Physical Review & Research International
4(1): 231-245, 2014

SCIENCEDOMAIN *international*
www.sciencedomain.org

Component Criticality Analysis: An Efficient Approach towards Minimizing the Risks of System Software Failure

Md. Mesbah-Ul-Awal[1*], Muhammad Sheikh Sadi[1] and Saikat Das[1]

[1]Department of Computer Science and Engineering, Khulna University of Engineering and Technology, Bangladesh.

Authors' contributions

This study was carried out in collaboration with all the authors. Author MSS is the honorable man who guided and supervised the work and designed the study materials and an author of this work. Author M-U-A has managed all literatures, studied existing works, analyzed through the systems and wrote the manuscript. Moreover, author SD contributed the study by implementing simulation platforms and performing different analytical operations. The final manuscript is read and approved by all authors.

Research Article

Received 31st March 2013
Accepted 24th August 2013
Published 16th October 2013

ABSTRACT

Component based approach mitigating the risk of system failure has been proposed by detecting of the most critical components which's malfunction leads the software system towards failure and refactoring them. Individual components have their own chances of occurring fault and these occurrences are silent most often as well as risky; the probability of failure also becomes high in such phenomena resulting large amount of wretchedness. Protection of components from being faulty can be ensured at the early phase of any structure design or modeling, if the criticality is measured previously. A remarkable number of risk minimization approaches have overlooked criticality consideration in component level which is pursued in this study. Criticality is determined in a significant way by measuring each component's complexity and considering meaningful ranking of components based on random error injection and analyzing failure modes as well as corresponding effects. Design Mode Fan in-Fan Out, Inter Component Variable Passing Rate (ICVPR) and Average of Variable Exchange (AVE) calculation have been incorporated for finding complexity. Component criticality measurement has been carried out and compared among components within the system. Redressing system software

[*]Corresponding author: Email: mesbah_ul_awal@yahoo.com;

failure only by complexity or severity measurement couldn't bring out satisfactory consequences in real time reactive scenario. However, maneuver of refactoring critical segments could be a way of deployment.

Keywords: Component criticality analysis; complexity analysis; severity ranking; fan in-fan out; ICVPR; risk minimization; system failure; FMECA.

1. INTRODUCTION

Any critical infrastructure needs to be designed not only for safety and efficiency but also for the risk minimization capability by measuring criticality so that it may maximize the survivability of the system during extreme events such as system failure, or any malicious events [1]. This paper explores component criticality analysis which is validated by automated system: Traffic Control System. Two modes of the system are analyzed here such as – design mode and execution mode. The criticality based approach facilitates the system designers by providing a simple means of estimating the sensitivity of different components according to the probability of being affected by errors. Some approaches may have error minimizing capability but it is much important and substantial to be aware of system failure during system design level and act so by partitioning the system structure into components, defining their risk measurement rules and thus ranking the critical components. This ranking approach allows the designers to prioritize components according to their criticality which will help to identify and mitigate the threats by ensuring improved system reliability. For any software application this would be a generalized approach having methodology consists of designing the system architecture and components such a way that they are capable of measuring complexity and determining severity to find criticality. The rate of complexity determines the inter-component and intra-component dependency and severity measurement classifies the components according to their impact on the system. Therefore, this modern methodology towards risk minimization has been demonstrated by a practical benchmark among critical components analyzing their various system impacts applied in both design and execution mode, discussed in this paper.

2. RELATED WORKS

There are so many researches regarding risk minimization of system failure. Most of these researches assume that system component fails independently nevertheless components failure not only affect other components but also vary according to their criticality level [2]. For the both back and front-end engineering it would an improved approach to minimize the risk of system failure if the component criticality evaluation is ensured during software system design phase. In the field of military tactics the importance of critical components identification is well established [3]. There are other approaches to the development of safety-critical systems that may including compliance to DOD-178B for good development practices, use of commercial off-the-shelf software components for front end engineering and different forms of software testing to detect threats. But criticality consideration by measuring complexity and severity can be a convenient deployment towards minimizing the risk of failure.

Playing with multi-level systems, ranking components over utilization, performance, importance to the entire system performance and capability etc have always been a great outcome. Depending on the objective of the ranking, one or more criteria could optionally be

selected to rank additional components. Different components may have different performance indices also, different utilizations and different impacts on the system performance when these elements are removed from service [4], [5]. However, complexity measurement was not integrated though, but meaningful credential of reliability assessment has been explored.

