CODAS:一个易扩展的静态代码缺陷分析服务

梁广泰 王千祥
(北京大学信息科学技术学院软件研究所 高可信软件技术教育部重点实验室 北京 100871)

摘 要 利用静态代码缺陷分析技术对软件进行早期缺陷检测,是提高软件质量的重要途径。静态代码缺陷分析工具(如FINDBUGS,JLINT,ESC/JAVA,PMD,COVERITY等)已经被证实可以成功地识别出大量的软件潜在缺陷[1-3]。然而,这类工具在可用性和有效性方面的不足严重限制了它们的进一步广泛使用。可用性不足包括a)每个独立缺陷检测工具只擅于检测特定类型的缺陷,需要配合使用才能全面检测缺陷;b)每个缺陷检测工具的安装、配置和运行占用了用户大量的时间、精力。有效性不足包括静态缺陷分析结果往往存在大量误报,并且会包括许多不重要的(不会引起程序员修复行为的)缺陷报告。为了解决上述问题,提出并构建了一个易扩展的"静态代码缺陷分析"服务(Code Defect Analysis Service,CODAS)。CODAS基于一个高度可扩展的架构设计,对多个独立的缺陷检测工具进行了封装和集成,并对缺陷检测报告进行了有效汇总和排序,从而充分发挥了各个独立工具的优势,大大提升了静态缺陷分析工具的可用性和有效性。

关键词 静态分析,代码缺陷分析,易扩展,服务
中图法分类号 TP391 **文献标识码** A
CODAS访问地址:http://codas.seforge.org

CODAS:An Extensible Static Code Defect Analysis Service

LIANG Guang-tai WANG Qian-xiang
(Key Laboratory of High Confidence Software Technologyies, Ministry of Education, Institute of Software, School of Electronics Engineering & Computer Science, Peking University, Beijing 100871, China)

Abstract Static defect analysis techniques are very useful in detecting defects at the early stage of software development process, which can improve the software quality effectively. The static code defect analysis tools such as FINDBUGS, JLINT, ESC/JAVA, PMD, and COVERITY can detect plenty of real defects, which has already been demonstrated[1-3]. However, these tools don't provide sufficient usability and effectiveness, which restricts their further application. The insufficient usability lies in two points. The first point is that each standalone tool is only good at detecting some certain types of defects, which means that developers need to use more tools to get a more comprehensive defect report. The second point is that developers need to manually setup, configure, and execute each standalone tool one by one, which is a very time-consuming process. The insufficient effectiveness lies in that: the static analysis warnings provided by these tools usually contain lots of false positives and also many trivial warnings that are not very important and won't be fixed by developers. In order to solve these issues, we proposed and implemented an extensible static defect analysis service: Code Defect Analysis Service (CODAS). Based on a highly extensible architecture, CODAS encapsulates and integrates multiple defect analysis tools seamlessly and also provides an effective warning prioritization algorithm, which synthesizes the advantages of different tools and improves their usability and effectiveness largely.

Keywords Static analysis, Code defect analysis, Extensible, Online service

1 引言

静态代码缺陷分析技术通过静态分析代码来推测程序运行时的表现行为,从而发现代码中可能存在的缺陷[4]。这类技术主要包括自动抽象解释[7]、定理证明[8]、模型检测[6]、符号执行[5]和基于缺陷模式的代码检查[9]等。静态缺陷分析技术通常针对独立于特定功能需求的编程缺陷(如空指针异常、资源泄漏、死循环等)进行检测。这类工具不仅可以自动化地检测软件系统发布前的早期缺陷,而且能够检测出很多通过常规测试手段难以发现的缺陷。近年来,这类工具的应用价

到稿日期:2011-05-07 返修日期:2011-07-01 本文受国家973项目(2009CB320703),国家自然科学基金(60773160),国家自然科学基金重点项目(61033006),国家创新研究群体科学基金(60821003)资助。
梁广泰 男,博士生,主要研究领域为代码缺陷分析等,E-mail:lianggt08@sei.pku.edu.cn;王千祥 男,博士,教授,主要研究领域为中间件、软件工程。

值逐渐得到了工业界的重视和认可[1-3,10]。

然而，目前的静态缺陷分析工具在可用性和有效性方面仍然存在着很多不足，制约了这类工具的进一步广泛应用。可用性方面的问题主要包括：a)每个独立缺陷分析工具都需要程序员手工地在其本地安装配置，这个过程会占用程序员大量的时间和精力；b)每个独立缺陷分析工具的运行依赖于程序员本地计算机的性能，当程序员本地机器的性能低下时，独立缺陷分析工具的计算效率不如人意；c)不同的缺陷分析工具优劣势不同，使用单一的缺陷分析工具无法较全面地进行缺陷检测。有效性方面的问题主要包括：a)静态分析技术往往静态地近似模拟受检项目的运行时过程，导致静态缺陷分析工具的缺陷报告中往往存在大量的误报(false positives)；b)除了误报，静态缺陷分析工具的缺陷报告中依然会存在很多“虽然正确但不足以引起程序员修复动作”(重要性或严重性较低)的缺陷报告(non-actionable true positives)。

