基于本体的并发错误测试工具推荐方法研究

郑 炜 黄月明 吴潇雪 冯 晨 蔺 军
(西北工业大学软件与微电子学院 西安 710129)

摘 要 随着并发系统测试关注度的日益提高,越来越多的并发系统测试工具不断出现。对于测试人员来说,能否正确选取并发系统测试工具也就成为了并发测试效率的决定因素之一。鉴于并发错误检测软件并不像传统测试软件那样被人们所熟知,提出一种基于本体设计的并发错误测试工具推荐方法。该方法分别根据并发错误类型、程序本身特征和用户具体需求推荐适合的并发错误测试工具,从而提高测试的效率。

关键词 并发错误,本体,测试工具推荐

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.11.031

Research on Recommendation of Concurrency Bug Testing Tools Based on Ontology

ZHENG Wei HUANG Yue-ming WU Xiao-xue FENG Chen LIN Jun
(School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710129, China)

Abstract With the concern of concurrent system testing increasing recently, new and different test tools are constantly coming into the market. For software test engineer, the ability of selecting test tools correctly has become the determining factors of testing efficiency. Because concurrent system test tools are not as well-known as those traditional test tools, we designed a recommendation approach based on ontology to promote concurrent system test tools for users. The recommendation approach recommends concurrent system test tools through three modules to improve test efficiency: the types of concurrency bugs, the features of the program to be tested and the requirement of users.

Keywords Concurrency bug, Ontology, Testing tool recommendation

1 引言

1.1 研究背景

随着云计算、大数据以及移动互联网等行业的快速发展,对系统软件的性能要求和软件复杂度都在急剧上升,而并发处理就是提高软件性能和处理效率的有效途径。然而并发系统在带来高性能的同时,也带来了巨大的质量风险,因此需投入大量的人力和时间对并发程序进行测试,但显然现有的传统测试技术在可靠性和性能上都无法满足并发系统测试的要求,如何提高并发测试的质量已成为学术界和工业界共同关注的焦点。并发测试软件并不像传统经典测试工具那样为人们所了解,因此测试人员在面对并发错误时需要相应的推荐方法来帮助他们选择合适的并发错误测试软件。

同时,本体已经成为知识工程中的一个重要工具,在知识获取、表示、分析和应用方面都具有重要的意义:1)本体研究促进知识工程中对本质知识的获取;2)本体研究可以显式地表示出领域知识和领域假设;3)本体研究使知识共享和知识重用成为可能[1]。本文提出了一种基于本体的推荐方法,该方法将本体引入到推荐方法中,采用本体的方式对用户和测试工具等资源信息进行表示。

1.2 研究目标

总体目标是设计基于本体的并发错误测试工具推荐方法,以本体为载体,充分利用不同测试工具针对的不同并发错误和并发错误所蕴含的语义信息进行并发测试工具的推荐,从而为用户推荐高效且正确的并发测试工具,提高并发测试可靠性,降低测试成本。具体研究工作如下:

1)对并发系统中的并发错误进行探究,对已知的并发错误进行分类并提取其典型特征,为并发错误及相关系统测试工具本体的构建打下基础;

2)对现有的测试工具进行研究,分析各自的特色与侧重点;

3)详细研究本体论,分3个模块建立并发错误及相关系统测试工具本体;

4)深入研究本体推理机及其实现技术,用当前主流的

到稿日期:2016-10-07 返修日期:2016-12-26 本文受国家自然科学基金(61402370)资助。

郑 炜(1975-),男,博士,副教授,主要研究方向为基于互联网+的智慧城市、云环境下的软件开发与验证、复杂系统软件的开发与验证,E-mail:wzheng@nwpu.edu.cn;黄月明(1994-),女,硕士生,主要研究方向为软件测试,E-mail:huangyueming@mail.nwpu.edu.cn;吴潇雪(1983-),女,博士生,主要研究方向为软件测试。

本体推理机实现对并发错误及相关系统测试工具本体的推理功能;

5)对构建的本体进行实验。

1.3　研究意义

软件测试一直是软件开发中的重要问题,其中最重要的问题是错误的重现,即通过不断地重现错误来反复观察系统在出错过程中的运行状态,不断丰富关于错误的知识,最终找到其根本原因。但是,近年来测试技术面临着新的挑战:一方面,随着多核平台的普及,多线程并发程序越来越丰富;另一方面,随着以云计算为代表的大规模并发系统的快速发展,在这种系统上运行的大规模并发程序也愈发普遍。