Component based analysis and development must be augmented with methods to restrain complexities that arises inherently or due to resource constrains. So component detection and verification is the basic challenging part of any component based criticality analysis. System structure should have a simplified design so that components understandability is maximized and the criteria can be defined for component validation [6]. It would enhance the design mode of any system to track events or any error tracking if occurred. If the code structure has the facility to insert or delete traced code then it is advantageous for the system to minimize the risk of failure by detecting the erroneous component and deleting or replacing that code. To enhance system reliability assessment, component ranking is achieved by component testability model in criticality based analysis [7]. The FMECA method of analysis covers the evaluation of failure modes, system-level effects, weighting criticality and likelihood of occurrence with impact's diagnosis of system failure. For ordering the risks coherent with component functionality, FMECA++ could also be pursued [8]. The weight of components, based on perceived criticality and likelihood of failure with the average detection feasibility score is suggested but practical erroneous simulation is also expected. It would be more efficient approach if the components can be ranked by studying the practical erroneous phenomena as the vulnerability significantly increasing the failure rate for advanced application components. There may be error in the code or data of the system structure referred by [9] which can lead the system to fail if the corresponding component cannot be identified. If identified then risk minimization operation can be applied and it would be the feasible consideration.

A scenario based approach for reliability analysis of software-based component [10] feels the importance of complexity in enhancing the criticality measurement of any system. Again, complexity measurement as a risk assessment methodology is agreed in few cases [11], [12] but not for dynamic environments and assuming a static value for complexity. One way of dealing with the problems of increased complexity in the design is component-based system development which promotes compositional development and component reuse [11]. However, the use of commercial-off-the-shelf components (COTS) in safety-critical system is highly unexplored. There are no obvious methods to incorporate knowledge about functions and resilience of COTS in safety assessments. By measuring complexity inter-component dependency and how much a component can have chance to be affected by errors is determined which results an improved measurement towards finding criticality. When approaches based on the system design model augmented with formal failure models, it is one sided in context of increased complexity because it does not deals with the matter of how a component behaves in its execution environment. However, in real time reactive scenarios, only structural complexity is not sufficient to represent several possible failures in components, computational dimension is also a key factor. In a nutshell, a risk minimization model is proposed based on the meaningful consideration of component criticality achievable by measuring complexity and severity.

3. A METHODOLOGY TO MINIMIZE THE RISKS OF SYSTEM SOFTWARE FAILURE

The factors contributing to the complexity of such embedded systems are resource constraints, safety-criticality, concurrency, and the time-dependent functionality required of the artifacts they control. The complexity of system requirements permeates into several stages of development, that type of complexity is suggested to be resolved at early stages. Complexity of real-time reactive system can be viewed in five dimensions: *representational*, *architectural*, *internal*, *functional*, and *computational*. In software systems, the complexity is generally calculated by the measurement of time and space and may arise from resource limitations. But here, we regarded complexity as a probability of being affected by error or any unwanted malfunction [13]. The components may be functions that posses certain properties or behavior to have control over the system. Complexity does not measure the impact of systems component failure in system functionality but it may show the rank of components according to their impact if any component experiences error. The measurement of impact over the system due to any component failure is termed as severity. It is the measurement of finding how much a system is affected when individual component has errors and results system failure. In this system methodology of measuring complexity and severity for any single component is proposed along with the risk minimization formula. The combination of these two measurements gives criticality of that single component from an angle. If a system takes longer time to process any problem and giving solution then it can be said that the system is more complex. Also a system may have two types of complexity-Structural and Behavioral which are studied as well. Behavioral complexity is viewed in terms of how the software components interact through message passing and the complexity analysis does not measure the impact of components in system functionality; it accelerates defining the ranking criteria of erroneous component. If the erroneous behavior is increased then the complexity will also be increased. So, to minimize risk it is needed to identify the components that are suffering from error before the execution of the system.

Severity is the measurement of how much the system is affected when it is experiencing any error. In other words, when a component is affected by error then the measurement of the impact on the system is referred to as severity measurement. This error propagation observation is needed to determine the most critical components. Ideally, soft-error fault-tolerance properties have taken into account already at design time, selecting high-level source-code constructs, algorithms and data-type abstractions and representations as well to ensure integrative design methods properly [14]. When severity measuring, fault tolerance of a system is typically determined afterwards, through testing. Fault injection can be done first as it is a testing approach similar to mutation testing. In this analysis error has been injected randomly around the usage role of the targeted variable. It is claimed that, the usage role is a good determiner for soft-error impact, thus enabling improvements in terms of better test-case selection and prioritization. Different components comprising the system need a tight methodology also to evaluate their relative importance on the basis of the investigation of the critical components with improved dependability functionality. The dependability may be found by injecting errors in the system randomly and observing the impact on the system for this injection [15]. This random injection of error shows a fair result of system failure and this approach helps to implement a fair risk minimization way. In a summary, the methodology to analyze component criticality includes,

1) Measurement of complexity
2) Severity of failure
3) Minimizing the system failure