基于上述分析，认为一个具有良好架构的在线静态代码缺陷分析集成服务可以在很大程度上缓解上述问题。在线集成服务所具有的优势主要包括：a)有效避免了程序员繁琐冗余的本地安装配置过程；b)释放了程序员本地机器的计算资源，让强劲的服务器运算能力为众多用户提供高效的缺陷分析服务；c)集成了多种缺陷分析工具以利于发挥各自优势，从而为程序员提供更为全面的缺陷检测报告；d)对多个工具的分析报告进行有效汇总和排序，可以将准确度高、重要程度高(以引起程序员修复行为)的缺陷报告优先报告，以提高程序员的排错效率。一个具有良好架构的在线静态代码缺陷分析集成服务存在如下挑战：a)缺陷检测能力的扩展性问题：如何保证当新的缺陷分析工具出现时快速高效地集成至现有系统中；b)数据存储能力的扩展性问题：如何保证随着用户量的增多，受检程序、中间处理结果、缺陷报告等数据的存储能力可以随需应变；c)缺陷报告的合理汇总和排序问题：如何有效识别出正确且能够引起程序员修复动作的缺陷报告，并将其优先报告，以提高程序员的排错效率。

针对上述问题和挑战，设计并构建了一个易扩展的在线静态代码缺陷分析服务 CODAS(Code Defect Analysis Service)。CODAS 基于一个高度可扩展的架构设计，对多个独立缺陷检测工具进行了有效封装和集成，对多个工具的缺陷检测报告进行了有效汇总和排序，为程序员提供了一个可以随时随地通过网络访问使用的在线代码缺陷分析集成服务。为了解决缺陷检测能力的扩展性问题，CODAS 设计了一个统一的“独立缺陷分析工具”服务化的封装接口，其有效屏蔽了底层被封装的独立缺陷分析工具的系统差异性，并提供了统一的访问方式以便于 CODAS 的灵活调用，使得 CODAS 的集成调度逻辑与底层工具实现进行了充分隔离，具有很强的通用性。当存在缺陷能力扩展需求时，只需要基于封装接口对新的缺陷分析工具进行服务化，就可以无缝地集成至 CODAS 中。为了解决数据存储能力的扩展性问题，我们基于分布式文件式系统 Hadoop 搭建了一个可扩展的数据存取服务，并将其独立部署，以供 CODAS 集中方便地进行数据(受检程序、中间分析结果、历史缺陷分析报告等)存储。为了解决缺陷报告的合理汇总和排序问题，我们基于机器学习的方式对开源软件的缺陷修复历史进行挖掘学习，从而自动化地识别出最有可能引起程序员修复行为的缺陷报告并将其优先报告[11]。

本文第 2 节整体介绍 CODAS 系统；第 3 节详细叙述 CODAS 在可扩展性方面的设计和实现；第 4 节在实验数据的基础上评估 CODAS 的缺陷分析能力；最后总结全文，并提出对未来工作的一些设想。

2 整体概述

CODAS 主要由 4 个部分组成：“数据存取服务”、“独立缺陷分析服务”、“其他基础服务”和“CODAS 组合服务”，如图 1 所示。

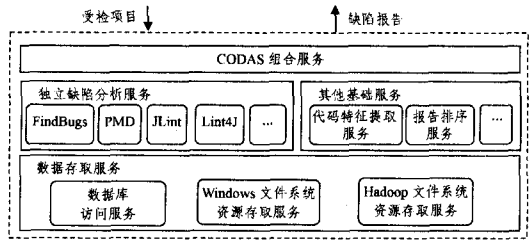

图 1 CODAS 设计架构图

“数据存取服务”负责为 CODAS 系统集中灵活地提供多种数据访问服务，主要包括“数据库访问服务”、“Windows 文件系统资源存取服务”和“Hadoop 文件系统资源存取服务”等。

“独立缺陷分析服务”指的是对集成至 CODAS 中的各个独立的静态缺陷分析工具分别进行封装后的服务集合。CODAS 中集成的各个独立缺陷分析工具(目前包括 FindBugs，PMD，JLint 和 Lint4J，后期可以根据需要灵活增加其他独立缺陷分析工具)都分别被封装成独立的服务，并以统一的标准化接口对外提供访问。基于统一的标准化接口对各个独立缺陷分析工具的封装，有效屏蔽了各个缺陷分析工具的系统差异性，为 CODAS 缺陷分析能力的可扩展性提供了良好的基础。

