一种基于调用链分析的特征定位方法

付 焜 钱文亿 彭 鑫 赵文耘
(复旦大学软件学院 上海 201203) (上海市数据科学重点实验室(复旦大学) 上海 201203)

摘 要 为了完成各种软件维护任务,如纠正错误、改进原有功能、添加新功能,开发人员经常需要确定需求特征与代码的对应关系。这种确定源代码中与给定需求特征相关的程序元素的过程称为特征定位。现有的特征定位方法主要根据用户提出的需求,在源代码中搜索相关的代码元素推荐给用户。然而这些零散的元素之间不具备任何关联,用户仍然需要人工地挖掘元素间的关系,来了解代码元素是如何相互配合、实现特定功能的。而通过与数据传递相关的方法调用链可以改进特征定位的实践方法。该方法能分析源代码,获取到所有的与数据传递相关的方法调用链,然后将根据用户提供的相关需求的关键字找到相关的调用链,推荐给用户。这种调用链不再是零散的代码元素,它能够反映出特定功能实现的流程,也能够更好地帮助用户理解程序。基于该方法实现的Eclipse插件工具已经在JEdit项目上进行了测试。结果显示该工具给出的推荐结果平均查准率可达55%。

关键词 特征定位,程序理解,调用分析,调用链推荐,Eclipse插件
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.008

Feature Location Method Based on Call Chain Analysis

FU Kun QIAN Wen-yi PENG Xin ZHAO Wen-yun
(School of Software, Fudan University, Shanghai 201203, China)
(Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)

Abstract In order to accomplish a variety of software maintenance tasks, such as fixing bugs, changing existent functionalities, or adding new features, developers often need to look for related codes of a feature in advance. Such a process to identify relevant program elements according to a given feature is called as feature location. Existing feature localization methods, mainly based on user demand, search related code elements in the source code, and recommend them to the user. However, these scattered elements do not have any connection with each other, and user may still need to manually find out the relationship between the elements, to understand how the code elements combine together to achieve a specific function. However, a new approach based on method-call-chains associated with data transferring can improve the feature location practice. This method can analyze the source code, extract all of the method-call-chains associated with the data transferring, and find relevant ones on the user's demand. This method-call-chain is not a simple code segment. It can reveal the implementation of specific functions, and help user to understand the program better. An Eclipse plug-in of this approach was evaluated on JEdit. And the precision of the results of the tool is average 55%.

Keywords Feature location, Program understand, Method calls analysis, Recommended, Eclipse plugins

1 引言

为了执行错误修复、功能增强等软件维护任务,开发者需要了解相关的功能在代码中是如何分布以及如何实现的。在软件工程中,确定与特定功能相关的实现代码的过程被称为特征定位[1,2]。近年来,为了帮助开发人员完成特征定位工作,越来越多的技术被提出,例如静态分析技术[3,4]、动态分析技术[5,6],以及多种技术相结合的方法[7]。

然而,现有的技术大多数都只能返回相关代码元素。开发者仍然需要耗费一定的精力来整合所得的结果。而本文提出的方法将推荐给开发者一些与特定功能相关的方法调用链。这种调用链不仅是一些与特定功能相关的代码元素,同时还是一种结构化的数据信息,能够反映其中包含的方法元素间的调用关系和相应功能的实现过程。

我们提出的方法以程序源代码和开发者的具体需求作为输入,提取出所有与数据传递相关的方法调用链,根据开发者的具体需求,从提取出来的调用链中找出与具体需求相关的调用链推荐给开发者,从而实现特征定位的功能。

到稿日期:2013-09-18 返修日期:2013-11-16 本文受国家"863"高技术研究发展计划项目基金(2012AA011202),教育部高校博士点基金(20100071110031)资助。

付 焜(1989—),男,硕士生,主要研究领域为软件维护,E-mail:13212010004@fudan.edu.cn;钱文亿 男,博士生,主要研究领域为软件维护;彭 鑫(1979—),男,博士,副教授,CCF高级会员,主要研究领域为软件维护、自适应软件系统;赵文耘(1964—),男,硕士,教授,CCF高级会员,主要研究领域为软件复用、电子商务。

最后，我们以 Eclipse 插件的形式实现了该方法，并在 JEdit 项目上应用了这个方法，模拟了 4 个用户输入，获取到了工具的推荐结果，并对结果进行了分析。

2 相关概念

我们先来分析一个软件维护的例子。

现有一个用 java 开发的项目 JEdit，它是一个开源的编辑器软件。现在开发人员要更改一下软件中变换选中文字字体的功能。他首先要做的就是弄清变换字体的功能是如何实现的。开始开发者使用 IDE 中的一些搜索工具在源代码中寻找与这个功能相关的代码元素，来找到相关方法：

org. gjt. sp. jedit. Buffer. editSyntaxStyle(JEditTextArea)