对高度并发的程序进行测试面临一些新的问题,其中最主要的问题是程序的非确定性。由于各种无法控制的非确定事件,在并发环境中难以通过反复运行一个程序来重建相同的运行状态,因而也难以重现错误,这使得以错误重现为基础的循环测试和在线交互测试失效。传统的测试技术主要通过测试器设置断点、单步执行等方法进行测试,但并发程序的不确定性使得这种方法在检查并发程序故障时的有效性大大降低。并发程序可能在某次执行中暴露出一个故障,但在之后的测试中不暴露该故障。与串行错误依赖于特定的输入不同,并发错误的显现不仅依赖输入,而且依赖线程交织顺序等不确定因素。并发错误的不确定性使得它们具有不可复现的特性,即错误可能在某次执行中暴露,但在之后的测试中却不暴露,因此又被称为 heisenbug[2]。

传统测试软件对 heisenbug 的测试能力较弱,而针对 heisenbug 的并发错误测试软件又不如传统软件那样为人们所了解,因此本文对并发错误测试工具进行推荐。

2　本体论及模型构建

2.1　本体概念与建模

哲学领域中对本体的解释为:“本体(Ontology)是关于事物客观存在的,是对客观存在事物的系统性解释和说明,关心的是客观现实的抽象本质”[3]。随着现代化信息技术的发展,本体技术被应用到了人工智能领域,它在一定程度上表示人脑中的共享知识和概念。本体技术在信息检索系统、知识工程、自然语言处理和软件工程等领域都有研究与应用,本文使用斯坦福大学医学院提出的七步法[4]来进行本体构建。七步法主要用于构建领域本体,具体步骤如图 1 所示。

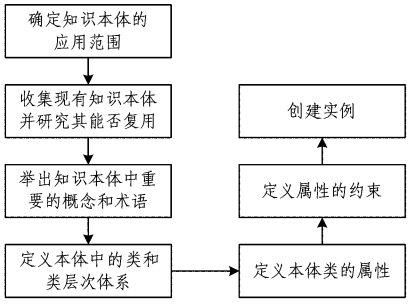


图 1　七步法过程

2.2　本体描述语言

目前已经存在的或正开发的本体描述语言有:RDF,OWL 和 DAML+OIL 等,本文主要使用前两种。

(1)RDF(Resource Description Framework),即资源描述框架。RDF 是 W3C 在 XML 的基础上推荐的一种标准,用于表示任何资源信息。RDF 提出了一个简单的模型来表示任意类型的数据。这个数据类型由节点和节点之间带有标记的连接弧组成。节点用来表示 Web 上的资源,弧用来表示这些资源的属性。因此,这个数据模型可以方便地描述对象(或者资源)以及它们之间的关系。RDF 的数据模型实质上是一种三元关系的表达,由于任何复杂的关系都可以分解为多个简单的三元关系,因此 RDF 的数据模型可以作为其他任何复杂关系模型的基础模型。RDF 模型图可以形象地表示三元组间的联系,弧和结点分别表示属性类型和语义网中的资源或属性值,如图 2 所示。

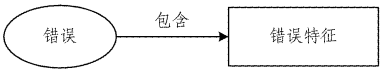


图 2　RDF 模型图实例

(2)OWL(Web Ontology Language)。相对于 RDF 提供了大量原语,OWL 能清晰地反映所描述概念之间的关系,能对相关领域信息进行有效的语义推理。因此,本文使用语义 Web 的标准本体语言 OWL 表示本体。相对于 XML,RDF 和 RDFS 语言,OWL 拥有更多的表达机制来表达形式语义。

3　并发错误及测试工具分析

3.1　并发错误的分类与特征

在所有类型的软件缺陷中,并发错误是最麻烦的类型之一。并发错误广泛存在于并发程序中,这是因为大部分的程序员都习惯于编写串行的程序,而非并发程序。为了对并发错误进行研究,需要对并发错误进行分类。本文通过对现有研究成果进行分析和总结,将并发错误分为死锁、数据竞争、原子性违背以及顺序违背 4 个大类,而每个大类下又有若干小分类。并发错误分类及其层次结构示例如图 3 所示。