“其他基础服务”包括了为辅助 CODAS 完成整体缺陷分析功能的一些其他的基础性服务。其中的“代码特征提取服务”可以根据特定的缺陷报告，分析出其相关的受检代码的特征信息，如该缺陷所在方法的代码行数、该缺陷所在类相关的缺陷报告总数等(详见文献[11])。“报告排序服务”则负责根据受检代码的特征信息为每个缺陷分析报告预测“排序得分”(准确且容易引起程序员修复行为的缺陷分析报告的排序得分较高)，继而为缺陷分析报告进行排序。

“CODAS 组合服务”通过对“独立缺陷分析服务”、“其他基础服务”和“数据服务”的组合调用、并发执行、同步控制等，向用户提供完整的缺陷分析服务。“CODAS 组合服务”面向各个独立缺陷分析服务的访问接口进行集成，提高了 CODAS 的可扩展性。

CODAS 的部署结构图如图 2 所示。CODAS 采用 B/S 架构，程序员通过浏览器访问特定 URL(http://codas.seforge.org)便可以开始享用该服务。在服务器端，CODAS 尽量使得其内部的功能模块化，进而独立部署，以提高整体的并发执行能力。CODAS 中的“数据服务”、“独立缺陷分析服务”和“其他基础服务”都分别被部署至独立节点，这样各个功能

模块可以尽量并发执行，从而提高整体系统的响应速度。除此之外，“CODAS组合服务”被部署为多个拷贝，以分销请求压力。“请求分发服务”负责根据整个系统的动态负载情况进行灵活的请求分发。

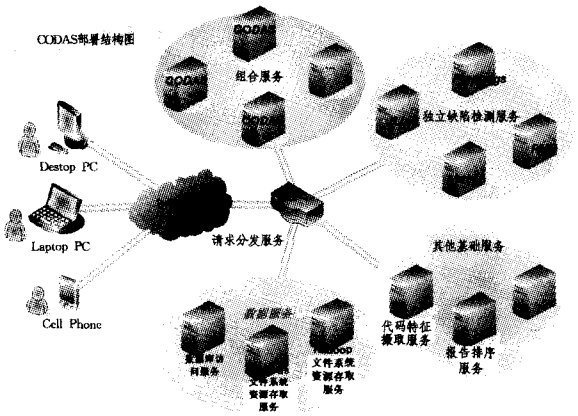

图2 CODAS部署结构图

3 技术要点详述

3.1 数据服务

为了满足CODAS灵活集中的数据存取需求，本系统采用3个独立节点分别提供服务：数据库访问服务、Windows文件系统资源存取服务以及Hadoop文件系统资源存取服务。为了确保数据安全性，3个数据服务器仅支持局域网范围内的内部访问。

数据库访问服务节点目前仅支持MySQL数据库。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了MySQL作为网站数据库。局域网内其他节点可以通过部署在该节点(其IP地址假如为192.168.1.1)上的MySQL数据库的JDBC连接URL(形如jdbc:mysql://192.168.1.1:3306/codas?user=root&password=your_password)进行局域网内的数据库访问。当系统存在其他类型数据库访问需求时，该节点支持的数据库类型可以随意扩展。

Windows文件系统资源存取服务通过一个独立节点(其IP地址假如为192.168.1.2)对外提供Windows文件系统资源的集中存储和访问功能。该节点的巨大磁盘空间通过“文件夹共享”的方式，在局域网范围内集中地提供文件存储与访问的功能。局域网内其他节点可以通过两种方式访问该“共享文件夹”：a)直接通过该“共享文件夹”的局域网访问路径(如\\192.168.1.2\share)进行访问；b)将该共享文件夹映射成为本地机器的网络驱动器(如z盘)，这样便可如同访问本地文件资源一样访问该共享文件夹中的任何文件。当该“共享文件夹”的空间紧张时，可以仅对该节点通过扩容或更换磁盘等方式解决(在该“共享文件夹”对外访问路径不发生变化的情况下，其他节点的配置信息和访问方式无需任何更改)。

Hadoop文件系统资源存取服务目前通过一个独立节点(其IP地址假如为192.168.1.3)对外提供Hadoop文件系统资源的集中存储和访问功能。Hadoop实现了一个分布式文件系统(Hadoop Distributed File System，HDFS)。HDFS的最大优势在于其非常强大的可扩展性。随着存储需求的不断增加，HDFS可以毫无限制地通过增加存储节点的方式快速扩展存储能力，且该扩展过程对于外界访问用户来说完全透明。CODAS系统中搭建的Hadoop文件系统通过URI(形如“hdfs://192.168.4.216:9000/user/codas/”)在局域网范围内对外提供存取操作。

