ON THE COMPLEXITY OF EXCEPTION HANDLING

PÉTER CSONTOS, ZOLTÁN PORKOLÁB

April 18, 2001

Dept. of General Computer Science
Eötvös Loránd University, Budapest
H-1117 Budapest, Pázmány Péter Sétány I/D.
E-mail: csonti@elte.hu, gsd@elte.hu
HUNGARY

Abstract

Exception handling is today a fundamental part of the modern programming languages. It provides a powerful and flexible alternative to the traditional error handling methods. It provides a way of explicitly separating error-handling code from ordinary one, thus making the program more readable and managable.

In fact, exception handling – as it is implemented in most programming languages – is an alternative way to control the program flow. Therefore it can be measured by the same metrics and tools than the ordinary procedural programs.

In this article we examine the complexity of different error-handling methods, giving a suggestion for measuring the complexity of exception-handling. We use the complexity metrics given by Judit Nyéky-Gaizler, Ákos Fóthi and Zoltán Porkoláb (presented in PhD. Workshop ECOOP 2000, Cannes), which is based on:
(1) the complexity of the control structure of the program,
(2) the complexity of data types used,
(3) the complexity of the data handling.

Because of the non-strctural property of exception handling, throwing exceptions can behave slightly differently in distinct languages. The reason for this can be the different degree of well-definedness of program behaviour in some rare situations, or the type of mandatoriness of catching an exception throwed in the given method.

After extending our measure of complexity to the trivial cases (simple explicite or implicite throwing of exceptions and catching them, inheritance hi-

erarchies of exception classes in object-oriented languages), we discuss several more sophisticated examples:
(1) The expression evaluation behaves differently in Java and C++: In Java, the evaluation order is in most cases strictly defined, while in C++ it isn't. In the example code:

```
array[i++] = foo();
```

if *foo()* throws an exception, i is incremented anyway in Java, while in C++, the result is not defined (example taken from *Java Programmer's FAQ* (the URL is http://java.sun.com/people/linden/faq_d.html). An other interesting case when i has an out of range value which raises exeption in Java, or we use user-defined subscript operator in C++ which also can throw exception.

(2) The uncatched exceptions should be indicated in the throws clause of the method in Java, otherwise the compiler reports an error. In C++, however, there's no mandatory throws clause. A function called *unexpected()* is called in case of an uncatched exception, and it happens runtime, so C++ programs seem to be more complex from this point of view because the compiler allows more freedom for the programmer.

(3) We compare the complexity of the Java technique called *finally* to the C++ catch all – ie. *catch(...)*. These two rather different techniques are used for the same fundamental reason in resource management.

First, we discuss these issues in a more detailed manner, and afterwards, propose solutions for the problems.