An Overview of Technical Models for Dynamic Analysis

Radhika D. Venkatasubramanyam and Snigdha Nayak

Abstract—This paper discusses the Technical Models, collectively referred to as DATM (Dynamic Analysis Technical Models), designed to classify results from dynamic analysis tools. The proposed technical models help structure these findings. Assessment of internal code quality can get simplified by this systematic categorization. DATM is different from the existing models that express internal code quality in terms of the quality attributes. DATM can be used in code reviews using dynamic analysis tools to classify issues from a technical aspect.

Index Terms—Dynamic analysis, technical model, software quality, code review.

I. INTRODUCTION

Software Quality comprises of external and internal quality. External software quality refers to the conformance of the software to the predefined functionality, as seen by the end users. Internal software quality refers to the aspects of the software as observed by the programmers. It involves a check for adherence to good coding practices in turn ensuring that the code is bug-free.

With software evolving and getting pervasive in all areas, it becomes extremely important that the code in the software is bug free. Internal code quality can be effectively determined mainly through code assessments which involve both static and dynamic analysis of code. The issues reported by these analysis tools can at times be exhaustive and diverse and hence need to be classified.

As a part of code assessments, quality models are used to map issues reported by static, dynamic and metric tools. Software quality models assess the software in terms of the quality attributes. Categorization based on the quality model alone has high relevance for the management team since it provides an overview of the project quality and facilitates a comparative study of projects in terms of quality.

There is also a need for analyzing the tool findings with relevance to the code. It is required to categorize the issues reported by dynamic analysis tools to enable better understanding for developers and managers from a technical perspective. DATM is an attempt to address this need.

The motivation for proposing DATM is that such a model helps relate the quality model with the technical findings of DAISQ.

II. DYNAMIC ANALYSIS TECHNICAL MODELS (DATM)

Dynamic analysis involves identification of run-time issues affecting performance and memory usage for an application. As part of code reviews, dynamic analysis tools are used. In DAISQ, the tool findings are summarized in a report consisting of management and technical summary that can be referred by both managers and programmers.

A. DAISQ (Dynamic Analysis for Internal Software Quality)

DAISQ is a process to systematically assess internal software quality using dynamic analysis. The process complements static analysis. The methodology involves identification of run-time defects by executing the application under review on a selected workflow using dynamic analyzers.

B. DAISQ Quality Model

The DAISQ quality model is defined in accordance with the quality model defined by the ISO 9126 [3], [4] standard consisting of 5 different quality attributes on the top level - Maintainability, Reliability, Efficiency, Security and Portability [1]. Quality model consists of quality attributes, their sub-quality attributes and run-time error mapping to sub-quality attributes. Fig. 1 summarizes the DAISQ Quality Model.

C. Motivation for Technical Model

The existing DAISQ quality model provides a quality perspective which is mainly useful to the management team. It is also required to have a technical classification that aids the programmers to resolve the identified issues. Through a technical model, programmers can relate the issues and categorize them in technical terms.

The motivation for proposing DATM is that such a model helps relate the quality model with the technical findings of DAISQ.

Manuscript received December 13, 2012; revised March 25, 2013.

R. D. Venkatasubramanyam is with System Development and Test Research Group at Corporate Research & Technologies India, Siemens Technology and Services Private Limited (e-mail: radhika.dv@siemens.com).

S. Nayak is with System Development and Test Research Group at Corporate Research & Technologies India, Siemens Technology and Services Private Limited (e-mail: snigdha.nayak@siemens.com).

DOI: 10.7763/LNSE.2013.V1.36
D. Overview of Proposed Technical Model

DATM refers to the technical classification of run-time issues detected by dynamic analysis tools and is represented in the form of a model to be used for categorization of these issues. The basis for categorization is discussed below.

