Assessment of Component Criticality with Proposed Metrics

Latika Kharb and Rajender Singh*
Research Scholar, Reader*
Department of Computer Science & Applications, M. D. University, Rohtak, Haryana, INDIA
latika.kharb@gmail.com

ABSTRACT
Typically, the development of component-based systems starts with a collection of existing components. As a component is typically developed in a system environment that is different from the environment of the final system where the component is integrated, so it is difficult to predict the component behavior in the new system. It is generally understood that building software systems with components has many advantages but the difficulties of this approach should not be ignored. Understanding the impact of COTS software component failures with respect to system safety is a crucial first step to the successful use of COTS software products in a safety-critical application. In order to use COTS functionality in a safety-critical application, it is imperative to examine how and to what extent the failure of the required COTS functionality can affect the safety of the system. Criticality analysis assesses the level of safety effect associated with individual failure modes of a component. Several factors can be used to determine the criticality of a failure mode of a component with regard to system safety namely, Hazard Severity Category, Failure Probability, Degree of Protection, Software Control Category. In this paper, we've proposed two metrics namely, Links Criticality Metric (LCM) and Size Criticality Metric (SCM). Along with this, some open problems in CBSE metrics and factors affecting the criticality are described. Finally, we concluded how our research is intended to measure the new features related to the introduction of the CBSD approach.

KEYWORDS
Component-based systems, COTS, safety-critical application, component technology.

OPEN PROBLEMS IN METRICS FOR CBSE
Recently, component-based software development is getting accepted in industry as a new effective software development paradigm [1]. Few companies or organizations can have the luxury of trying to develop a complex system from scratch; more and more software applications are being built using commercial-off-the-shelf (COTS) products. This shift towards COTS-based solution is largely due to its massive cost and time saving compared with the in-house developments. By taking into account safety concerns early at the selection stage, the proposed approach can potentially help with the subsequent integration of the selected COTS product and certification of the final system. We believe that, if the COTS acquisition process is properly managed, the “better, cheaper, faster” proposition may be achieved for the use of COTS software products in safety-critical applications. Depending on system planning criteria used, a single component failure may or may not affect the reliability of supply to customers. All components of the system are important and some components are more critical than others. Understanding the impact of the failure of a COTS software component with respect to key system properties (e.g. safety) is crucial to the successful use of COTS software products in a critical (safety-, security-, or mission-critical) application. Cost-effective software project management will focus resources towards intensive validation of those areas with highest criticality [2]. In a recent survey, several proposals for the quantitative assessment of components and assemblies have been identified, based upon their functional properties and several recurrent problems in those proposals, which are also common in metrics proposals for other purposes, such as:

(1) Lack of a quality framework occurs when a metrics definition is not framed by a particular quality model;
(2) Lack of an ontology occurs when the architectural concepts to be quantified, either of functional or non-functional nature, are not clearly defined, namely in their interrelationships;
(3) Lack of an adequate formalism occurs when metrics are defined with a formalism that requires a strong mathematical background, which is often not held by practitioners. This limitation hampers metrics usability.
(4) Lack of computational support occurs when metrics proponents do not produce tools for metrics collection, or when they do not make them available to other researchers and practitioners.
(5) Lack of flexibility occurs when metrics collection tools are available, but they are either proprietary, or, if they are open source, the metrics definitions are somehow obscurely tangled in the code. The latter hampers the assessment of the correction of their algorithm, as well as the ability to modify them.
(6) Lack of validation occurs when independent cross validation is not performed, mainly due to difficulties in experiment replication. Such validation is required before widespread acceptance is sought.

In this paper, we present an approach to mitigate these problems. We use the Component Criticality Metrics (CCM) as a representation for components.
RESEARCH PROBLEM: CRITICALITY OF A COMPONENT

Criticality Analysis examines the degree of contribution that each individual failure mode of a component has with respect to system safety and provides the basis for determining the required level of assurance for each failure mode of the potential COTS functionality. We relate our metrics suite with the subcomponents. Criticality analysis must assess the level of safety effect associated with individual failure modes of a component. Several factors can be used to determine the criticality of a failure mode of a component with regard to system safety. Component criticality analysis is a way of ranking the relative importance of each identified component level failure mode as to system safety. Recent research suggests that complexity measure/metrics are not adequate for a CBS as they focus mainly on either lines of code (LOC) or information based on object and class properties. There is therefore a need to develop a new technique for measuring the complexity of a component. The criticality of a component failure mode could be determined with reference to factors like:

• Hazard Severity Category
  The criticality of a component failure can only be determined with reference to the system hazard that can be caused by the failure. If a component failure cannot cause any system level hazard, or it can only cause system hazard(s) of low severity level (e.g. negligible), it can be said to have a low criticality level with respect to system safety. However, the severity of a system hazard alone is not sufficient to determine the criticality of a component failure, as failures of one or more other system elements may also be contributing factors to the occurrence of the system hazard. Therefore, the degree of contribution a component failure has towards the occurrence of the system hazard must also be evaluated in order to determine the criticality of the component failure.