3.2 独立缺陷分析服务

为了提高缺陷分析能力的可扩展性，CODAS采用松耦合的方式集成各个独立缺陷分析工具。集成至CODAS的各个缺陷分析工具被封装部署为独立的Web服务，并且以统一的“外部访问接口”对外提供访问。基于统一的标准化接口对各个独立缺陷分析工具的封装，有效屏蔽了各个缺陷分析工具的系统差异性，为CODAS缺陷分析能力的可扩展性提供了良好的基础。表1—表6分别给出了其中的6个标准化接口的具体描述。

表1 接口1“根据指定位置获取受检程序压缩包”

请求方法1	
请求格式	String retrieveProgramUnderAnalysis (String compressedFileLocalPath)
功能描述	根据Windows文件系统本地路径获取特定位置的受检程序
输入参数	compressedFileLocalPath[String]：受检程序压缩包的Windows文件系统访问路径
输出参数	compressedFileName[String]：获取后的受检程序压缩包的名字
处理过程	为了提高分析速度，将位于Windows文件系统特定位置的受检程序压缩包(该压缩包可能位于网络驱动器上)复制到缺陷分析工具的工作空间
请求方法2	
请求格式	String retrieveProgramUnderAnalysis (String compressedFileURL)
功能描述	根据网络URL获取特定位置的受检程序
输入参数	compressedFileURL [String]：受检程序压缩包的网络访问URL字符串
输出参数	compressedFileName[String]：获取后的受检程序压缩包的名字
处理过程	根据给定的URL将受检程序压缩包下载到缺陷分析工具的工作空间
请求方法3	
请求格式	String retrieveProgramUnderAnalysis (String compressedFileURI)
功能描述	根据Hadoop文件系统(HDFS)统一资源标识符(URI)获取特定位置的受检程序
输入参数	compressedFileURI [String]：受检程序压缩包的Hadoop文件系统统一资源标识符
输出参数	compressedFileName[String]：获取后的受检程序压缩包的名字
处理过程	根据给定的Hadoop文件系统访问路径将受检程序压缩包下载到缺陷分析工具的工作空间

表2 接口2“对受检程序进行格式转换或结构调整”

请求方法	
请求格式	String formatProgramUnderAnalysis (String compressedFileName)
输入参数	compressedFileName [String]：获取后的受检程序压缩包的名字
输出参数	formattedProgramName[String]：调整后的受检程序目录名称
处理过程	根据特定缺陷分析工具的要求，对受检程序压缩包进行解压缩，源码编译，结构调整，配置文件补充等操作

表3 接口3“对受检程序启动缺陷分析过程”

请求方法	
请求格式	Boolean analyzeProgramUnderAnalysis (String formattedProgramName)
输入参数	Result[Boolean]：执行结果(true代表顺利完成，false代表分析过程中出现异常)
输出参数	formattedProgramName[String]：调整后的受检程序目录名称
处理过程	通过命令行调用或API程序调用等方式启动缺陷分析工具对特定受检程序的缺陷分析过程

表 4　接口 4“解析缺陷分析报告并以统一的格式保存”

请求方法 1	
请求格式	Boolean parseReportAndSaveIntoDatabase (String formattedProgramName)
功能描述	解析缺陷分析报告并提取特定信息保存至数据库中
输入参数	formattedProgramName[String]：调整后的受检程序目录名称
输出参数	Result[Boolean]：执行结果(true 代表顺利完成，false 代表分析过程中出现异常)
处理过程	根据不同缺陷分析工具的情况，解析其生成的缺陷分析报告，提取出必要的缺陷报告信息，以统一的形式保存至数据库中
请求方法 2	
请求格式	Boolean parseReportAndSaveAsXMLFile (String formattedProgramName)
功能描述	解析缺陷分析报告并保存成 XML 格式缺陷分析报告
输入参数	formattedProgramName[String]：调整后的受检程序目录名称
输出参数	Result[Boolean]：执行结果(true 代表顺利完成，false 代表分析过程中出现异常)
处理过程	根据不同缺陷分析工具的情况，解析其生成的缺陷分析报告，提取出必要的缺陷报告信息，以统一的形式保存至 XML 格式缺陷分析报告文件中
请求方法 3	
请求格式	Boolean parseReportAndSaveAsHTMLFile (String formattedProgramName)
功能描述	解析缺陷分析报告并保存成 HTML 格式缺陷分析报告
输入参数	formattedProgramName[String]：调整后的受检程序目录名称
输出参数	Result[Boolean]：执行结果(true 代表顺利完成，false 代表分析过程中出现异常)
处理过程	根据不同缺陷分析工具的情况，解析其生成的缺陷分析报告，提取出必要的缺陷报告信息，以统一的形式保存至 HTML 格式缺陷分析报告文件中