In order to formulate the model, a bottom-up approach, of grouping findings from various dynamic analysis tools into sub-technical attributes and further into technical attributes, is used. For this, issues reported by various dynamic analysis tools were identified. Based on these issues, sub-attributes were identified. Issues which were highly relevant to dynamic analysis and common across tools were retained. A high-level categorization into attributes was then done by grouping sub-attributes.

A generic model of DATM is represented in Fig. 2.

In DATM, technical attributes represent the first level of grouping of technical findings from dynamic analysis tools.
- Sub-technical Attributes
- Technical Issues

The sub-technical attributes form the second level of classification in DATM.
- Technical Issues

The results obtained directly from dynamic analysis tools are termed as technical issues and help in identifying potential errors in the application.

To summarize, technical attributes comprise of sub-technical attributes and sub-technical attributes comprise of technical issues. Fig. 3- Fig. 5 provide a summarized representation of the technical models for the three languages: C#, Java and C++. The model was created based on issues reported by a set of identified dynamic analysis tools. The tools were selected based on their suitability to analyze client projects. For C++, DevPartner [5], AQTime Pro [6], Rational Purify and Quantify [7] were considered. For C#, Yourkit .NET Profiler [8], AQTime Pro, DevPartner and dotTrace [9] and were considered. For Java, Yourkit Java Profiler [10], Eclipse Memory Analyzer [11], Eclipse TPTP [12] and JProfiler [13] were shortlisted.

III. Application of DATM

DATM is planned to be used in reports obtained by applying DAISQ on projects from various product development divisions of Siemens. It can be used to provide a technical perspective to the findings obtained from dynamic analysis tools.

TABLE I: MAPPING OF TECHNICAL ATTRIBUTES TO QUALITY ATTRIBUTES FOR C#

<table>
<thead>
<tr>
<th>Technical Attribute</th>
<th>Sub-technical Attribute</th>
<th>Technical Issue</th>
<th>Reliability</th>
<th>Efficiency</th>
<th>Maintainability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>Time measurement</td>
<td>CPU time</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wall time</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time spent with children</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time spent in method</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Thread time</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Performance bottlenecks</td>
<td>Frequency of function calls</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Execute hot spots</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exception handling</td>
<td>Exceptions</td>
<td>Unmanaged exception was handled by the CLR</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Large Object Memory usage</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Short medium and long lived objects</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Strongly connected objects</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Held objects</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Methods with most leaked memory</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Objects of most leaked memory</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Garbage Collection</td>
<td>Garbage Collection</td>
<td>Garbage Collection Event Initiated</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Language specific</td>
<td>.NET Issues</td>
<td>Base Class Dispose not called</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Finalizer Leak</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Missing Dispose method</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In addition it is important to relate these technical attributes to the existing quality model in order to complete the overview of code quality. Table I provides the mapping of C# DATM to Reliability, Efficiency and Maintainability quality attributes. These quality attributes were selected based on their relevance to the Siemens projects. Also, most of the findings reported by dynamic analysis tools mapped to the above quality attributes.

Example: In order to understand the technical model, consider the C# technical model in Fig. 3, with the sub-technical attributes for Performance and technical issues for Time Measurement listed as below.
- Technical Attributes:
  - Performance
  - Concurrency
  - Exception Handling
Memory

C# Specific Issues

Sub-technical attributes for Performance:

Time Measurement

Performance Bottlenecks

Technical Issues for Time Measurement:

CPU time

Wall time

Time spent with children

Time spent in method

Thread time

Language Specific Issues attribute is a generic term used to group miscellaneous issues relevant to each particular language. For example, C# specific issues attribute includes sub-technical attributes of Exception Handling and IDisposable Interface which are specific only to C# language. The technical issues relevant to each of these can be seen in Table I.

IV. UNITS

DATM is currently being applied in various Siemens projects. One such implementation case study is discussed in this section. A project from the Industry sector has been chosen. The code size is 100 KLOC and written in C#. Some dynamic analysis tools were selected to run on this project (Yourkit .NET Profiler, SpeedTrace, AQTime Pro).