• Failure Probability
  It’s the likelihood of the occurrence of a component failure. However, determining a software component’s failure probability, even qualitatively, can be very difficult if not impossible.

• Degree of Protection
  It’s the level of protective measures devised for a component failure that can result in a system level hazard. If the occurrence of the failure alone will lead to the system hazard, the severity of the system hazard would consistently reflect the failure mode’s criticality.

• Software Control Category
  It’s the degree of control that software exercises over the hardware. It measures the amount of control that software components have with respect to identified system hazards.

PROPOSED CRITICALITY METRICS FOR COMPONENTS

Critical component is to be identified as early as possible by software developer. For a software tester this component requires substantial testing effort. Every possible scenario for this critical component has to be tested, particularly if it is a base component, in which wrong operations can be inherited by the subcomponents. In this research, we propose a metric to identify the critical components but we use an approach to metrics definition and collection that is different from the one originally proposed by the original authors [4].

(I) Links Criticality Metrics (LCM)
Link Criticality is defined as a condition in which a component has many links to other components. For Links Criticality, we take average of the number of links and the number of connected components. A component will be called critical if the LCM has reached a certain threshold value.

\[
\text{Links Criticality Metrics (LCM)} = \frac{\# \text{Links}}{\# \text{Components}}
\]

Where
\[
\# \text{Links} = \text{Sum of the Number of Links} \quad \# \text{Components} = \text{Sum of the Number of Components}
\]

(II) Size Criticality Metrics (SCM)
Size Criticality is defined as a condition in which the size of component might induce a problem. The bigger software volume, the more error will be found during the development stage. For size criticality, we count the number of constituents and the thousand lines of code and take their average and check if the value has exceeded a certain threshold value.

\[
\text{Size Criticality Metrics (SCM)} = \frac{\# \text{Constituents}}{\# \text{KLOC}}
\]

Where
\[
\# \text{Constituents} = \frac{\text{LOC}}{\text{Number of operations/ components}} \quad \# \text{KLOC} = \text{Thousand Lines of Code}
\]

By knowing this critical number, a software developer will get a better understanding of which component face higher risk than other components, and then an optimized resource allocation can be deployed for that critical component. This critical number is an indication of the risk associated with the component.

EXPECTED BENEFITS

The primary objective of the criticality analysis is to assess the importance to safety of the software components within the COTS and to show there is segregation between software components with different safety importance [5]. For a software tester, critical component needs to be exhaustively tested. We can actually combine the packing and interaction density with our proposed critical metrics. The critical metrics is actually more component specific. Link Criticality and Size Criticality are used to locate which component has higher critical value than others. The proposed metrics could measure the criticality of the components. It is intended to aid CBSE developers by recognizing the criticality of the design; it would provide good information for developers to predict the effort to
Assessment of Component Criticality with Proposed Metrics

finish an integrated component. Software tester can have detailed information about which components should be stressed tested based on its criticality. Empirical evaluation of the metrics suite should be done so that the user confidence of the metric suite can be established. It is expected that the metrics can be used to:

• Find the critical and complex part of the component. The measurement result will indicate which component is critical than others, and which is more complex than others. Some threshold values could be used to indicate if a component needs more considerations than others;
• Measure the degree of the complexity and criticality of a system, and later on can be used to classify the software system;
• Derive a mechanism/formula to predict time and effort required to develop a system using integrated components from industry data.
• Gather some feedback on the metrics from software designers and testers in less than 6 months after they use the metrics.

The metrics proposed in this paper can be used to identify the criticality of the metrics. By recognizing a critical component, it should give a contribution on the effort and cost estimation. This information should help a software project leader to estimate better. The metrics suite can also be incorporated in a CASE (Computer Aided Software Engineering) tool that helps designers to build the software. A component-based CASE tool can include the metrics suite for helping the developer in designing the software. The developer can get an insight of which component has high complexity or risking the criticality. Adding the constraint in the proposed metrics could yield another method of measuring CBSE software development.

CONCLUSION
Proposed metrics should be further quantified and validated properly so that they can prove to be quite helpful in controlling and monitoring all-important activities of CBD. Besides, more component-based metrics should be proposed, qualified and validated. In this paper, we’ve proposed two metrics namely, Links Criticality Metric (LCM) and Size Criticality Metric (SCM). Along with this, some open problems in CBSE metrics and factors affecting the criticality are described. Finally, we concluded how our research is intended to measure the new features related to the introduction of the CBSD approach.

FUTURE SCOPE
Our metrics is still in our early research and much work on data collection has to be done to test our hypotheses. This research has proposed a set of metrics suite to be used in software component integration. We believe these metrics is based on a measurement theory, and has to be validated first with some properties of software metrics. We hope that these metrics would help component-based developer (and integrator) to identify the level of complexity and criticality of a component in an integrated software system. By understanding this level it can be used to predict time and effort. We expect by gathering real data and analyzing it, some useful results can be produced by using this metrics. A further study of criticality on software component metric should help provide a basis for significant future progress.

REFERENCES

Copy Right © INDIACom – 2008