Architectural Level Risk Assessment Tool Based on UML Specifications [1]

T.Wang, A. Hassan, A. Guedem, W. Abdelmoez,
K. Goseva-Popstojanova, H. Ammar
Lane Department of Computer Science and Electrical Engineering,
West Virginia University, Morgantown, WV26506-6109
{twang, hassan, guedem, rabie, katerina, ammar}@csee.wvu.edu

Abstract

Recent evidences indicate that most faults in software systems are found in only a few of a system's components [1]. The early identification of these components allows an organization to focus on defect detection activities on high risk components, for example by optimally allocating testing resources [2], or redesigning components that are likely to cause field failures. This paper presents a prototype tool called Architecture-level Risk Assessment Tool (ARAT) based on the risk assessment methodology presented in [3]. The ARAT provides risk assessment based on measures obtained from Unified Modeling Language (UML) artifacts [4]. This tool can be used in the design phase of the software development process. It estimates dynamic metrics [5] and automatically analyzes the quality of the architecture to produce architectural-level software risk assessment [3].

1. Introduction

There is an increasing need for a tool that can be used to track the quality of software products during the software design phase. Some of the proposed tools get static metrics from source code [6], but source code metrics are affected by the programming language and programming style. When calculating the metrics from architectural descriptions such as UML, we achieve independency of languages and human factors [7]. Other metrics tools [8] only produce static metrics to describe the model with a limited capability to accurately represent the dynamic behavior of the architecture. We developed a tool for risk assessment at the architectural level, ARAT, that produces the dynamic metrics from UML diagrams. Using the methodology proposed in [3], ARAT estimates risk factor of a software system in hieratical fashion. This paper is organized as follow. Section 2 presents an overview of the tool structure. Section 3 provides some examples of the outputs from the tool. Finally, Section 4 concludes the paper.

2. Structure of the tool

The UML use case model of the tool is shown in Figure 1. The tool consists of six main use cases as follows.

Estimate dynamic metrics

Based on cyclomatic complexity [10], the ARAT computes the dynamic complexity [5] of the UML state charts as a dynamic complexity metric. It also computes dynamic coupling between components based on the message exchange between components in the sequence diagram [5].

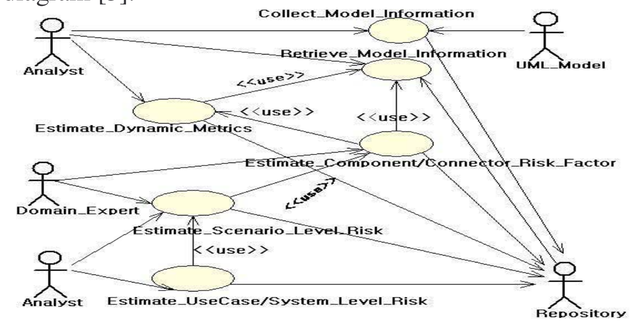

Collect_Model_Information
Analyst
Retrieve_Model_Information
UML_Model
<<use>>
<<use>>
<<use>>
Estimate_Dynamic_Metrics
Estimate_Component/Connector_Risk_Factor
Domain_Expert
<<use>>
Estimate_Scenario_Level_Risk
<<use>>
Analyst
Estimate_UseCase/System_Level_Risk
Repository

Figure 1. Use case diagram of the ARAT tool

Collect model information

Using a commercial tool Rose Real Time [9] as a front end for our tool we transform the visual UML model to a textual format data. We use Rose Script to transform UML diagrams to textual data.

Estimate component / connector risk factor

Based on the definition of risk [11], we calculate risk factors of each component (connector) in the architecture as

[1] This work is funded in part by grants to West Virginia University Research Corp. from the National Science Foundation Information Technology Research (ITR) Program grant number CCR-0082574, and from the NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP) managed through the NASA Independent Verification and Validation (IV&V) Facility, Fairmont, West Virginia.

a product of the dynamic complexity (coupling) and the severity level of a failure. The severity level of a failure of component/connector is estimated using FMEA [12].

Estimate scenario risk factor

The tool automatically constructs the Markov chain that represents the control flow graph of the active components and connectors in a specific scenario based on the textual representation of the UML sequence diagrams. The scenario risk factor for each severity level is computed using this Markov chain and the estimated values of component/connector risk factors [3].

Estimate use case and overall system risk factors.

The risk factors of each scenario in a specific use case are aggregated to calculate the use case risk factor [3]. Using the risk factor for each use case, the tool calculates the overall system risk factor [3].

3. Illustration of the tool outputs

Due to a space limitation we only illustrate some of the output results provided by the ARAT. These results are for the pacemaker [12] which is an implanted device that assists cardiac functions of the heart when the underlying pathologies make the intrinsic heartbeats low. Pacemaker is an example of a critical real-time application because the failure of the software operation of the device can cause loss of a patient's life. Figure 3 presents the identification of the critical components in the pacemaker example provided by ARAT. Thus, the components that have high risk factors with catastrophic severity in multiple scenarios are the most critical components that would require more careful development and/or more testing effort.