表 5　接口 5“获取特定格式的缺陷分析报告位置”

请求方法	
请求格式	String getReportLocation(String format)
输入参数	format[String]：待获取访问路径的缺陷分析报告所对应的具体存储格式
输出参数	reportFilePath[String]：格式化后的特定格式的缺陷分析报告访问完整路径
处理过程	根据不同缺陷分析工具的封装过程，获取到特定格式的缺陷分析报告的访问路径并返回

表 6　接口 6“获取缺陷分析过程的异常描述信息”

请求方法	
请求格式	String getLogMessage (String formattedProgramName)
输入参数	formattedProgramName[String]：调整后的受检程序目录名称
输出参数	logMessage [String]：执行过程中的异常日志消息
处理过程	根据不同缺陷分析工具的封装过程，获取其执行过程中的最后报告出的异常信息并返回

3.3　其他基础服务

为了更为合理地排序缺陷分析报告以提高程序员的排错效率，CODAS 为每一条缺陷报告计算一个“排序得分”(Ranking Score)，并以此对缺陷分析报告进行排序。在收集到各个缺陷分析工具的分析报告后，CODAS 基于机器学习的方式，根据 3 类影响因子，为每个缺陷分析报告计算“排序得分”。3 类影响因子为缺陷分析报告提供描述信息、基于缺陷分析报告的统计信息以及与缺陷分析报告相关的受检代码特征信息等。

3.3.1　代码特征提取服务

CODAS 中独立部署的“代码特征提取服务”负责为缺陷分析报告相关的受检代码提取 8 个特征信息：文件深度(以文件行数百分比的形式计算出的缺陷相关代码位置所在的文件深度)、代码长度(缺陷相关的代码文件的代码行数)、注释长度(缺陷相关的代码文件的注释行数)、注释代码比例(缺陷相关的代码文件的注释行数与代码行数的比值)、方法深度(以文件行数百分比的形式计算出的缺陷相关的方法所处的文件深度)、方法长度(缺陷相关的方法的行数)、方法调用者数量(调用了该缺陷相关方法的方法数量)、方法调用数量(被该缺陷相关方法掉用的方法数量)。针对这些代码特征进行提取的背后原理详见文献[11]。

3.3.2　报告排序服务

CODAS 中独立部署的“报告排序服务”负责基于上述 3 类影响因子为每条缺陷分析报告预测排序得分，并按照得分的高低进行排序。该预测过程基于机器学习的方式，通过一个提前训练好的预测器进行预测。该预测器的训练过程所依赖的训练集是通过我们之前提出的方法[11]自动构建的。该训练集自动构建过程的核心思想是：通过挖掘开源项目的代码跟踪系统(issue-tracking systems)和版本控制系统(source-code repositories)中存储的信息，自动化地学习，总结出开源项目的缺陷修复历史，并据此对静态缺陷分析报告进行自动化评估，从而得到训练集。该训练集的构造过程主要分成 4 个主要步骤：通用缺陷修复版本的识别(根据代码跟踪系统和版本控制系统中记录的信息，识别出版本控制系统中的修复通用缺陷的代码提交版本)；通用缺陷相关代码的定位(通过通用缺陷修复过程中的代码变化情况反推通用缺陷相关的代码位置)；缺陷分析报告的生成(应用静态缺陷分析工具对开源项目的某特定版本进行缺陷检测，以获取缺陷分析报告)；训练集的提取(利用挖掘出的通用缺陷相关代码等信息对缺陷分析报告进行自动评估，以生成训练集)。该训练集的自动构建过程详见文献[11]。

3.4　组合服务

“CODAS 组合服务”以 Web 页面形式捕捉用户需求并通过对“独立缺陷分析服务”、“其他基础服务”和“数据服务”的组合调用、并发执行、同步控制等操作，为用户提供完整的缺陷分析服务。“CODAS 组合服务”面向各个独立缺陷分析服务的访问接口进行集成，提高了 CODAS 的可扩展性。