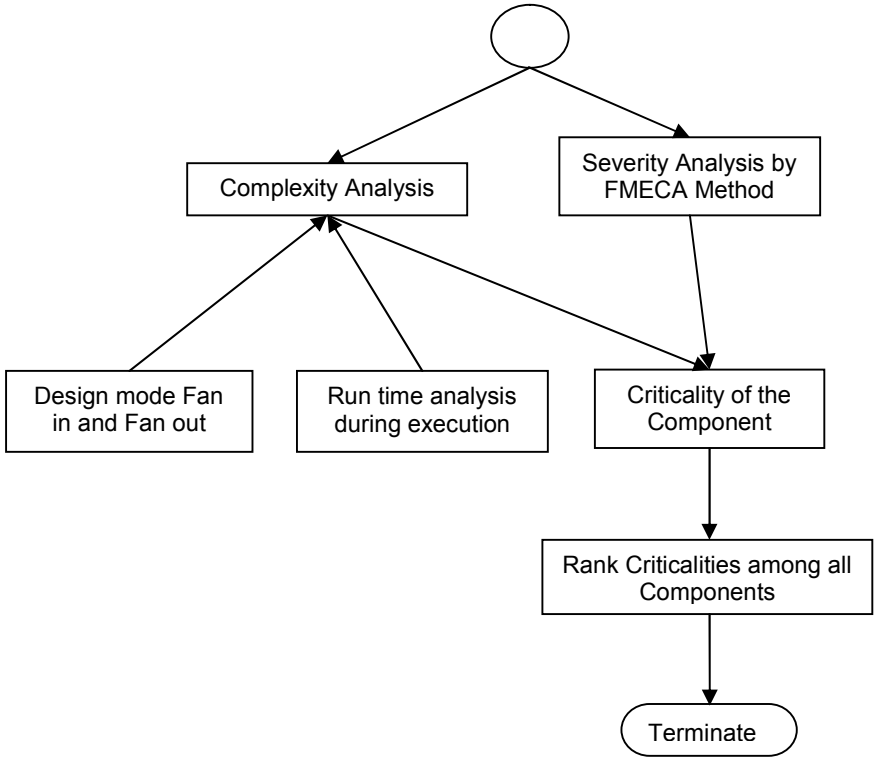


Fig. 1. Approach towards detecting critical components by measuring relative criticality

The combined result of complexity measurement and severity of failure is then taken as a measure of criticality. This complexity and severity determines how critical a system component is. In Fig. 1 it is shown that how this approach minimizes risks of system failure by calculating their criticality in an efficient way based on components.

3.1 Measuring Component Complexity

In component based analysis, representational and cognitive complexities can be met by choosing a good notation to describe components by which the component is defined exactly. Visual representations help to understand the non-linearity in component interactions. Any component can have two different view points on the basis of frame and architecture. The black-box defines the interface types and the particular grey box view is a structured implemented version of frame [13]. Here only two modes of complexity measurement are considered, though inclusion of other modes might be possible. However, in the design mode fan-In and fan-Out of each component is calculated which refers the entering and leaving variables among the components. On execution mode the number of variables that are being exchanged among the components is calculated during a certain period of time covering measurement of time-dependent functionalities. After that the average frequency of each component is calculated and added it with design mode fan-In fan-out of corresponding components.

3.1.1 Fan in and fan out

To measure the relationships between files and between components the design mode Fan-in and Fan-out values are calculated. Here Fan-in and Fan-out used to measure their complexity of the static structure of the code. It is treated graphically and visualizes components as nodes and Fan-in is the number of links coming into a component and Fan-out is the number of arrows going out of a node. So, for design mode,

Fan-in (component) = number of components that call it,
Fan-out (component) = number of components this component calls.

If the value of Fan-in is high then it indicates a heavily called component where low Fan-out value indicates less dependency on other components. A high Fan-out indicates strongly coupled code which means more complex to execute and test. The measurement of these values gives decision of being a component to be complex.

3.1.2 Calculating inter component variable passing rate (ICVPR)

The quantities of variables that are being exchanged among the components are calculated for a certain time interval and obtain the total variable passing amount of each component. If $COMP_{(i)}$ and $COMP_{(i+1)}$ be any two states of a component such as,

$$COMP_{(i)} \rightarrow COMP_{(i+1)} \tag{1}$$

Then the total amount of variable passed among each component is calculated for a certain time interval. In each time interval variable passing amount may vary because of different variable call in or out on that instant of time interval. If *ICVPR* is expressed in equation, it becomes,

$$ICVPR = \frac{variable\ \ exchange\ \ in\ \ COMP_i \rightarrow COMP_{(i+1)}}{observation\ \ time\ \ interval} \tag{2}$$

So this experiment is repeated for several time intervals and measured variable passing rate.