Table II presents the mapping of technical issues to the selected quality attributes in the DAISQ quality model.

Table II: Rating of Technical Issues in DATM with Mapping to Quality Model

<table>
<thead>
<tr>
<th>Technical Attribute</th>
<th>Technical Sub-attribute</th>
<th>Issues</th>
<th>Reliability</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception Handling</td>
<td>Exceptions</td>
<td>Reserved exceptions on thrown Critical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>Object Allocation</td>
<td>Objects of most leaked memory Possibly critical</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Large object heap usage Possibly critical</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Null fields Possibly critical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Garbage Collection</td>
<td>Garbage Collection leak Less critical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance Bottlenecks</td>
<td>Execution hot spots</td>
<td>Possibly critical</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Frequency of function calls Less critical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time spent in method Less critical</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3. Technical model for C#.

Fig. 4. Technical model for Java.

Fig. 5. Technical model for C++.
The Rating for each of the technical issues is also provided. The Rating scale chosen is:

- **Not Critical**: No immediate action required.
- **Less Critical**: Immediate action required, but the issues should be addressed opportunistically.
- **Possibly Critical**: Difficult to confirm criticality; either more analysis is needed or context information is required to confirm its criticality.
- **Critical**: Short term action required.
- **Very Critical**: Immediate action required

The Effort and Importance associated with each of the issues, with respect to the quality model, is also obtained and is as shown in Table III. The Effort and Importance scale is: Low (for low effort and less importance), Medium and High (for high effort and high importance).

**TABLE III: RATING, EFFORT, IMPORTANCE OF QUALITY ATTRIBUTES**

<table>
<thead>
<tr>
<th>Quality Attribute</th>
<th>Quality Sub-attribute</th>
<th>Rating</th>
<th>Effort</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Runtime Stability</td>
<td>Critical</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Correctness</td>
<td>Possibly Critical</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Time Behavior</td>
<td>Critical</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Resource Utilization</td>
<td>Possibly Critical</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

Based on the three factors of – Rating, Effort and Importance, a decision can be taken on the project release; whether to go ahead with the release or to delay the release till all the issues are fixed. In case of the project under discussion, it has few issues with Rating – Critical, Importance - High, and Effort – High; hence a release decision will be taken only after the fix to these issues is made.

V. CONCLUSION AND FUTURE WORK

DATM is useful in organizing large number of technical findings. This helps in reducing the time and effort required to analyze the large reports generated by the tools.

The model also helps identify technical areas for improvement within a project. It is possible to modify the proposed technical model to any new dynamic analysis tool. Hence, it can easily be used in code assessments involving dynamic analysis tools.

Such classification can further be used to prioritize issues that are to be fixed for the project. It also helps the programmer to select categories in the technical model that have higher impact on the code quality. The proposed model is an attempt to address technical classification in DAISQ. The technical issue classification is based on the dynamic analysis tools evaluated. Fig. – Fig. 5 represent the classification based on the findings from a subset of tools. More tools can be classified and integrated into the model.

As of now, we have tried implementing DATM on a limited number of Siemens projects. We plan to use it as a part of DAISQ reviews more extensively.

REFERENCES


Snigdha Nayak received her Master in Information Systems Management from Carnegie Mellon University, USA in 2004. She has worked with General Motors, India from 2005-2007. She has been working as Associate Member Technical Staff with Siemens Corporate Research and Technologies group, India since 2007. Her research interests include various topics of software engineering such as static analysis, code clones and dynamic analysis.

Radhika D. Venkatasubramanyam received her M.S. degree in Computer Science from University of Houston, USA in 2004. She has worked with General Motors, India from 2005-2007. She has been working as Associate Member Technical Staff with Siemens Corporate Research and Technologies group, India since 2011. Prior to joining Siemens, author has worked with Tata Research Development and Design Center in Pune, India from 2006-2009. Her research interests include dynamic analysis, static analysis and unit testing.