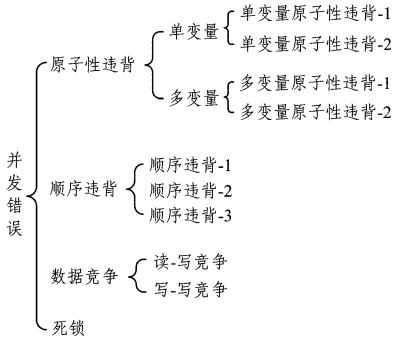


图 3　并发错误分类及其层次结构实例

此外,还可以根据已知的程序性能特点来判断可能导致的并发错误,这就需要对现有的并发错误分类进行研究,并提取其特点。通过对并发错误引发条件和状态信息等进行分析,整理得到并发错误的 21 条相关属性[5],具体描述如表 1 所列。

表 1 并发错误的相关属性

编号	属性
1	至少有一个线程处于等待状态
2	至少有一个线程处于执行状态
3	至少有一个线程处于就绪状态
4	所有线程都在等待一个被其他线程占用的锁
5	至少有一个线程处于等待状态超过一定可接受时间
6	所有线程都处于执行状态
7	没有线程可以继续执行
8	线程数量大于空闲处理器内核数量
9	有不正确或者非预期的结果出现
10	所有线程占用一个锁
11	至少有线程占用一个锁
12	对共享内存的访问来自多个线程
13	对共享内存的访问中至少有一个是“写”操作
14	对共享内存的访问目标内存位置相同
15	共享内存访问未受同步机制保护
16	共享内存访问的目标内存地址只有一个
17	共享内存访问的目标内存地址存在多个
18	对某一共享内存的访问至少有两个，一个“读”，一个“写”，“读”访问早于“写”访问
19	对同一共享内存至少有两个“写”访问，中间没有任何“读”访问发生
20	对共享内存的访问次序中至少有一个执行次序是正确的
21	语句执行具有原子性要求

对表 1 进行观察总结和分析，得到的结果如表 2 所列，其中“√”表示该并发错误类型具有该属性，空白表示该错误没有此特点。

表 2 并发错误属性和分类的关系

		数据竞争		顺序违背			原子性违背			
							单变量		多变量	
属性编号	死锁	内存不一致	写写竞争	顺序违背—1	顺序违背—2	顺序违背—3	单变量原子性违背—1	单变量原子性违背—2	多变原子性违背—1	多变原子性违背—2
1	√			√			√		√	
2		√	√		√	√	√	√	√	√
3						√		√		√
4	√									
5	√									
6		√	√	√						
7	√									
8		√	√	√	√	√	√	√	√	√
9						√		√		√
10	√									
11	√				√	√	√	√	√	√
12		√	√	√	√	√	√	√	√	√
13		√	√	√	√	√	√	√	√	√
14		√	√	√	√	√	√	√	√	√
15		√	√							
16							√	√		
17									√	√
18		√								
19			√							
20				√	√	√				
21							√	√	√	√

3.2 测试工具分析

由于并发错误难以被传统经典测试工具检测，因此必须使用特殊的并发错误测试软件，而不同测试软件的算法和设计思想不同，会导致其针对的并发错误类型也有所不同。

表 3 根据针对的不同的并发错误类型，介绍一些测试工具，“√”表示该工具针对此类型的并发错误。

表 3 并发测试工具及其针对的错误类型

	死锁	数据竞争	原子性违背	顺序违背
findbugs	√			
Jest	√	√	√	√
Intel Thread Checker	√	√		
AtomFinder			√	
Keshmesh	√		√	
Relay			√	√
ThreadAnalyzer	√			
Mtrat		√		√
Doctor watson	√	√	√	√
Coverity Prevent	√			
PMD	√	√	√	√
Valgrind		√		
Parallel Inspector	√	√		
UNICON	√			√

4 基于本体的推荐模型

本文使用美国斯坦福大学研究人员基于 Java 研制开发的开放源码软件 protégé 来构建本体。由于用户给出的信息量具有不确定性，文中的本体分为 3 个模块，根据所获取信息的不同来进行推荐。1)最佳的情况是，用户对待测程序中可能存在的并发错误类型有一定的认知，然后通过构建的本体和规则进行相应测试工具的推荐，即针对特定并发错误类型进行工具推荐。2)当用户只了解被测程序的一些明显的特点但并不了解它会导致哪种特定的错误时，系统在获取用户给出的特点后先推理可能存在的并发错误类型，再根据该错误类型给出推荐的测试工具。3)用户对待测程序并不了解，只能给出期望的测试结果(如可视化的测试结果等)，系统根据用户提出的需求进行推荐。模块关系如图 4 所示。