3.1.3 Average of variable exchange (AVE)

Previously the variable passing rate among the components is calculated based on several time intervals. Here's one thing should be mentioned that inter component variable passing is treated as the message in and out among the components. Average of Variable exchange and share can be determined by Equation 3 given below,

$$AVE = \frac{\sum_{i=1}^{n} ICVPR}{n} \tag{3}$$

So getting various variable passing rates, then average them with respect to the total no of occurrence. Finally measuring the complexity of each component the fan in-fan out of design mode and the average value of variable in-out on execution mode is simply added.

3.2 Measuring Severity of the Components

A single soft error in a particular component could have a greater effect than multiple soft errors in another or a set of components. For this reason, the effects of soft errors in the whole system should be analyzed by injecting transient faults (which will create soft errors if activated) into each component. These results are merged with the component's complexities to obtain a better measure of their impact on the software system if it is affected by soft errors.

3.2.1 Injecting errors externally

To find the severity of a component, it is decided to inject external error to view the performance of the system and thus can easily identify the component where a severe system failure occurs and operations have been performed into component level. By injecting error on different components separately, the system can easily overviewed whether severe failure occur or not for that erroneous component. To inject error in the simulator system code is translated into binary and injected malfunction exclusively. In context with the binary mode error injection, static variables in the component structure having real time values are also mistuned to observe the rapid effect in the system. Error injection operation has been performed in both black box and grey box testing fashion and the afterward impacts have been taken into measure to compute severity.

3.2.2 Failure modes and effects criticality analysis (FMECA)

Failure mode, effects, and criticality analysis (FMECA) is a bottom-up, inductive analytical method which may be performed at either the functional or piece-part level. This proposed model has been operated, analyzed and experimented into component level of the system software. FMECA extends FMEA by including a criticality analysis, which is used to chart the probability of failure modes against the severity of their consequences. The result highlights failure modes with relatively high probability and severity of consequences, allowing remedial effort to be directed where it will produce the greatest value. After that during the system definition phase of analysis, system has been partitioned into several modules fulfilling component criteria and they are defined such as Traffic module, Vehicle module, Alarm module, Car crash control module.

Before detailed analysis takes place some assumptions are defined to categorize the result of severity. According to the assumption the components are set weight. Failure Mode Identification is an important issue to find out failure mode and failure mode ratio. Failure can be identified by if there is any untimely operation occurs, loss of output, erroneous or invalid output. FMECA usually involves very large data sets; a unique identifier is assigned to each item and to each failure mode of each item. Now, based on the consequences of effect components are assigned severity classification for each failure mode of each unique component. Here is a small set of classification of severity similar to [16] in Table 1,

Table 1. Severity categories according to impact

SL no	Severity level	Rank	Impact on the system
1	Catastrophic	10	Rapid car crash Occurs
2	Serious	9	Multiple car crash on same instances
3	Extreme	8	Single car crash on individual instances
4	Major	7	Predicting clash between two cars, one car does not stop
5	Significant	6	Traffic light delay is changed or no signal
6	Moderate	5	Traffic signal is not obeyed
7	Low	4	Frequently cars come not on random speed
8	Negligible	3	Speed breaker does not invoked
9	Very minor	2	Not reducing speed after getting alarm
10	No Effect	1	No alarm for high speed car

Eventually, a weighted value of each component is given with the help of prioritized category, referred severity

3.2.3 Measuring criticality

Criticality is measured by taking the product of component complexity and severity which is termed as modal criticality (C_m). After getting the complexity and severity of each component from the analysis, criticality of corresponding component is calculated. Therefore,

$$\textit{Modal Criticality, } C_m = (\textit{Complexity} \times \textit{Severity}) \quad (4)$$

After completing the criticality measurement of each component in different severe situations, qualitative probability level (Rank) and quantitative criticality is sorted thereby. Now decision could be taken to identify critical components and likelihood failure rank of the component for which the risk mitigation is desired.

3.2.4 Risk minimization of system failure

From criticality measurement system designers can make decision that whether there is any error in the system or not where to change to fix the problem. Criticality helps in these cases of predicting erroneous components. As much as the product of severity rank and complexity will increase, the risk of system failure is increased. This helps the designers to predict the malicious nodes of the system where criticality is to minimize as well as changing the malicious affecting codes. This would be a structural approach which will redesign the system architecture according to the components criticality level. As a result inter-component dependency in other terms, Fan-in and Fan-out would be changed which will reduce execution time variable exchange rate as well as reduce the total criticality. So, for the improved redesigning paradigm Refactoring is a suitable approach. The proposed risk minimization approach based on experimental analysis has been shown in Fig. 2.

