Java程序异常信息分析插件的研究与设计

宋道远 贲可荣
(海军工程大学计算机工程系 武汉 430033)

摘 要 异常处理是一种有效提高软件健壮性的方法,处理不当将导致严重的软件失效。提出一种通过分析Java程序异常信息、由开发平台给出异常处理代码提示的方法,以提高开发效率,并提出一种包含异常结构的Java程序异常控制流图构造方法,用于程序分析和优化。基于Eclipse开发环境,设计了一个异常信息分析插件,用于分析Java程序异常信息,给出了代码提示,生成了异常控制流图,以帮助开发人员更快更好地书写异常处理代码。

关键词 异常处理,插件设计,异常控制流图
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.08.023

Research and Design of Java Exception Information Analysis Plugin

SONG Dao-yuan BEN Ke-rong
(Department of Computer Engineering, Naval University of Engineering, Wuhan 430033, China)

Abstract Exception handling is an efficient method to improve the software robustness, while an inappropriate handling will cause a serious software failure. This paper proposed a method for analyzing the Java exception information and giving a code suggestion to improve the efficiency of the development. We proposed a method for constructing the exception control flow graph for Java with exception construct, which will be used to code analyse and optimization. Based on the Eclipse, this paper designed a Java exception information analysis plug-in, to analyse the Java exception information, gave a code suggestion and produced the exception control flow graph, which will help developers write exception handling code faster and better.

Keywords Exception handling, Plug-in design, Exception control flow graph

1 概述

在复杂的面向对象系统中,往往超过三分之二的代码用于检测可能导致错误的条件。经验表明,错误处理代码出现问题的概率很高。异常处理不当可能会导致软件系统故障,对于高可靠性的系统,后果更加严重[1]。研究表明,程序员尤其是初学者对程序的异常处理机制理解不够,而且认为异常处理不及程序的功能设计重要[2]。设计一个辅助开发人员书写异常处理代码的工具,可以有效提高程序开发效率和软件的健壮性。

目前,在异常处理分析工具方面已经有不少研究。

(1)针对Ada语言的ADAPT工具,可以分析异常传播的路径,发现不可达的异常处理程序,发现定义但没有使用的异常,发现从主程序传播到操作环境的异常等。

(2)Xept工具适用于无法获取程序源代码的情形,用于异常检测和错误恢复。Xept工具对于无法获取程序源代码而只存取到模块接口的软件特别有效[3,4]。

(3)分析C++程序中异常处理的静态分析工具CETool[5],可以有效地分析C++程序的异常传播,它既可以获得异常处理结构的局部信息,也可以获得异常处理结构的全局信息,这对于有效地分析异常的传播以及改进异常处理结构都有很大的帮助。

(4)对于Java程序,Martin P. Robillard提出一种提高Java程序健壮性的方法和工具Jex[6,7]。Jex是一种能够分析Java程序中异常信息的静态分析工具。对于每一个Java类,Jex都能输出类中包含在方法体中的所有异常信息。Byeong-Mo Chang和Jang-Wu Jo在文献[8]中提出一种基于集合框架的静态分析工具,这个工具能够获取Java程序的异常传播路径,并且从静态分析信息中构建异常传播图。

2 插件设计

目前,国内外针对Java程序的异常分析工具已经不能满足开发人员的需要。Jex工具通过分析源代码生成包含异常信息的中间文件,开发人员只能通过分析中间文件,对异常点进行分析,不能直观地获取整个程序的异常传播路径。Byeong-Mo Chang等开发的工具在开源项目Jipe上实现,目前Java程序的开发环境通常采用Eclipse集成开发环境,很少采用Jipe。本文基于Eclipse集成开发环境,设计了一个用于分析Java程序异常信息的插件。

目前,Eclipse已经成为主流的Java开发环境。除了运行

到稿日期:2013-05-28 返修日期:2013-07-20 本文受国家自然科学基金(61272108),软件工程国家重点实验室开放基金(SKLSE2012-09-38)资助。

宋道远(1988—),男,硕士生,主要研究方向为软件质量保证技术,E-mail:hustsdy@qq.com;贲可荣(1963—),男,博士,教授,主要研究方向为软件工程、人工智能等。