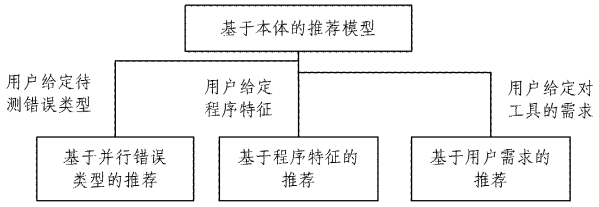


图 4 推荐模型的架构

另外，若用户给出的信息量足够大或一个模块给出不止一种的推荐工具时，若用户希望查找含有死锁类型的并发错误(基于特定并发错误类型)，且要求测试工具开源，测试工具可应用于 Windows 平台等(基于用户需求)，3 个模块可进行交互。当涉及模块的交互时可通过 2 个模块单独推荐，筛选出共同的结果，从而提高准确率。

4.1 基于特定并发错误类型进行工具推荐

根据图 3 以及 3.2 节介绍的专门针对某个类型的并发错误测试工具可以构建本体，实体类如图 5 所示，类之间的关系如图 6 所示。

建立本体之后，还要使用 Jena 推理机对其进行完善。实现本体推理的关键前提是制定本体推理规则，推理规则说明

了概念之间的关联性和推导关系。推理规则的格式为:

[rule_name:(a R_1 b)(b R_2 c)→(a R_3 c)]

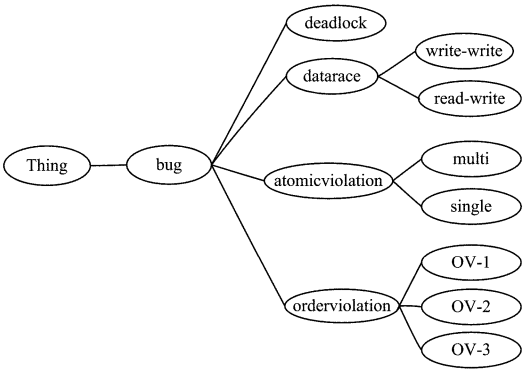
其中,rule_name 是规则的名称,R_1,R_2,R_3 分别表示 a 和 b,b 和 c,a 和 c 之间的关系,其中 R_3 是由 R_1 和 R_2 推理出来的,即通过已知的 a 和 b,b 和 c 之间的关系可以推出隐含的 a 和 c 之间的关系。

根据上述关系,在针对特定并发错误类型进行工具推荐的模块中定义推理关系:若某工具 a 可以对特定错误 b 进行检测,而测试人员希望对遇到的用例程序 c 进行针对错误 b 的检测,则系统将推荐使用工具 a。在本体推理规则中描述为:

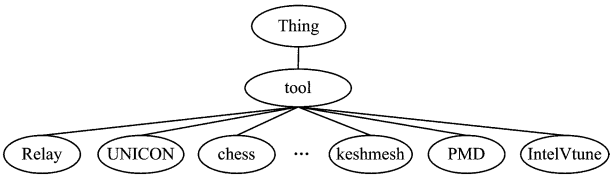
[rule1:(?a solve ?b)(?c has ?b)→(?c canfix ?b)]

由于关系可逆,因此亦可表示为:

[rule2:(?b solvedby ?a)(?b in ?c)→(?c canfix ?b)]



(a)基于特定错误类型进行工具推荐的本体实体类(1)



(b)基于特定错误类型进行工具推荐的本体实体类(2)

图 5　基于特定错误类型进行工具推荐的本体实体类

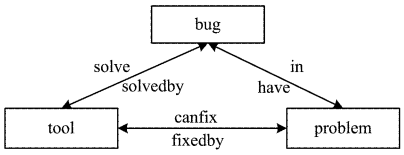


图 6　基于特定错误类型进行工具推荐的本体类关系

4.2　基于程序特征进行工具推荐

根据表 1 和表 2 可整理出现存并发错误分类的特点,根据这些特点,可以使用 protégé 建立合适的基于程序特征进行工具推荐的本体。实体类如图 7 所示,类之间的关系如图 8 所示。