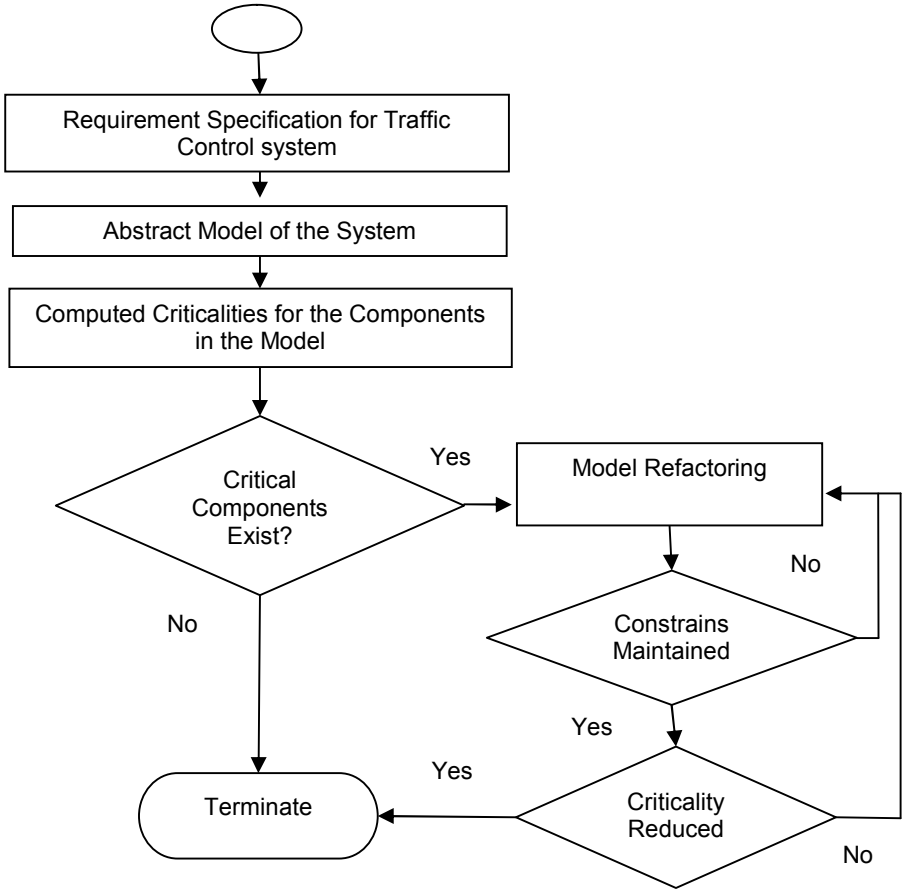


Fig. 2. Steps to minimize the risks of system failure by refactoring

4. EXPERIMENTAL ANALYSIS

To implement this thesis a simulator of an automated traffic system is designed. It is an automated system where the analysis is accomplished. In accordance to this evaluation various components also designed to fulfill the system based on relevant criteria, such as - Traffic Module(TM), Alarm Module (AM), Vehicle Module (VM), Car Crash Control Module (CCCM). It is a safety critical system where Traffic Module(TM) controls traffic lights to run the vehicles in a systematic order. When each vehicle comes to the traffic point it checks whether it is GREEN or RED or YELLOW signals. RED signal forces to be stopped at traffic point for 25 seconds and the YELLOW allows all vehicles to get prepared to stop or start within 15 seconds. If any vehicle acquires over speed closed to the traffic junction such that the system may feel risk of any accident then the Alarm Module generates an alert for that vehicle and the speed of that vehicle is lessen. Each vehicle generates random speed and run by this speed. As this system referred as a safety critical system there is another component here that always monitors each car with all other cars if there is any clash or not. It always checks the minimum distance between cars to avoid car crash as well as system failure. When the simulator is executed it has an interface looks like Fig. 3.

