Detecting Missing Thrown Exceptions in Enterprise Systems: an Empirical Study

Cristina Marinescu
LOOSE Research Group
“Politehnica” University of Timișoara, Romania
cristina.marinescu@cs.upt.ro

ABSTRACT
Commonly enterprise systems are implemented using the object-oriented and relational paradigms, among which the communication is performed using various library methods for manipulating the persistent data. Most of the times the involved library methods throw different exceptions. An improper handling mechanism for these exceptions in the source code may bring different problems at runtime and hamper its maintenance. In this work we introduce an approach that automatically detects the methods from the source code which reveal an improper mechanism for handling exceptions involving database operations. The detected methods should be refactored in order to increase the reliability of the application, as well as its maintenance.

Categories and Subject Descriptors
D.2.2 [Software Engineering]: Design Tools and Techniques

General Terms
software quality assessment, static analysis

1. MOTIVATION
According to [1], a system without a proper exception handling is likely to crash continuously, which renders it useless for practical purposes. We introduce an approach which provides us with those methods from the data source layer of enterprise systems which make use of an idiomatic style for dealing with exceptions and, consequently, are good candidates for refactoring.

2. APPROACH

In order to detect which are the methods from the data source layer which use an idiomatic style for handling exceptions (i.e., where the missing thrown exceptions flaw is encountered) we need to: (1) find out the methods responsible for retrieving/storing the persistent data (i.e., the methods from the data source layer). In order to find them we use a simple approach – a method is considered to belong to the data source layer if it invokes one or more methods from the library that provides the API for manipulating the persistent data (e.g., the method invokes the executeQuery() method from the java.sql package); (2) select from the data source methods only those which do not throw exceptions. We implemented within our tool DATES [2] the introduced approach which provide us with the methods from the data source layer which reveal a missing thrown exceptions flaw.

3. RESULTS
In order to evaluate our approach we have conducted a case study based on several data sources (4 enterprise systems). The source code of the analyzed systems varies from 336Kb up to 14.3Mb. For example, our approach provides us with the method exportTable. It has two parameters (filePath, tablename), the last being the name of a table whose information is intended to be saved into the file filePath. Because the method doesn’t throw any exception and returns void its clients will not be able to determine when dealing with an empty file if the queried table stores no data or an error involving the database operation was encountered (e.g., table tablename not found).

4. CONCLUSIONS AND FUTURE WORK
In this work we introduce an approach that automatically detects methods from enterprise systems which does not properly handle the exceptions related to information storage/retrieval. In most of the cases those methods should be refactored. We intend to extend the tool support in order to be able to use it upon enterprise systems written using other technologies and to continue the evaluation of the introduced approach against other enterprise systems.

5. REFERENCES