CODAS 与用户之间的具体交互过程及响应过程如图 3 所示。当用户存在缺陷分析需求时，用户请求会首先到达 CODAS 服务器的“负载均衡节点”，该节点会根据当前的负载情况选择一个合适的 CODAS 组合服务节点来响应用户请求。CODAS 服务器中的“CODAS 组合服务”被部署为多个实例(不同实例被部署至不同的节点之中)，负载均衡节点会记录单位时间内已分发至各个节点的请求数量，新的用户请求会被分发至单位时间内响应请求量最小的节点上。接收到用户请求后，“CODAS 组合服务”会首先生成用户访问界面，以响应用户请求。通过该访问界面，用户可以上传符合一定格式要求的受检程序压缩包，并启动缺陷检测过程。CODAS 组合服务在完整接收到用户上传的受检程序后(目前实现的 CODAS 将受检程序统一保存至 Hadoop 文件系统之中)，通过并发地调用部署于不同节点的独立缺陷分析服务来提高响应速度。由于每个独立缺陷分析服务对外提供了一组统一的访问接口，因此对于每个独立缺陷分析服务，CODAS 组合服务采用了一致的调用过程。对于每个独立缺陷分析服务，CODAS 会首先通知其受检程序的访问路径。根据受检程序的访问路径，每个独立缺陷分析服务会将其下载至各自的工作空间中。在完成受检程序的下载过程后，CODAS 组合服务通过调用各个缺陷分析服务的接口对受检程序进行解压缩、格式转换或结构调整等，之后启动各个缺陷分析服务的具

体缺陷分析过程。在各个缺陷分析服务完成缺陷分析后，CODAS组合服务通过相应的接口调用解析各个分析服务的缺陷分析报告，并以统一的格式将缺陷分析报告集中地保存至数据库中(以统一的方式存储缺陷分析报告有利于屏蔽各个独立缺陷分析服务的报告格式的异构性)。在完成缺陷检测过程后，将启动缺陷排序过程。CODAS根据不同的影响因子为每条缺陷分析报告计算排序得分。该排序得分的高低反映了缺陷分析报告的准确程度(Precision)和引起程序员修复动作的概率(Actionability)的高低。根据该“排序得分”进行报告降序排列的结果会将“准确程度高且更容易引起程序员修复行为”的缺陷分析报告优先报告给程序员。这种报告排序方式可以在一定程度上提高程序员的排错效率。在进行排序之前，CODAS首先为每条缺陷分析报告计算排序得分影响因子值。影响因子分为3类：第一类影响因子为缺陷分析报告提供的描述信息，该类信息可以直接从各个缺陷分析服务的报告中得到；第二类影响因子是基于缺陷分析报告的统计信息，该类信息通过对所有的缺陷分析报告进行自动化统计而得到；第三类影响因子是每条缺陷分析报告相关的代码特征的提取，该类信息的提取是通过部署于独立节点上的“代码特征提取服务”完成的。影响因子的具体介绍及其对应的提取过程参考文献[11]。在完成影响因子值的提取后，CODAS利用内置的、提前训练好的“排序得分预测器”为每一条缺陷分析报告计算“排序得分”。该排序得分预测器的训练集构造及其训练过程详见文献[11]。完成排序得分的预估后，CODAS会根据排序得分对所有的缺陷分析报告进行降序排列，形成缺陷分析报告并返回给用户。用户可以基于该返回的缺陷分析报告进行相应的缺陷修复动作。

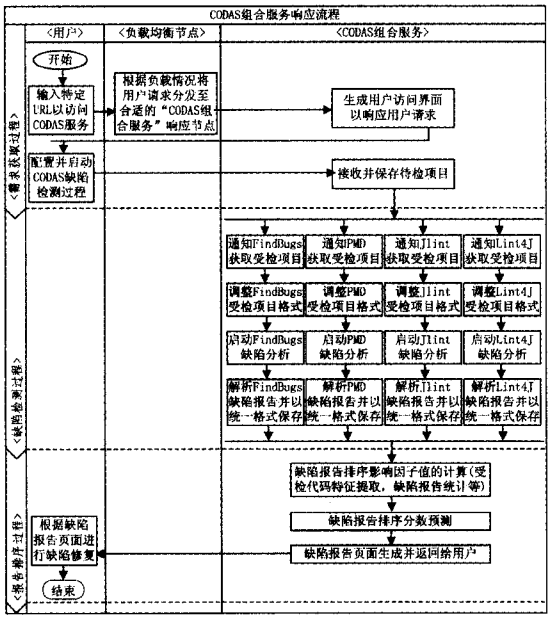

图3 CODAS组合服务响应流程图

4 实验

为了评估CODAS的有效性，设计了如下实验。实验通过评估受检项目的CODAS缺陷分析报告的质量来评估CODAS的有效性。本试验中选取的受检项目包括Lucene，Spring，ANT，Log4J，Tapestry(受检项目详细信息见表7)。

表7 受检项目信息表

受检项目	社区	版本管理系统	受检版本号
Lucene	Apache	SVN	1500
Spring	Apache	SVN	15301
ANT	Sable. MCGill	SVN	6300
Log4J	Apache	SVN	20
Tapestry	Apache	SVN	4800