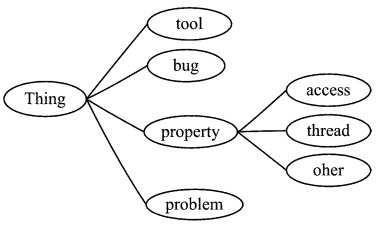


图 7　基于程序特征进行工具推荐的本体实体类(bug 类和 tool 类已在基于特定并发错误类型进行工具推荐的模块中说明)

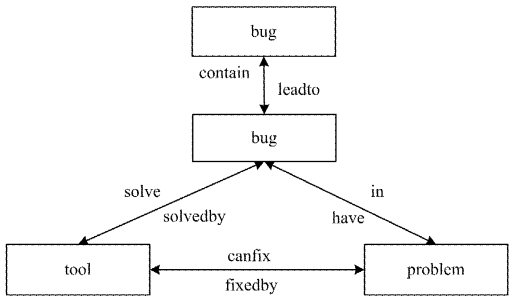


图 8　基于程序特征进行工具推荐的本体类关系

同样地,构建完本体后也需要使用 Jena 推理机对其进行完善,定义推理规则:对用户给出待测程序的一些特点,通过初步推理推测其属于哪一种现存的并发错误,再找出针对该并发错误的并发系统测试工具,从而最大可能地满足用户的需求,以达到提高效率并准确找到所需的测试工具的目的。即若待测程序 a 的特点 b,c 和 d 共同导致并发错误 e,而 f 是针对 e 的并发系统测试工具,则为该用户推荐并发系统测试工具 f。因此,添加如下推理规则:

[rule1:(?a contain ?b)(?a contain ?c)(?a contain ?d)(?b leadto ?e)(?c leadto ?e)(?d leadto ?e)(?e solvedby ?f)→(?f canfix ?a)]

4.3　基于用户需求进行工具推荐

还存在一种不容忽视的情况,即用户对并发错误分类和程序本身的特点都不了解,只能提出一些类似可视化、价格低之类的需求,因此再添加一个根据用户需求进行工具推荐的本体。

根据 4.2 节中提到的相关信息,构建基于用户需求进行工具推荐的本体。实体类如图 9 所示,类之间的关系如图 10 所示。

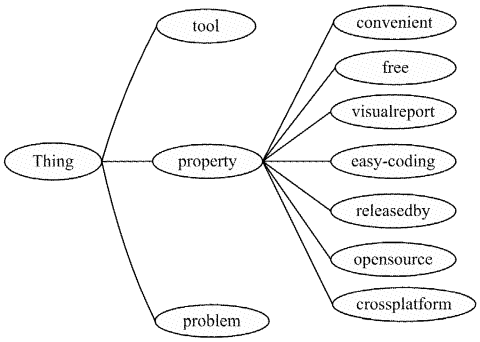


图 9　基于用户需求进行工具推荐的本体实体类

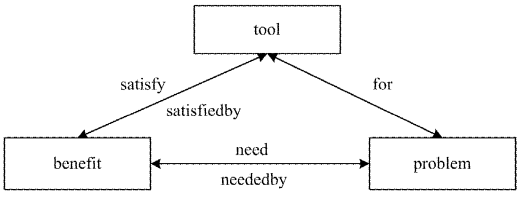


图 10　基于用户需求进行工具推荐的本体类关系

建立本体之后,还要使用 Jena 推理机对其进行完善。定义推理规则:若用户需要满足性质 b,c,d 的测试工具,而测试工具 e 恰好可以满足这些性质,则系统将推荐使用工具 e。在本体推理规则中描述为:

[rule1:(?a need ?b)(?a need ?c)(?a need ?d)(?e satisfy ?b)(?e satisfy ?c)(?e satisfy ?d)→(?e for ?a)]

5 实验

推荐本体采用 Java 语言开发,实验所使用的机器是一台 CPU 为 Intel(R) Core(TM) i3-3110M 2.40GHz、内存为4GB 的普通笔记本电脑,所使用的操作系统为 Windows 7 旗舰版,Java 运行环境为 JDK 1.7。测试的基准为来自卡耐基梅隆大学提供的一些并发错误实例,判断成功的标准是推荐工具的耗时比所有成功检测错误的工具的平均耗时更少。在基于特定错误类型进行工具推荐模块和基于程序特征进行工具推荐模块的实验中,利用表 3 中的测试工具对并发错误实例进行测试,并计算成功找到错误的平均耗时;然后根据程序特性分别进行推荐,使用本体推荐的错误测试工具来进行测试,若推荐软件测试失败或超过记录的平均耗时,则认为推荐失败,否则认为推荐成功。对于基于用户需求进行工具推荐模块,给出一些用户的需求,再使用本体推荐的错误测试工具来进行测试,并计算符合用户需求的概率。