之后开发者以该方法为中心，通过依赖、位置关系找出如图 1 中所示的两组方法调用。其中，前一组方法调用从用户界面的功能组件中得到了用户所指定的字体信息，而后一组则更新相关数据，并刷新显示视图。

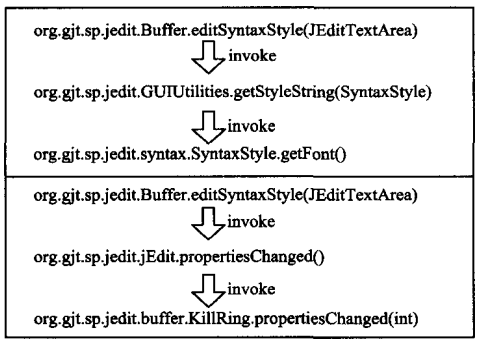

图 1 维护人员通过方法调用找出的两组方法

开发者所找出的两组方法调用，实际上就是我们所关注的“与数据传递相关的方法调用链”。

在软件系统中，一个业务功能在运作的过程中往往会有一些非常重要的数据，亦即我们在上面提到的调用链所传递的数据。而这些数据在软件调用链上传递的过程，也就是软件系统逐渐实现功能的过程。因此，可以说这种方法调用链在一定程度上能够反映与之相关的功能是如何实现的。

3 方法介绍

3.1 方法概述

我们所提出的方法在源代码中提取出方法调用链，并在开发者给出功能相关的关键字后将相关的调用链推荐给开发者，辅助特征定位。整个方法实现的过程如图 2 所示。

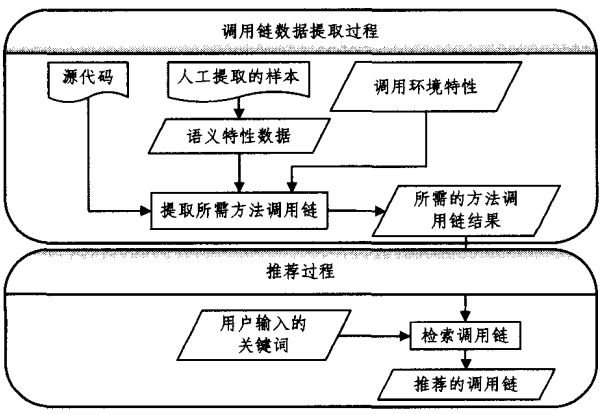

图 2 方法流程图

该方法主要分为两个过程：

1)调用链的抽取。我们将对源代码进行分析，从源代码中提取出方法调用链。

2)推荐过程。这个过程需要源代码中所有的方法调用链数据和开发者给出的与需求相关的关键字。我们将根据关键字，找出相应的方法调用链，推荐给开发者。

3.2 调用链数据的提取过程

调用链的提取过程主要有以下步骤：获取语义特性、获取调用环境特性和提取所需方法调用链。

3.2.1 语义特性的获取

我们所定义的语义特性是一组具有实际语义的词汇，它们能够反映出相关方法在语义上的特征。

首先需要一个软件系统中与数据传递相关的方法样本集。我们请一位对 JEdit 比较了解的专家为我们提供一些源代码中与数据传递相关的方法。然后提取样本集中方法在管理数据或是在传递数据的过程中出现的词汇。在此，我们使用 Eclipse 插件平台上的 JDT 开发包分析源代码，提取带语义的数据如方法名、参数名、注释等。将方法提取出的词汇集看作文本，然后计算每个词汇的 TF-IDF 值，并将同一词汇的值累加，最后取得前 16 个词汇。这个词汇集合也就是我们需要的语义特征数据。

3.2.2 调用环境特性的获取

本方法中所指定的方法调用的环境特性是指：与数据传递相关的方法调用中，发生调用和被调用的两个方法处于不同的模块中。系统可以通过某次方法调用将一些数据信息传递到另一个模块中，进而完成数据的进一步处理。同时，由于以上的特性，需要对要分析的软件系统进行模块划分。我们使用了人工划分的方式，将原代码中的元素以类为单位划分为视图(View)、控制(Control)、数据存储(Storage)这 3 个模块。

3.2.3 提取所需方法调用链

接下来我们先根据语义特性的具体数据，从源代码中提取出一个方法的初步候选集。再根据调用环境特性，对初步候选集进行扩充。最后根据方法的依赖关系，将这些方法连接成调用链，作为最后的结果。具体过程如下：