实验中，通过对受检项目的缺陷检测报告进行质量评估来评估CODAS的有效性。实验中，采用“前n条报告的准确度”(top n warnings' precision)这一指标进行评估。如果一个受检项目的缺陷报告的确在后期的项目代码中被修复，则认为该缺陷报告是准确的。通过这样的判断原则，便可以计算出排在报告前列的任意前n条报告的准确度值。图4分别展示了统计出的Lucene，Spring，ANT，Log4J，Tapestry等项目的缺陷报告的累积准确度变化图。图4中的坐标点(x,y)代表排在报告的前x条缺陷分析报告的准确度为y，其中标为“CODAS”的曲线代表了CODAS缺陷报告的累积准确度变化情况，其中标为“FindBugs”，“PMD”，“Jlint”和“Lint4J”的曲线分别代表了CODAS所集成的独立缺陷分析工具FindBugs，PMD，Jlint和Lint4J的各自缺陷报告的累积准确度变化情况。

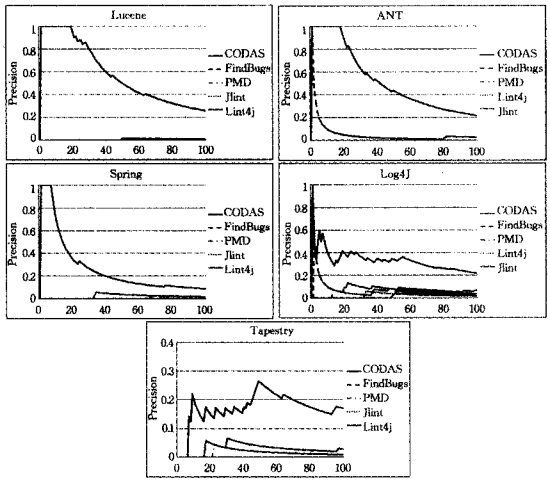

图4 CODAS缺陷报告累积准确率评估图

通过实验结果图可以发现，CODAS通过对多个工具的有效集成和合理排序，大大提高了前序报告的准确程度。对于所有受检项目来说，CODAS在其前100条缺陷报告的范围内的准确度一直都高于各个独立缺陷工具。而对于受检项目Lucene，ANT和Spring来说，CODAS在这些受检项目上所报出的前19，18，7条报告的准确度始终保持100%，前29，23，8条报告的准确度始终高于80%。然而，各个独立缺陷工具的报告的准确度一直处于较低的水平(低于10%)。实验充分说明了CODAS通过集成多个独立缺陷检测工具并进行合理排序，有效地改善了缺陷分析报告质量，这将在很大程度上提高程序员的缺陷排除效率。

结束语 为了解决现有缺陷分析工具的问题和不足，本文介绍了我们设计并构建的一个高度可扩展的在线静态代码缺陷分析服务CODAS(COde Defect Analysis Service)。CODAS基于一个高度可扩展的架构设计对多个独立缺陷检测工具进行了有效的封装和集成，对多个工具的缺陷检测报告进行了有效的汇总和排序，为程序员提供了一个随时随地可

(下转第64页)

的信息安全威胁评估模型是可行的，并且符合评估标准。

结束语 本文针对信息安全威胁评估问题，将模糊综合评价法和AHP方法相结合，识别出关键信息资产。建立层次模型后，考虑不同威胁发生的可能性，分别对每一层指标因素进行评估，从单资产威胁开始分析，再对综合威胁进行模糊综合评价，最后得到威胁评估结果。实验表明，本方法能够对信息系统面临的直接或间接的威胁进行有效评估，为科学管理提供了有力依据。但信息安全威胁存在动态复杂性，可采取预防、阻止、转移、检测和恢复来处理安全事件。为了有效地控制信息安全威胁，可采取5种方法来提高安全性：加密、软件控制、硬件控制、策略和过程、物理控制。不管采用何种安全措施，都要按照以下4大原则进行考虑：最易渗透原则、最弱环节原则、适度保护原则、有效性保护原则。

参考文献

[1] 赵冬梅，刘金星，马建峰. 基于改进小波神经网络的信息安全风险评估[J]. 计算机科学，2010，37(2)：90-93

[2] 宫亚峰，赵波，徐金伟. 再论信息安全资产的识别与评估[J]. 计算机安全，2010，2：5-10

[3] 何可，李晓红，冯志勇. 面向对象的威胁建模方法[J]. 计算机工程，2011，37(4)：21-23

[4] 陈清明，张俊彦. 信息安全风险评估工具及其应用分析[J]. 信息安全与通信保密，2010，1：93-95

[5] 蔡林，刘学忠. 基于攻击路径图的威胁评估方法[J]. 计算机应用，2009，6(29)：74-76

[6] Saaty T L. How to Make a Decision: The Analytic Hierarchy Process[J]. Interfaces, 1994, 24(6): 19-43