内核之外，Eclipse 的所有组件都是插件。异常信息分析插件在 Workbench 和 Workspace 这两个插件上扩展。异常信息分析插件的功能包括 Java 异常信息的显示、提供异常节点的异常传播路径、生成异常传播路径图；根据已有异常信息，在恰当位置给出建设性的异常处理代码提示；生成函数间和函数内的异常控制流图等。整个插件的系统结构如图 1 所示。

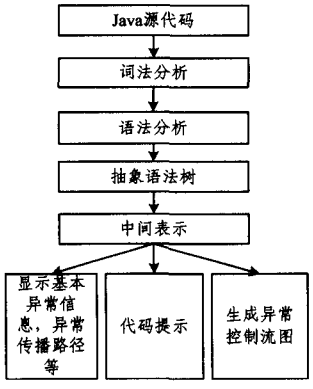

图 1　异常信息分析插件的系统结构图

开发人员给出需要进行异常分析的 Java 源代码，插件中的分析器对代码进行词法分析和语法分析，生成抽象语法树（AST）。抽象语法树中包含了异常信息，其可以用来进行程序的中间表示。通过分析中间表示，插件为开发人员提供建设性的异常处理代码，为程序分析提供函数内和函数间的异常控制流图。

2.1　代码提示模块

开发人员为了加快开发速度，往往会忽略对异常处理的考虑，并试图用一个 catch 语句去捕获程序产生的所有异常，如直接使用 catch(Exception e)语句。这样做是不值得推荐的，因为在使用 try-catch 语句时，我们的本意是，try 块内预期会出现某种异常，然后用 catch 语句去捕获该异常，由于 java. lang. Exception 是所有异常的基类，直接使用 catch(Exception e)语句意味着我们将处理所有的异常，达不到我们想处理某个特定异常的目的，这就失去了 Java 语言异常处理机制的意义。

根据文献[9]中的 Java 程序异常处理策略，在异常处理中，在条件允许的情况下应尽可能地去处理产生的异常。catch 语句应尽可能地指定具体的异常类型，当有多种异常类型时，可以使用多个 catch 语句进行捕获。

代码 1 中的异常处理方式是开发人员经常采用的方法。readFile 方法尝试读取文件，当给定的文件不存在时，会抛出异常 FileNotFoundException。开发人员为了节约时间和精力，会直接使用 catch(IOException e)来捕获 FileNotFoundException 异常，甚至使用 catch(Exception e)。但是，我们期望 catch 语句能够处理更加准确的异常 FileNotFoundException，而不仅仅是它的父类 IOException 或 Exception。

代码 1

```
void readFile()throws FileNotFoundException{}
void main(){
try{readFile ();}
catch(IOException e){}
```

为了解决此类问题，插件通过分析抽象语法树和中间表示文件，结合 Java 源代码，对 catch 语句捕获的异常类型进行分析，根据异常类型的层次关系，确定 catch 语句是否满足异常处理的精度要求，并在恰当的位置给出异常处理代码框架，提示开发人员书写更优秀的异常处理代码。代码 2 在代码 1 的基础上，增加了对 FileNotFoundException 异常的处理，提高了异常处理的精度。通过代码提示，把开发人员从繁重的异常处理中解脱出来，以便其更加专注于程序的逻辑设计，提高了开发效率。

代码 2

```
void main() {
try{readFile ();}
catch(FileNotFoundException e) {}
catch(IOException e) {}}
```

2.2　异常控制流图生成模块

当 Java 程序产生异常时，程序的执行会中止，如果没有相应 catch 语句进行捕获，异常会沿着程序调用链逆向传播，当异常到达程序的主函数时仍然没有得到处理，程序就会终止。异常的传播导致控制流发生变化，已有程序分析方法很少考虑程序异常对控制流和数据流的影响，在分析包含异常结构的程序时，为了精确描述包含异常信息的控制流，需要在普通控制流图基础上引入异常节点，构造异常控制流图[10]。