1)按照下式计算其方法的语义特性值：

$$C_s(m)=|\{x|x\in S_{check}\wedge x\in S(m)\}|$$

其中，$C_s(m)$表示方法 m 的语义特性值，S_{check} 代表 3.2.1 节中提取出的词汇集合，$S(m)$ 代表在分析过程中从方法 m 中抽取出来的词汇集合。

计算完之后设置一个阈值 α，然后把语义特性值大于该阈值的方法组成一个集合，即初步候选集。

2)根据调用环境特性扩充初步候选集。首先我们定义方法调用的环境特性值：

$$C_e(m_1,m_2)=\begin{cases}1, & T(CL(m_1))\neq T(CL(m_2))\\ 0, & T(CL(m_1))=T(CL(m_2))\end{cases}$$

$C_e(m_1,m_2)$代表方法调用的环境特性值，其中 m_1、m_2 为两个方法，且 m_1 调用 m_2。$CL(m)$代表方法 m 所在的类，而 $T(c)$代表类 c 所在的模块。

在扩展初步候选集时，首先考虑在初步候选集中的方法 m，计算出每个调用的环境特性值，并将该数值累加到调用和被调用的方法上，最终获得的即为每个方法的环境特性值。

按照这样的方式，对于一个包含在初步的候选集中的方法 m，若它所调用过的方法的集合为 $Callee(m)$，而调用过 m 且被包含在初步的候选集中的方法集为 $Caller(m)$，那么该方法在最终获得的环境特征值为：

$$C_e(m)=\sum_{m_i\in Call(m)}C_e(m,m_i)+\sum_{m_j\in CallFromSet(m)}C_e(m,m_j)$$

定义 M' 为被 m 调用的，且不在初步候选集中的方法的集合中，对于 M' 中的每个方法 m'，其环境特性值为：

$$C_e(m')=\sum_{m_i\in CallFromSet(m')}C_e(m',m_i)$$

计算所有方法的环境特性值，并将数值大于零的方法添加到初步候选集中得到结果集。再将结果集中每个方法的语义特征值和环境特征值求和，作为方法的特性值。

3)现在已经获取到方法的集合，还需将方法连接成调用链。具体方法如图 3 所示。将候选集中的方法放入一个列表中，见图 3(a)。然后遍历调用关系，当发现方法 A 调用方法 C，找到 C 节点，将其列为 A 节点的子节点，并将 C 节点从列表中删除，见图 3(b)。这样遍历结束后，如图 3(c)所示，得到方法调用树，再次遍历树，获取方法调用链。

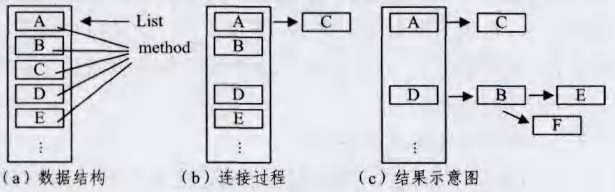

图 3　调用链连接算法示意图

然后将调用链中所有方法的特性值求和得到一个调用链的特性值。该数值最终能够反映该调用链的推荐程度。最后持久化调用链数据，以便后续过程使用。

3.3　推荐过程

得到方法调用链数据后，用户输入关键字，我们遍历所有的方法调用链，对每个调用链获取到其中包含的方法的全名，然后在这些方法名中搜索关键字，并记录关键字出现的次数。

找出所有方法名中出现过关键字的方法调用链，并按照关键字出现的次数从高到低排序。对关键字出现次数相同的调用链，再用特性值对其进行排序。然后返回前 β 个结果(具体数值将在试验中设定)，作为推荐给开发者的方法调用链。至此便完成了推荐功能。

4　工具实现

我们实现了一个 Eclipse 插件来完成 3.3 节中的推荐工作。将项目调用链数据文件放到项目的根目录下。然后在 Eclipse 项目的右键菜单中选 Reference Link Analysis 目录下的 Search，便会出现 Search Dependence Links 视图，点击视图上的添加按钮，输入特定的关键词，就能显示最终的结果，如图 4 所示。