[7] Ashenden D. Information security management: A human challenge[J]. Information Security Technical Report, 2008(13): 195-201

[8] Garcia-Alfaro J, Barbeau M, Kranakis E. Analysis of Threats to the Security of EPC Networks[C]// Communication Networks and Services Research Conference. 2008

[9] OPR13等级测评报告[R]. 50000843002-10-5001-01. 2009：111

[10] 易昆南，晏玉梅. 改进的DS/AHP方法及应用[J]. 重庆理工大学学报：自然科学版，2011，25(5)：121-126

(上接第18页)

以访问并使用的在线代码缺陷分析集成服务云。在整体介绍CODAS之后，本文详细叙述了CODAS在可扩展性方面的设计和实现。为了解决缺陷检测能力的扩展性问题，CODAS设计了一个统一的“独立缺陷分析工具”服务化的封装接口。为了解决数据存储能力的扩展性问题，基于分布式文件式系统Hadoop，搭建了一个可扩展的数据存取服务并将其独立部署，以供CODAS集中方便地进行数据(受检程序，中间分析结果、历史缺陷分析报告等)存储。除此之外，我们也对数据库服务和Windows文件资源存取服务进行了独立部署，以满足CODAS的其他数据访问需求。为了解决缺陷报告的合理汇总和排序问题，我们基于机器学习的方式对开源软件的缺陷修复历史进行挖掘学习，从而自动化地识别出最有可能引起程序员修复行为的缺陷报告并将其优先报告[11]。为解决运算性能问题，我们尽可能将系统中的各个功能模块化进而独立部署，以提高整个系统的响应时间和并发访问量。本文还通过实验，分别评估了CODAS的有效性。通过实验，发现CODAS的缺陷报告能使得Lucene，ANT，Spring等受检项目的前22，19，7条缺陷报告的准确度始终高于90%。实验说明CODAS能够有效预测出准确且能够优先引起程序员修复动作的缺陷报告并将其优先报出。

在未来工作中，将围绕CODAS的可用性和有效性方面的不足继续完善本在线缺陷分析集成系统。我们可以通过提供同一受检项目不同版本的缺陷分析报告的比较功能，帮助程序员快速定位新版本引入的缺陷报告，以提高其排错效率。除此之外，还可以通过探索更好的在线缺陷报告反馈功能，高效地收集程序员对于缺陷分析报告的权威评价，从而进一步优化缺陷分析报告的排序效果。

参考文献

[1] Copeland T. PMD applied[M]. Centennial Books, 2005

[2] Ayewah N, Hovemeyer D, Morgenthaler J D, et al. Experiences using static analysis to find bugs[J]. IEEE Software, 2008, 25(5): 22-29

[3] Nanda M G, Gupta M, Chandra S, et al. Making Defect-finding Tools Work for You[C]// Proceedings of the International Conference on Software Engineering (ICSE). 2010: 99-108

[4] Nielson F, Nielson H R, Hankin C. Principles of Program Analysis(Corrected 2nd printing)[M]. Springer, 2005

[5] Boyer R S, Elspas B, Levitt K N. SELECT—a formal system for testing and debugging programs by symbolic execution[C]// Proceedings of the International Conference on Reliable Software. 1975: 234-245

[6] Clarke E M, Grumberg J O, Peled D A. Model Checking[M]. MIT Press, 2000

[7] Cousot P, Cousot R. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints[C]// Proceedings of the 4th ACM Symposium on Principles of Programming Languages (POPL). 1977: 238-252

[8] Lint4J[EB/OL]. http://www.jutils.com/

[9] Hallem S, Chelf B, Xie Y, et al. A System and Language for Building System-Specific, Static Analyses[C]// Proceedings of the ACM SIGPLAN 2002 Conference on Programming Language Design and Implementation(PLDI). 2002: 69-82

[10] Wagner S, Jrjens J, Koller C, et al. Comparing bug finding tools with reviews and tests[C]// Proceedings of the 17th International Conference on Testing of Communicating Systems. volume 3502 of Lecture Notes in Computer Science, June 2005: 40-55

[11] Liang Guang-tai, Wu Ling, Wu Qian, et al. Automatic Construction of an Effective Training Set for Prioritizing Static Analysis Warnings[C]// Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering. 2010: 93-102

[12] Hovemeyer D, Pugh W. Finding Bugs is Easy[C]// Companion to the 19th annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages and Applications (OOPSLA). 2004: 92-106

[13] Jlint[EB/OL]. http://artho.com/jlint