文献[10]针对包含异常结构的 C++程序，提出一种构造函数内异常控制流图的方法。Java 语言和 C++语言在异常处理机制上的主要区别是 Java 引入了 finally 关键字，并去掉了 catch(…)方法。由于 finally 的引入，在 Java 程序中，无论 try 程序块中是否发生异常，catch 程序块是否捕获异常，finally 语句都会被执行（如果 try 块内部有一条 System. exit (0)语句，则程序会终止，finally 语句不会被执行；若 try 块执行期间，突然断电，finally 也不会执行），这使得 try、catch、throw 等语句的执行顺序和 C++不同。由于文献[10]中异常控制流图构造方法不再适用于 Java，针对 Java 语言的特点和其与 C++语言的区别，本文在文献[10]的基础上，提出一种适用于 Java 程序的异常控制流图构造方法。

限于篇幅，这里只构造函数内的异常控制流图，构造算法如下：

输入：Java 源代码

输出：函数内异常控制流图

1）构造基本控制流图。

2）生成一个程序入口节点 ENTRY，指向程序的第一条语句，生成一个程序出口节点 EXIT，生成一个异常退出点 except_exit 和正常退出点 norm_exit，其中 except_exit 和 norm_exit 节点均指向 EXIT 节点。

3）包含 try，catch，throw，finally 关键字的语句均生成一个节点。

4）对于可能抛出异常（已检查异常，不处理未检查异常）的语句，首先生成一个与该异常类型关联的节点，抛出异常的语句后分出两条路径，T 和 F。一条路径 T：将该语句指向这个异常节点；另一条路径 F：连接该语句到下一条语句。若该语句在 try 块中，连接异常节点到对应的最近一个 catch 节点，若该语句不在 try 块中，直接连接异常节点至异常退出点 except_exit；若该语句后存在 finally 节点，应将该节点首先连接到 finally 节点，然后将 finally 块的最后一条语句连接至 except_exit 节点。

5）对于 try 节点，直接连接其与下一个语句，对于 try 块的最后一条语句，若 try 块后不存在 finally 节点，连接该语句至 try-catch 块后的第一条语句，若 try 块后存在 finally 节点，连接该语句到 finally 节点，然后将 finally 块的最后一条语句连接至 norm_exit 节点。

6）对于 catch 节点，该节点后分出两条路径，一条路径 T：连接该节点和下一条语句；另一条路径 F：连接该节点和 try 语句对应的下一个 catch 节点，或者上一层的 catch 节点（若有嵌套 try-catch 语句），若这些节点都没有，连接该节点到 except_exit 节点；若该 catch 节点后存在 finally 节点，应将该节点首先连接到 finally 节点，然后将 fi-

nally 块的最后一条语句连接至 except_exit 节点。

7)对于 throw 节点,生成一个与抛出的异常关联的节点,若该语句在 try 块中,连接异常节点到对应的最近一个 catch 节点,若该语句不在 try 块中,连接异常节点至异常退出点 except_exit;若 throw 节点后存在 finally 节点,应将该节点首先连接到 finally 节点,然后将 finally 块的最后一条语句连接至 except_exit 节点。

8)对于 finally 节点,直接连接其与下一条语句。

相比于文献[10],本算法引入了 finally 节点,并对原有的 try、catch、throw 等节点进行了改进,以符合 Java 语言的异常传播特点。下面是一个函数内异常控制流图的构造实例。

代码 3

```
public static void readFile(String fileName) throws IOException{
1. File file=new File(fileName);
2. Reader reader=null;
3. try{
4. reader=new InputStreamReader(new InputStream(file));}
5. catch(e) {
6. throw new ReadException ();}
7. finally{
8. reader. close;}}
```

按照上述构造方法构造的控制流图如图 2 所示。

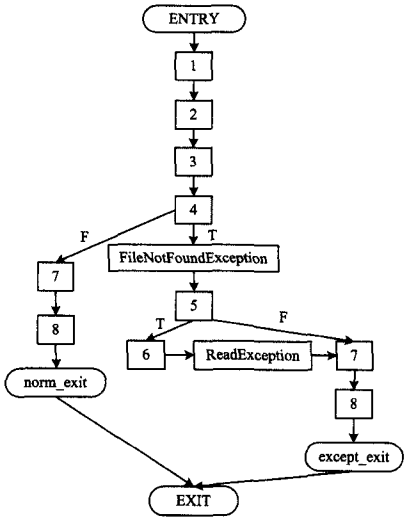

图 2 异常控制流图

插件在程序抽象语法树和中间表示的基础上,分析程序的异常信息,并根据上述构造方法,生成异常控制流图,在此基础上,开发人员进行控制流和数据流分析、程序优化等,进而提高软件健壮性。