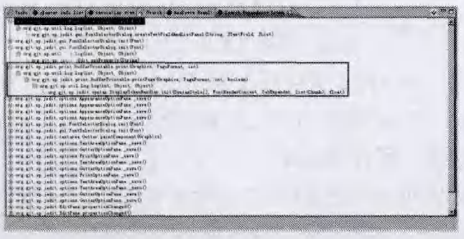

图 4　工具界面图

每条返回结果以一个折叠树的形式返回，将树逐一打开便可以看到一条完整的调用链所包含的各个方法。

5　案例分析

我们用工具对一个开源的 java 项目 jEdit 进行分析，并测试了推荐工具的最终使用效果。

5.1　语义特性数据提取

根据 3.2.1 节，首先请 jEdit 的专家挑选了 141 个与数据传递相关的方法。然后提取这些方法中的词汇，并进行 TF-IDF 值的计算。处理后共获得 214 个词汇，并按照 TF-IDF 值进行排序。人工检查后，最终获取到 18 个具有较高的 TF-IDF 值且与数据处理相关的词汇，如表 1 所列。最后依据这些数据从源代码中检索与数据传递相关的方法。实际检索时，我们将阈值 α 设置为 10。

表 1　语义信息统计表

Tags	TF-IDF	Tags	TF-IDF
property	7.57	Set	1.94
select	3.86	method	1.89
add	2.68	selection	1.73
text	2.53	append	1.66
name	2.42	string	1.24
log	2.23	stream	1.18
value	2.06	exception	0.78
get	2.04	read	0.37
buff	2.02	write	0.21

5.2　工具使用情况

在实际使用工具的过程中，我们将推荐结果的数量设置为 25，并模拟了 4 个需求关键字的输入，然后请一位了解 JEdit 的领域专家查看产生的结果是否满足使用者的需求。

表 2 为统计的结果。关键字是用户输入的与其想要查找的功能相关的关键字。下面的比例为返回的结果中，经过专家判定，与相应特征相关的调用链占总的推荐结果的百分比。其平均值达到了 55%。

表 2　准确率统计表

关键字	相关调用链的准确率
Font	80%
high light	76%
selection length	36%
filter Shortcuts	28%
平均值	55%

这样的结果表明，在推荐数量设置为 25 的情况下，平均每个任务能够为用户推荐 13 条以上的调用链，而这些调用链能够反映相关特征的实现过程，而不仅仅是一些还需用户去整理的代码元素。所以我们的方法的确能够帮助用户更好地执行特征定位任务。

结束语　本文针对如何帮助开发人员尽快地定位与特定功能相关的代码元素的问题，提出了用“与数据传递相关的方法调用链”来改进特征定位实践，给出了一个方法来获取到源代码与数据传递相关的方法调用链，并根据开发者的需求关键字推荐出相关的方法调用链。最后实现了一个相关工具并对该方法进行了测试，得到了较为满意的结果，改进了开发人员特征定位的实践，使开发人员获得了更好的特征定位体验。

参 考 文 献

[1]　Biggerstaff T J, Mitbander B G, Webster D. The concept assign-

ment problem in program understanding[C]// Proceedings of the 15th International Conference on Software Engineering. IEEE Computer Society Press, 1993: 482-498

[2] Dit B, Revelle M, Gethers M, et al. Feature Location in Source Code: A Taxonomy and Survey[J]. Journal of Software: Evolution and Process, 2013, 25(1): 53-95

[3] Trifu M. Using Dataflow Information for Concern Identification in Object-Oriented Software Systems[C]// Proceedings of European Conference on Software Maintenance and Reengineering (CSMR' 08). 2008: 193-202

[4] Robillard M P. To pology Analysis of Software Dependencies [J]. ACM Transactions on Software Engineering and Methodology, 2008, 17(4): 1-36

[5] Wong W E, Gokhale S S, Horgan J R, et al. Locating program features using execution slices[C]// Proceedings of IEEE Symposium on Application-Specific Systems and Software Engineering and Technology (ASSET'99). March 1999: 194 -203

[6] Eisenberg A D, De Volder K. Dynamic Feature Traces: Finding Features in Unfamiliar Code[C]// Proceedings of 21st IEEE International Conference on Software Maintenance (ICSM'05). Buda pest, Hungary, September 2005: 337-346

[7] Zhao W, Zhang L, Liu Y, et al. SNIAFL: Towards a Static Non - interactive Approach to Feature Location[J]. ACM Transactions on Software Engineering and Methodologies (TOSEM), 2006, 15(2): 195-226

(上接第 6 页)

[23] Aamodt A. Knowledge-Intensive Case-Based Reasoning and Sustained Learning[C]// ECAI. 1990, 90

[24] Aamodt A. Knowledge-intensive case-based reasoning in creek [M]// Advances in Case-Based Reasoning. Springer Berlin Heidelberg, 2004: 1-15

[25] Recio-Garía J A, Díaz-Agudo B. Ontology based CBR with jCOLIBRI[M]// Applications and Innovations in Intelligent Systems XIV. Springer London, 2007: 149-162