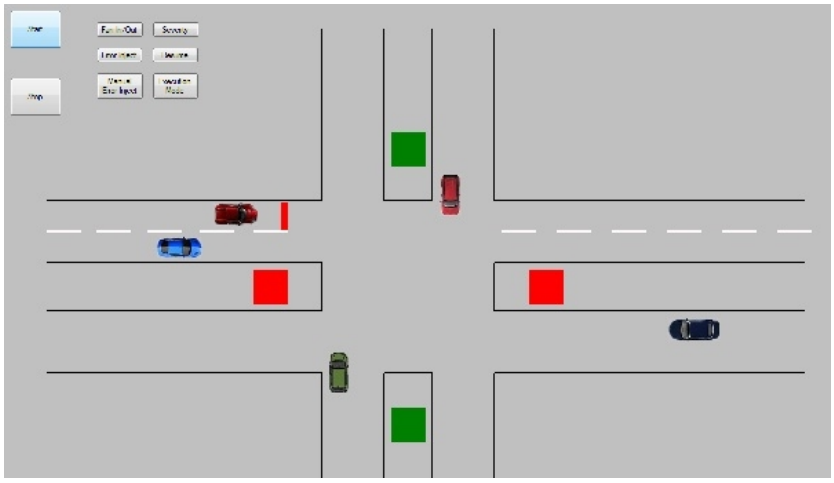


Fig. 3. Interface of designed simulator

4.1 Complexity Analysis

An interface of a component is an instance of either notifies-interface type or receives-interface type. The architecture is primitive if its structuring is to be provided in an underlying implementation (outside the scope of component specification language). A non-primitive architecture includes several subcomponents nested to several levels. The designed traffic system is shown in Fig. 4.

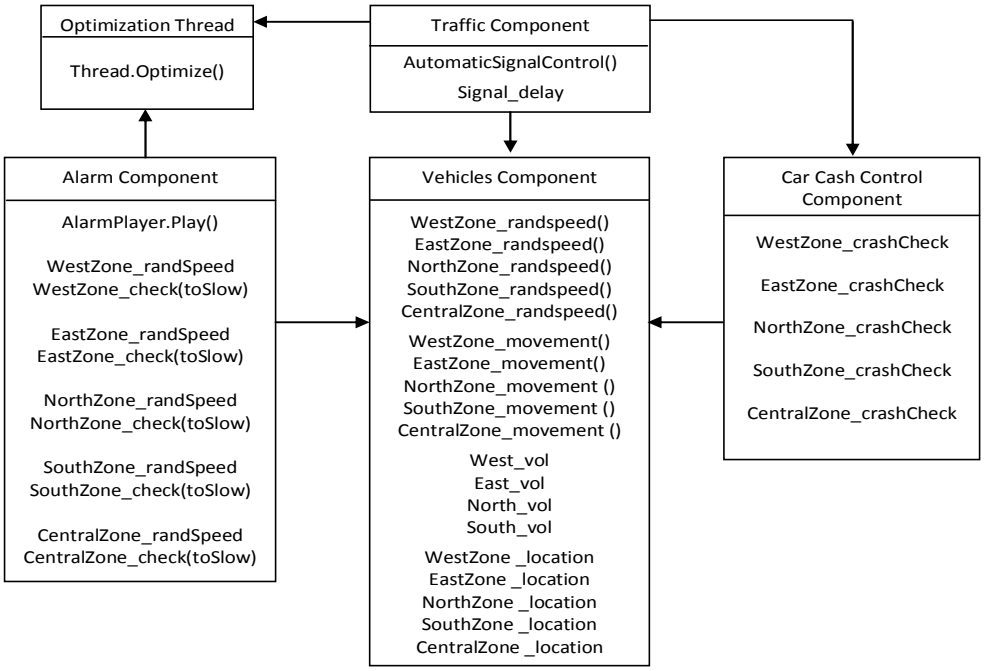
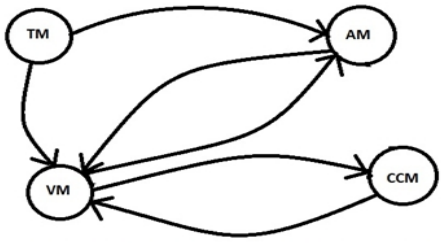


Fig. 4. UML class diagram of developed traffic control system

Since the class diagram is the main building block in object oriented modeling, It is used for both general conceptual modeling of the application and the model is then translated into programming code. Traffic component is the main class which coordinates the other classes. The Vehicle class describes the physical location of the objects having the highest level of dependency on the other classes. It also shows the count of vehicle passed over during a particular time. The other classes are Alarm and Crash Control where alarm class is accountable for generating system alerts such as over-speed alert, Traffic Signal Break alert etc. Accident control operation is performed by Car Crash Control class as well as it calculates total number of objects is passing through the road. Execution mode evaluation is performed considering Fan-in and Fan-out. But the efficient evaluation criterion is ensured when variable passing rate is determined in execution mode with respect to the execution time. These methodologies are applied in the Traffic Control System explained above. The four individual components have four individual behaviors to control the traffic system automatically. The dependency graph based on complexity of components is shown below in Fig. 5.



Here,
TM= Traffic Component
AM= Alarm Component
VM= Vehicles component
CCM= Car Crash Control Component

Fig. 5. Component dependency based on variable passing

ICVPR of each component has been calculated according to Equation 2. Here the experiment has been accomplished considering observation time interval is 5 minutes and reported accordingly. Similarly variable exchange among the components for 15 minutes also calculated. If the *ICVPR* measurement repeated for consecutive n times for the same time interval k and after that the average is calculated, it contributes a meaningful measurement of finding complexity of execution mode for each component achievable by using Equation 3. For each k=5 minutes and repeated n=4 times, here is the analysis result that is shown in the Table 2.