(1)基于特定错误类型进行工具推荐模块的实验

例如,对目标程序进行针对原子性违背的检测,总共 14 个软件的平均耗时是 0.03s,根据本体推荐使用 AtomFinder 或 Relay 工具进行检测,工具的检测耗时分别为 0.026s 和 0.029s,少于 0.03s。虽然时间差较小,但 0.03s 是 14 个软件的平均耗时,其中部分工具的检测时间较长,因此仍然认为推荐成功,且本次成功率为 100%。

(2)基于程序特征进行工具推荐模块的实验

例如,对目标程序进行测试,已知目标程序对语句的原子性要求,且进程数量大于空闲处理器内核数量,总共 14 个软件的平均耗时是 0.03s,根据本体推荐使用 AtomFinder 或 Relay 工具进行检测,工具的检测耗时分别为 0.026s 和 0.029s,少于 0.03s,认为推荐成功,且本次成功率为 100%。

(3)基于用户需求进行工具推荐模块的实验

例如,用户希望使用开源、简单易懂、有可视化结果的工具,经本体推荐,认为 Jest 可以满足用户需求,经实际使用测试得到 Jest 确实可以满足用户要求,则认为对此程序的推荐成功率为 100%。

通过对 10 个具体实例进行实验,得到的结果如表 4 所列。计算可知,基于特定错误类型进行工具推荐模块的成功率达到 88.71%,基于程序特征进行工具推荐模块的成功率达到 71.07%,基于用户需求进行工具推荐模块的成功率达到74.82%,平均成功率为 78.2%。由于条件所限,在未来的研究中会进行更多的实验。

表 4 推荐结果成功率

实例	基于特定错误类型/%	基于程序特征类型/%	基于用户需求/%
1	63.6	54.5	75.0
2	81.8	72.7	57.8
3	100	62.5	66.3
4	83.3	66.7	89.3
5	100	66.7	67.2
6	87.5	62.5	70.0
7	100	63.3	67.5
8	80.0	100	88.4
9	100	80.0	76.7
10	90.9	81.8	90.0

结束语 通过对本体和并发错误以及并发错误测试工具的研究,给出并实现了一个简单的基于本体的并发错误相关测试工具的推荐方法。实验证明,该方法可以完成相应测试工具的推荐,但是还存在一些不足:

(1)本文构建的本体中只包含了已知的易分类的并发错误和比较典型的几种并发系统测试工具,因此本体的覆盖范围小,推理后知识点之间的联系还不够。

(2)本文的本体规模较小,且全部通过手工构建,下一步的目标是研究如何通过机器学习等方法来实现本体的自动、半自动构建与完善。

(3)需要进一步在较大规模数据上进行实验验证。

随着研究工作的进一步开展,相信以上的展望会逐步实现,并发错误及其相关系统测试工具本体的构建会更加完善,也能为用户推荐更为高效的并发系统测试工具,从而提高测试效率。

参 考 文 献

[1] RODRÍGUEZ M Á. Creating a semantically-enhanced cloud services environment through ontology evolution[J]. Future Generation Computer Systems,2014,32(1):295-306.

[2] LEESATAPORNOWONGSA T,GUNAWI H S. SAMC:a fast model checker for finding heisenbugs in distributed systems (demo)[C]// Proceedings of the International Symposium on Software Testing and Analysis(ISSTA'15). 2015:423-427.

[3] CSUTORAS C,KISS A. Ontology visualization methods—a survey[J]. ACM Computing Surveys,2015,39(4):415-416.

[4] ASUNCIÓN GÓMEZ-PÉREZ,M,CORCHO O. Ontological Engineering[J]. Advanced Information & Knowledge Processing, 2004,47(2):69-75.

[5] ASADOLLAH S A,HANSSON H,SUNDMARK D,et al. Towards Classification of Concurrency Bugs Based on Observable Properties[C]// International Workshop on Complex Faults and Failures in Large Software Systems. IEEE Press,2015:41-47.