[26] Juarez J M, Salort J, Palma J, et al. Case Representation Ontology for Case Retrieval Systems in Medical Domains[C]// Proceeding of the IASTED International Conference on Artificial Intelligence and Applications (AIA2007). 2007: 168-173

[27] Plaza E, McGinty L. Distributed case-based reasoning[J]. The Knowledge engineering review, 2005, 20(3): 261-265

[28] d'Aquin M, Lieber J, Napoli A. Decentralized case-based reasoning for the Semantic Web[M]// The Semantic Web-ISWC 2005. Springer Berlin Heidelberg, 2005: 142-155

[29] d'Aquin M, Lieber J, Napoli A. Decentralized case-based reasoning and Semantic Web technologies applied to decision support in oncology[J]. The Knowledge Engineering Review, 2013, 28 (4): 425-449

[30] Bergmann R, Kolodner J, Plaza E. Representation in case-based reasoning[J]. The Knowledge Engineering Review, 2005, 20 (3): 209-213

[31] de Mantaras R L. Case-based Reasoning[M]. Springer-Verlag, Heidelberg, 2001

[32] Cao M J, Shang F H, et al. Research on Scalable Case Representation and its Retrieval Based on Description Logic[C]// Second International Symposium on Knowledge Acquisition and Modeling. 2009: 316-318

[33] Lieber J, Napoli A. Using Classification in Case-Based Planning' [C]// ECAI. PITMAN, 1996: 132-136

[34] Hunter J, Drennan J, Little S. Realizing the Hydrogen Economy Through Semantic Web Technologies[J]. Intelligent Systems, IEEE, 2004, 19(1): 40-47

[35] Tversky A. Features of similarity [J]. Psychological Review, 1977, 84: 327-352

[36] Borgida A, Walsh T, Hirsh H. Towards Measuring Similarity in Description Logics[C]// Proceedings of the 2005 International Workshop on Description Logics. Edinburgh, Scotland, UK, 2005

[37] d'Amato C, Fanizzi N, Esposito F. A Semantic Similarity Measure for Expressive Description Logics[C]// Proceedings of Convegno Italiano di Logica Computazionale. Rome, Italy, 2005

[38] Rada R, Mili H, Bicknell E, et al. Development and Application of a Metric on Semantic Nets[J]. IEEE Transactions on Systems, Man and Cybernetics, 1989, 19(1): 17-30

[39] d'Amato C, Fanizzi N, Esposito F. A Semantic Similarity Measure for ALN Description Logics[C]// Proceedings of Convegno Italiano di Logica Computazionale. Rome, Italy, 2005

[40] Otanon S, Plaza E. On similarity Based on a Refinement Lattice [C]// Lecture Notes in Computer Science, 5650. Springer, 2009: 240-255

[41] Sanchez-Ruiz-Granados A A, Ontanon S, Gonzalez-Calero P A, et al. Measuring Similarity in Description Logics Using Refinement Operators[C]// Lecture Notes in Artificial Intelligence, 6880. Springer, 2011: 289-303

[42] d'Amato C, Fanizzi N, Esposito F. Analogical reasoning in description logics[M]// Uncertainty reasoning for the Semantic Web I. Springer Berlin Heidelberg, 2008: 330-347

[43] d'Amato C, Staab S, Fanizzi N. On the Influence of Description Logics Ontologies on Conceptual Similarity[C]// Lecture Notes in Computer Science, 5268. Springer, 2008: 48-63

[44] Baader F, Sertkaya B, Turhan A Y. Computing the Least Common Subsumer wrt a Background Terminology[J]. Journal of Applied Logic, 2007, 5(3): 392-420

[45] Armengol E, Plaza E. Using Symbolic Descriptions to Explain Similarity on CBR[C]// Proceedings for the 8th International Conference of the ACIA. Alguer, Italy, 2005: 239-246

[46] Armengol E, Plaza E. Symbolic Explanation of Similarities in Case-based Reasoning[J]. Computers and Artificial Intelligence, 2006, 25(2/3): 153-171

[47] Plaza E, Armengol E, Ontañón S. The Explanatory Power of Symbolic Similarity in Case-based Reasoning[J]. Artificial Intelligence Review, 2005, 24(2): 145-161

[48] Napoli A, Lieber J, Courien R. Classification-based Problem Solving in CBR[M]// Smith I. & Faltings B., eds. Advances in CBR, Springer-Verlag, 1996

[49] Watson I. Case-based Reasoning is a Methodology, not a Technology[J]. Knowledge-based Systems, 1999, 12(5/6): 303-308

[50] Torta G, Torasso P. The role of OBDDs in controlling the complexity of model based diagnosis[C]// Fifteenth international workshop on principles of diagnosis (DX'04). 2004