Table 2. Table of component complexity for k=5 and n=4

Components	ICVPR for 1st 5 minutes	ICVPR for 2nd 5 minutes	ICVPR for 3rd 5 minutes	ICVPR for 4th 5 minutes	AVE	Complexity
Traffic Module(TM)	80	80	88	80	82	0.00027
Alarm Module(AM)	371810	400886	240093	258644	317858.25	1.0595
Vehicle Module(VM)	30040	23247	23782	25120	25547.25	0.0851
Car Crash Control Module(CCM)	908	690	838	782	804.5	0.00268

Expressing the analyzed data in a graph the comparison can be visualized of each component during certain execution period of time k. The graph representation in Fig. 6 helps prioritize components according to their reliability quality. Comparison of variable passing rate of all modules with respect to different time period,

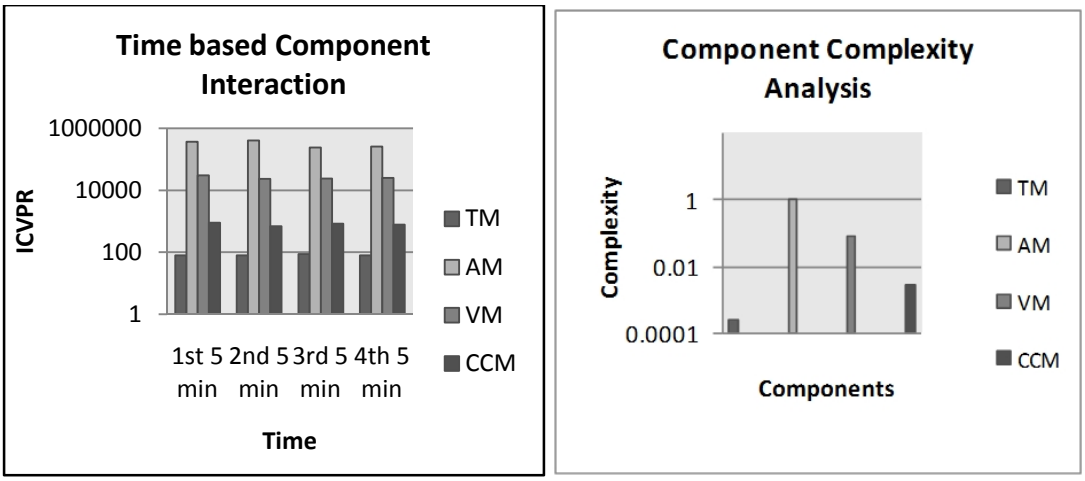


Fig. 6. Comparison of Complexity from different viewpoints

Therefore, it is a simple way of deciding and detecting the most critical component.

4.2 Severity Analysis

For the measurement of severity the Failure Mode and Effect Criticality Analysis (FMECA) method is used here. The measurement of severity determines that how much the system is affected when any error is occurred. A single soft error in a particular component can have the effect larger than the effect caused by multiple soft errors in any component. Here errors have been injected to each component to observe the impact due to error. After that these results are merged with complexities. This results a better measurement of their impact on the system. Here, those impacts are ignored which caused permanent damage to the system and the impacts by which the system experiences chaos are counted. Some errors

do not let the system to be executed as a result the system becomes unusable. So that such types of error are ignored. Based on the impact on the system components are weighted according to their severity. Weight of the components according to their severity after error injection is shown in the Table 3.

4.3 Criticality Analysis

In this analysis the modal component criticality is determined by measuring the product of complexity and severity as shown in Equation 4 and also system criticality is not taken into concern. For any increase in complexity there is a high probability that not only the severity will be increased but also total risks of system failure will be increased in proportionate to time. Hence, component criticality is taken as the product of overall complexity and severity. The experimental result is shown in Table 3 with practical benchmarks,

Table 3. Components Benchmarked Weight and Criticality

Components	Weight or rank	Criticality
Traffic Module(TM)	**5**	**0.00135**
Alarm Module(AM)	**8**	**8.476**
Vehicle Module(VM)	**7**	**0.5957**
Car Crash Control Module(CCM)	**10**	**0.0268**