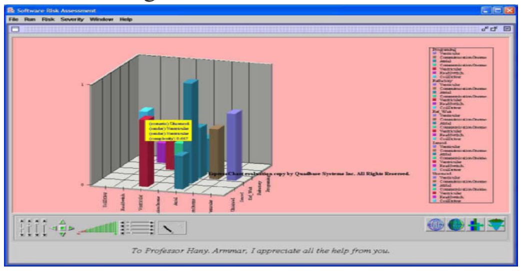
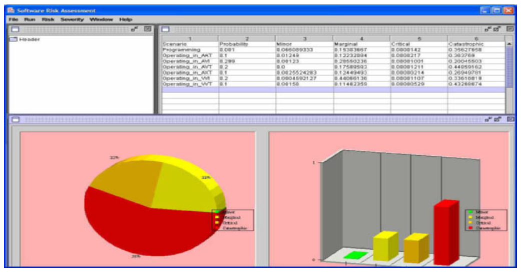

Software Risk Assessment
File Run Risk Severity Window Help
To Professor Hany Ammar, I appreciate all the help from you.

Figure 3. Identification of critical components

The distribution of the overall system risk factor among severity classes is presented in Figure 4. We see that the overall system risk factor is mostly distributed among marginal and catastrophic severity classes, which confirms that this is a high risk system.

4. Conclusion and future work

In this paper, we present ARAT, a tool for architectural level risk assessment based on UML specifications. The tool enables early assessment of risk and hence makes it possible for the analyst to identify critical components/connectors and scenarios/use cases early in the software lifecycle. The output of the tool can guide the allocation of development and testing effort based on critical use cases, scenarios, components, and connectors. Our future plan is to further extend the tool so that it computes static metrics, as well as to collect, store and analyze data which is used for interpretation of quality metrics, even though the result maybe not as sensitive and complete as dynamic metrics for early risk assessment. In addition, we plan to integrate the hazard analysis methodology into our tool to allow automatic and precise estimation of the severity level for each architectural element.



1	2	3	4	5	6
Scenario	Probability	Minor	Marginal	Critical	Catastrophic
Programming	0.001	0.066089333	0.15383667	0.0808142	0.35627658
Operating_in_AAT	0.1	0.01249	0.12232884	0.0808217	0.363769
Operating_in_AAI	0.299	0.08123	0.28560236	0.08081001	0.20045503
Operating_in_AVT	0.2	0.0	0.17589593	0.08081211	0.44859162
Operating_in_AAT	0.1	0.0825524283	0.12449493	0.08080214	0.26949781
Operating_in_VVI	0.2	0.0804692127	0.44066136	0.08081107	0.33616818
Operating_in_VVT	0.1	0.08156	0.11462359	0.08080529	0.43269874

Figure 4. Distribution of the overall system risk factor

5. References

[1] N. Fenton, N. Ohlsson, "Quantitative Analysis of Faults and Failures in a Complex Software System", *IEEE Trans. Software Engineering*, Vol. 26, No. 8, pp. 797 -814, 2000.

[2] W. Harrison, "Using Software Metrics to Allocate Testing Resources", *Journal of Management Information Systems*, Vol. 4, No. 4, 1988, pp. 93-105.

[3] K. Goseva-Popstojanova , A. Hassan, A. Guedem, W. Abdelmoez, D. Nassar, H. Ammar, A. Mili, "Architectural-Level Risk Analysis using UML", submitted for publication.

[4] J. Rumbaugh, I. Jacobson, G. Booach, *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.

[5] A. Hassan, W. M. Abdelmoez, R. M. Elnaggar, H. H. Ammar, "An Approach to Measure the Quality of Software Designs from UML Specifications," *7th International Conference Information Systems, Analysis and Synthesis*, 2001, Vol.IV, pp.559-564.

[6] M. Stojanovic, K. El-Emam, "ES1: A tool for collecting object-oriented design metrics", *NRC/ERB-1087*, May 2001.

[7] M. Hitz, K. Neuhold, "A Framework for Product Analysis", OOPSLA *1998 Workshop on Model Engineering*, Methods and Tools Interaction with CDIF, 1998.

[8] L. Nenonen, J. Gustafsson, J. Paakki A. Inkeri Verkamo, "Measuring object - oriented software architectures from UML diagrams", Proc. *4th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, 2000, pp. 87-100.

[9] UML Language Resource Center: Unified Modeling anguage, Standard Software Notation, http://www.rational.com.

[10] T. McCabe, "A Complexity Metrics", *IEEE Trans. Software Engineering*, Vol.2, No.4, 1976, pp 308-320.

[11] H. Ammar, T. Nikzadeh, J. Dugan, "A Methodology for Risk Assessment of Functional Specification of Software Systems Using Coherent Petri Nets", *Proc. 4th Inernationall Software Metrics Symposium,* 1997, pp. 108-117.

[12] S. Yacoub, H. Ammar, "A Methodology for Architectural-Level Reliability Risk Analysis," *IEEE Trans. Software Engineering*, Vol. 28, No. 6, 2002, pp.529-547.