. Furthermore, the model has been validated using the benchmark case study and the result proved useful for two types of quality, in particular: estimation accuracy and complexity. The proposed model is capable of estimating the system reliability with an acceptable accuracy level when compared with the existing models (ARAUPT, LCBRM, CUORM). On the other hand, the proposed model also has shown the capability to accurately estimate reliability of most complex call sequences as compared with ARAUPT models. This model is compared with the proposed models because they share the same paradigm and method for reliability estimation. The rationale behind this contribution is twofold, namely: an improvement of estimation accuracy and ability to motivate model consistency. It is suggested that the proposed estimation model can improve the estimation accuracy for the system, especially for a complex scenario. This is due to the fact that the proposed model considered the reliability of all the relevant resource reliability factors prior to the current resource reliability calculation.

REFRENCES:

[1] R. Pietrantuono, S. Russo, and K. S. Trivedi, "Software Reliability and Testing Time Allocation: An Architecture-Based Approach," *IEEE Trans. Softw. Eng.*, vol. 36, no. 3, pp. 323–337, May 2010.

[2] D. Hong and T. Gu, "A UML Model Based White Box Reliability Prediction to Identify Unreliable Components," *Softw. Integr. & Reliab.*, 2011.

[3] M. R. Lyu, "Software reliability engineering: A roadmap," in *2007 Future of Software Engineering*, 2007, pp. 153–170.

[4] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: a survey," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 295–310, May 2004.

[5] A. Immonen and E. Niemelä, "Survey of reliability and availability prediction methods from the viewpoint of software architecture," *Softw. Syst. Model.*, vol. 7, no. 1, pp. 49–65, Jan. 2007.

[6] S. S. Gokhale, W. Eric Wong, J. R. Horgan, and K. S. Trivedi, "An analytical approach to architecture-based software performance and reliability prediction," *Perform. Eval.*, vol. 58, no. 4, pp. 391–412, Dec. 2004.

[7] K. S. T. Goseva-Popstojanova, K, "Architecture-based approach to reliability assessment of software systems," *Perform. Eval.*, vol. 45, no. 2–3, pp. 179–204, Jul. 2001.

[8] A. Filieri, C. Ghezzi, V. Grassi, and R. Mirandola, "Reliability Analysis of Component-Based Systems with Multiple Failure Modes," pp. 1–20, 2010.

[9] R. Reussner, "Reliability prediction for component-based software architectures," *J. Syst. Softw.*, vol. 66, no. 3, pp. 241–252, Jun. 2003.

[10] C. Liu, J. Zheng, and Z. Ren, "An improved reliability model based on software architecture," *2010 2nd Int. Conf. Comput. Eng. Technol.*, pp. V7–218–V7–223, 2010.

[11] T. Harju, "Graph theory," *LectureNotes*, 2011.

[12] S. S. Gokhale and K. S. Trivedi, "Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework," *IEEE Trans. Reliab.*, vol. 55, no. 4, pp. 578–590, Dec. 2006.

[13] K. Tyagi and A. Sharma, "Reliability of component based systems: a critical survey," *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 6, pp. 1–6, 2011.

[14] M. Palviainen, A. Evesti, and E. Ovaska, "The reliability estimation, prediction and measuring of component-based software," *J. Syst. Softw.*, vol. 84, no. 6, pp. 1054–1070, Jun. 2011.

[15] S. Yacoub, B. Cukic, and H. H. Ammar, "A Scenario-Based Reliability Analysis Approach

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

for Component-Based Software," *IEEE Trans. Reliab.*, vol. 53, no. 4, pp. 465–480, Dec. 2004.

[16] S. Bernardi, J. Campos, and J. Merseguer, "Timing-Failure Risk Assessment of UML Design Using Time Petri Net Bound Techniques," vol. 7, no. 1, pp. 90–104, 2011.

[17] M. A. Isa, D. N. A. Jawawi, and M. Z. M. Zaki, "Model-driven estimation approach for system reliability using integrated tasks and resources," *Softw. Qual. J.*, Jun. 2013.

[18] C. Lauer, R. German, and J. Pollmer, "Fault tree synthesis from UML models for reliability analysis at early design stages," *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 1, p. 1, Jan. 2011.

[19] S. Distefano, M. Scarpa, and A. Puliafito, "From UML to Petri Nets: The PCM-Based Methodology," *IEEE Trans. Softw. Eng.*, vol. 37, no. 1, pp. 65–79, Jan. 2011.

[20] S. Mohanta, G. Vinod, a. K. Ghosh, and R. Mall, "An approach for early prediction of software reliability," *ACM SIGSOFT Softw. Eng. Notes*, vol. 35, no. 6, p. 1, Nov. 2010.

[21] S. Joseph, "A Model for Reliability Estimation of Software based Systems by Integrating Hardware and Software," *Sci. Technol.*, pp. 26–29, 2011.

[22] S. M. Yacoub and H. H. Ammar, "A Methodology for Architecture-Level Reliability Risk Analysis," *IEEE Trans. Softw. Eng.*, vol. 28, no. 6, pp. 529–547, 2002.

[23] X. Su, H. Liu, Z. Wu, X. Yang, and D. Zuo, "SA based software deployment reliability estimation considering component dependence," *J. Electron.*, vol. 28, no. 1, pp. 118–125, Jul. 2011.

[24] V. Garousi, L. C. Briand, and Y. Labiche, "A UML-based quantitative framework for early prediction of resource usage and load in distributed real-time systems," *Softw. Syst. Model.*, vol. 8, no. 2, pp. 275–302, Aug. 2008.

[25] J.-H. Lo, C.-Y. Huang, I.-Y. Chen, S.-Y. Kuo, and M. R. Lyu, "Reliability assessment and sensitivity analysis of software reliability growth modeling based on software module structure," *J. Syst. Softw.*, vol. 76, no. 1, pp. 3–13, Apr. 2005.

[26] C. Hsu and C. Huang, "An Adaptive Reliability Analysis Using Path Testing for Complex Component-Based Software Systems," vol. 60, no. 1, pp. 158–170, 2011.

[27] R. C. Cheung, "A User-Oriented Software Reliability Model," *IEEE Trans. Softw. Eng.*, vol. SE-6, no. 2, pp. 118–125, Mar. 1980.

[28] S. S. Gokhale, "A simulation approach to structure-based software reliability analysis," *IEEE Trans. Softw. Eng.*, vol. 31, no. 8, pp. 643–656, Aug. 2005.

[29] K. Go!seva-Popstojanova and S. Kamavaram, "Assessing uncertainty in reliability of component-based software systems," *Softw. Reliab. ...*, pp. 307–320, 2003.

[30] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Trans. Softw. Eng.*, vol. 23, no. 12, pp. 736–743, 1997.

[31] T. J. McCabe and C. W. Butler, "Design complexity measurement and testing," *Commun. ACM*, vol. 32, no. 12, pp. 1415–1425, Dec. 1989.

[32] H. Okamura and T. Dohi, "Software Reliability Growth Model with Normal Distribution and Its Parameter Estimation," *Framework*, 2011.