5. CONCLUSIONS

This research is motivated by the fact of measuring individual component criticality of a system and establishing a general methodology that has ability to minimize the risks of a system failure. Several relevant cases have been studied for gathering knowledge to accomplish this research. This analysis covers significant sorts of risks minimization tasks that prevents software system components from occurring failure. Though error injection is a tough task, but it is handled here logically without breaking system consistency and appreciable advancement has been achieved. How much severe damage may consequent if any system software fails due to critical components have already discussed and given an improved way to get rid of disaster of system failure. Thus the criticality analysis represented in this paper will have great benefit towards minimizing risks of system failure. There is an open scope to extend this research by considering propagation of failure in computing criticality and merging the consequences with existing measurements. Though this part remains uncalculated here, a logical prediction of failure propagation is covered based on severity which leaded this research to decide most critical components successfully. The research could be extended to get increased level of dimensions in calculating complexity and minimum threshold level of the criticality could be maintained to achieve efficient refactoring.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Abdel-Rahim P. Oman, Johnson BK, Sadiq RA. Assessing surface transportation network component criticality: A multi-layer graph-based approach. IEEE Xplore. DOI: 10.1109/ITSC.2007.4357801. Accessed 3/10/2010.
   Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4357801.
2. Atef Mohamed, Mohammad Zulkernine. Failure type-aware reliability assessment with component failure dependency. IEEE Xplore. DOI: 10.1109/SSIRI.2010.12. Accessed 11/01/2011.
   Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5502850.
3. Hassan Reza, Steve Buettner, Varun Krishna. A method to test Component Off-The-Shelf (COTS) used in safety critical systems. IEEE Xplore. DOI: 10.1109/ITNG.2008.217. Accessed 09/01/2011.
   Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4492477.
4. Hamoud GA. Assessment of transmission system component criticality in the de-regulated electricity market. IEEE Xplore. Accessed 29/12/2010.
   Available:http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4912650&queryText%3DAssessment+of+transmission+system+component+criticality+in+the+de-regulated+electricity+market.
5. Ramirez-Marqurez JE, Coit. Composite importance measures for multi-state systems with multi-state components. IEEE Transaction on Reliability. 2005;54(3):517-529.
6. Hamoud G, lee L, Tonegouzzo J, Watt G. Assessment of component criticality in customer delivery systems. IEEE Xplore. ISBN: 0-9761319-1-9. Accessed 16/12/2010.
   Available:http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1378793&queryText%3DAssessment+of+component+criticality+in+customer+delivery+systems.
7. Jerry Gao, Ming-Chih Shih. A component testability model for verification and measurement. IEEE Xplore. DOI: 10.1109/COMPSAC.2005.17. Accessed 28/10/2010.
   Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1508112.
8. Baybutt M, Nanduri S, Kalgren PW, Bodden DS. Seeded fault testing and in-situ analysis of critical electronic components in EMA power circuitry. IEEE Xplore. DOI: 10.1109/AERO.2008.4526606. Accessed 8/01/2011.
   Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4526606.
9. Adam Piotrowski, Dariusz Makowski, Grzegorz Jablo´nski, Andrzej Napieralski. The automatic implementation of software implemented hardware fault tolerance algorithms as a radiation-induced soft errors mitigation technique. IEEE Xplore. DOI: 10.1109/NSSMIC.2008.4774657. Accessed 25/02/2011.
   Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4774657.
10. Yacoub SM, Cukic B, Ammar HH. Scenario-based reliability analysis of component-based software. IEEE Xplore. DOI: 10.1109/ISSRE.1999.809307. Accessed 4/02/2011.
    Avaiable: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=809307.
11. Elmqvist J, Nadjm-Tehrani S. Tool support for incremental failure mode and effects analysis of component-based systems. IEEE Xplore. DOI: 10.1109/DATE. 2008.4484792. Accessed 14/03/2011.
    Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4484792.
12. Ping Gao, Su Wu. Improved criticality analysis of railway locomotive components by simulation. IEEE Xplore. DOI: 10.1109/RAMS.2007.328132. Accessed 25/01/2011.
    Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4126401.

13. Zheng M, Alagar VS. Complexity of Component-based Development of Embedded System. World Academi of Science, Engineering and Technology. Accessed 12/02/2011.
Available: www.waset.org/journals/waset/v8/v8-138.pdf.
14. Cortellessa V, Grassi V. A modeling approach to analyze the impact of error propagation on reliability of component-based systems. Component-Based Software Engineering, Springer Berlin Heidelberg. 2007;4608:140-156. ISBN: 978-3-540-73550-2. DOI: 10.1007/978-3-540-73551-9_10.
15. Abdelmoez W, Nassar DM, Shereshevsky M, Gradetsky N, Gunnalan R, Ammar HH, et al. Error propagation in software architectures. IEEE Xplore. DOI: 10.1109/METRIC.2004.1357923. Accessed 16/03/2011.
Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1357923.
16. Seyed-Hosseini SM, Safaei N, Asgharpour MJ. Reprioritization of failures in a system failure mode and effects analysis by decision making trial and evaluation laboratory technique. Reliability Engineering & System Safety. 2006;91(8):872-81.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sciencedomain.org/review-history.php?iid=283&id=4&aid=2269