结束语 本文提出通过分析异常类型和层次关系,给出代码提示的方法,以解决开发人员注重程序逻辑、忽视异常处理的问题。针对 Java 语言的异常处理特点,本文在已有的异常控制流图基础上引入了 finally 节点来重新定义 try、catch、throw 等异常节点的构造方法,提出了 Java 异常控制流图的构造方法,为程序分析奠定了基础。Eclipse 平台下的异常信息分析插件可以显示 Java 程序异常信息、代码提示,生成异常控制流图,帮助开发人员合理地使用 Java 异常机制和程序分析,提高程序开发效率和软件健壮性。

参 考 文 献

[1] Sawadpong P, Allen E B, Williams B J. Exception Handling Defects: An Empirical Study[C]// 14th International Symposium on High-Assurance Systems Engineering. Omaha, USA, October 2012: 90-97

[2] Rashkovits R, Lavy I. Students' Misconceptions of Java[C]// 7th Mediterranean Conference on Information System. Guimaraes, Portugal, September 2012: 1-21

[3] 姜淑娟,徐宝文. 异常处理——一种提高软件健壮性的方法[J]. 计算机科学,2003,30(9):169-172

[4] 姜淑娟. 异常传播分析技术及其应用研究[D]. 南京:东南大学,2006

[5] 姜淑娟,徐宝文,姜元鹏. 一个异常传播分析工具的设计与实现[J]. 计算机科学,2008,35(7):277-279

[6] Robillard M P. Analyzing exception flow in Java programs [D]. The University of British Columbia, 1999

[7] Robillard M P, Murphy G C. Designing Robust Java Programs with Exception[C]// 8th ACM SIGSOFT international symposium on Foundation of software engineering. 2000: 2-10

[8] Chang B-M, Jo Jang-wu. Visualization of Exception Propagation for Java using Static Analysis[C]// 2nd IEEE International Workshop on Source Code Analysis and Manipulation. Canada, 2002: 173-182

[9] 陈红跃,张宏军,陈刚. Java 异常处理策略研究[J]. 计算机技术与发展,2012,22(7):9-12

[10] Zhang Yan-mei, Jiang Shu-juan, Zhao Xue-feng. Analysis of Object-oriented Programs with Exception-Handling Constructs[J]. International Journal of Advancements in Computing Technology, 2012, 4(1): 505-515

(上接第 96 页)

后的研究中,将会增强加密效果,保护个人手机中图像的隐私。

参 考 文 献

[1] 张云鹏,左飞,翟正军. 基于混沌的数字图像加密综述[J]. 计算机工程与设计,2011,32(2):463-466

[2] Pareek N K, Patidar V, Sud K K. Image encryption using chaotic logistic map [J]. Image and Vision Computing, 2006, 24(9): 926-934

[3] Yoon J W, Kim H. An image encryption scheme with a pseudo-random permutation based on chaotic maps [J]. Commun Nonlinear Sci Numer Simulat, 2010, 15(12): 3998-4006

[4] Wang Yong, Wong K-W, Liao Xiao-feng, et al. A new chaos-based fast image encryption algorithm [J]. Applied Soft Computing, 2010, 11(1): 514-522

[5] Singh N, Sinha A. Chaos based multiple image encryption using multiple canonical transforms [J]. Optics & Laser Technology, 2010, 42(5): 724-731

[6] Lin Qiu-hua, Yin Fu-liang, Mei Tie-min, et al. A blind source separation-based method for multiple images encryption [J]. Image and Vision Computing, 2008, 26(6): 788-798

[7] Liao Xiao-feng, Lai Shi-yue, Zhou Qing. A novel image encryption algorithm based on self-adaptive wave transmission [J]. Signal Processing, 2010, 90(9): 2714-2722

[8] Chen W, Quan C, Tay C J. Optical color image encryption based on Arnold transform and interference method [J]. Optics Communications, 2009, 282(18): 3680-3685

[9] Gao Tie-gang, Chen Zeng-qiang. A new image encryption algorithm based on hyper-chaos [J]. Physics Letters A, 2008, 372(4): 394-400