第 1 章 综述

随着互联网与计算机的发展，已经产生了数以亿万计的软件项目。以在开源软件库Github[①]上托管的项目为例，2018年官方统计[②] 数量已超过9600000 个，且增速达到40%。海量的软件项目也引出了繁重的软件维护工作，对开发人员提出了巨大的挑战。因此，如何辅助开发人员在软件维护过程更高效的完成代码修改工作，成为软件维护领域的研究热点。开发人员在修改代码过程中，会对软件中其他代码实体产生潜在的影响，必需对这些代码做相应的修改，从而提高了开发人员维护工作的难度和成本。同时，在代码修改完成后，通常需要经过代码审查人员的多次审查，才能最终完成代码修改任务。因此，分析代码修改会产生的影响范围以及预估代码修改的完成周期对提高软件维护的效率有关键作用。大数据背景下，如何从软件维护过程中产生的数据里挖掘出有用信息，用于辅助后续的代码修改任务，成为非常关键的研究问题。在Github等版本控制系统中，存在海量软件项目维护过程中产生的数据信息，代码修改信息以提交(Commit) 的形式保存。另外，在Gerrit[③]代码审核软件中也存在大量代码修改的审核信息。因此，本文针对代码修改的影响分析以及代码修改的完成周期进行深入研究，通过挖掘软件过程中的历史数据，辅助代码修改影响分析和预测代码修改的完成周期，提高代码修改任务的效率和质量。

1.1 论文研究背景与意义

软件的可维护性和可修改性是软件固有的重要特性。软件维护是整个软件生命周期中最关键的一环，占据着70%以上的比重，其主要任务是迎合市场和用户的新需求、修复软件运行过程中的存在的错误、以及对软件性能的优化。软件维护工作被认为是软件生命周期中，最困难和最费人力的工作[1]。软件维护过程中的核心是代码修改工作，由于软件系统的整体性以及系统中各部件的相互依赖关系，代码修改将不可避免地对修改以外的部分产生影响，从而影响

① Github, https://github.com
② Github, https://octoverse.github.com
③ Gerrit, https://www.gerritcodereview.com/

软件的稳定性。为了预估软件维护和代码修改过程的影响范围和程度，修改影响分析成为代码修改任务中的重要一步[2]。另外，代码修改任务的最后一步是代码审核，通常代码修改需要经过多次“修改－审核－再修改”的环节才能最终完成，提前对代码修改任务的完成周期进行预估，能有效降低软件维护工作的成本。随着软件系统变得越来越庞大和复杂，修改影响分析和修改完成周期的预测是软件维护过程中提高维护效率和质量的有效方法。

大数据背景下，通过挖掘历史数据中的有效信息，再根据历史信息对研究对象进行分析的研究方式得到了广泛的应用，而且研究结果往往有较高的适用性和准确性[3, 4, 5]。在ICSE、ASE和ICSM等计算机软件工程顶级会议中，关于数据挖掘在软件工程中应用的研究也吸引了广泛的关注。软件仓库中记录着软件演化的完整历史数据，包括：程序运行数据，缺陷跟踪数据，历史代码修改数据，代码审查数据。这些数据可用于挖掘重要信息，例如项目如何演变[6, 7]，开发人员如何合作[8, 9]，代码修改可能影响的范围[10, 11] 等。如图1-1所示，我们总结了在软件工程研究问题中运用数据挖掘方法的总体流程。软件维护和演化过程中存在大量软件工程过程中数据，特别是当前，开源项目广泛托管在Github等版本控制系统中、开源代码审核软件也应用更加广泛，使得软件维护和演化的相关数据获取更加便利。同时，近年来，机器学习取得飞速的发展，在各个领域的应用，都起到了对研究分析的推动作用。在软件工程领域，结合机器学习方法的研究工作，也取得了丰硕的成果。在本文，我们利用机器学习方法，研究开源项目中的代码修改数据和代码审核数据对代码修改影响分析以及对代码修改的完成周期预测的辅助作用。

开源项目在版本控制系统中的维护和演化，都是通过代码提交（Commit）的形式进行的。代码提交中的数据包括（如图1-2所示[①]）：（1）提交的注释文本，（2）修改前后的代码版本，（3）修改涉及范围，（4）提交编号，（5）提交作者和时间。这些数据包含软件演化过程中修复和改进信息，对后续维护工作起着重要作用[12]。代码修改数据在多个软件工程的研究领域都起到了辅助作用，例如：代码审核评论的自动生成[13, 14]，缺陷预测[15, 16]，修改影响分析[10, 11] 等。其中，修改影响分析有助于开发人员在修改代码时，预估可能影响的其他代码实体，辅助开发人员更加高效的完成代码修改工作。修改影响分

[①] Spring Boot, https://github.com/spring-projects/spring-boot/commit,2018

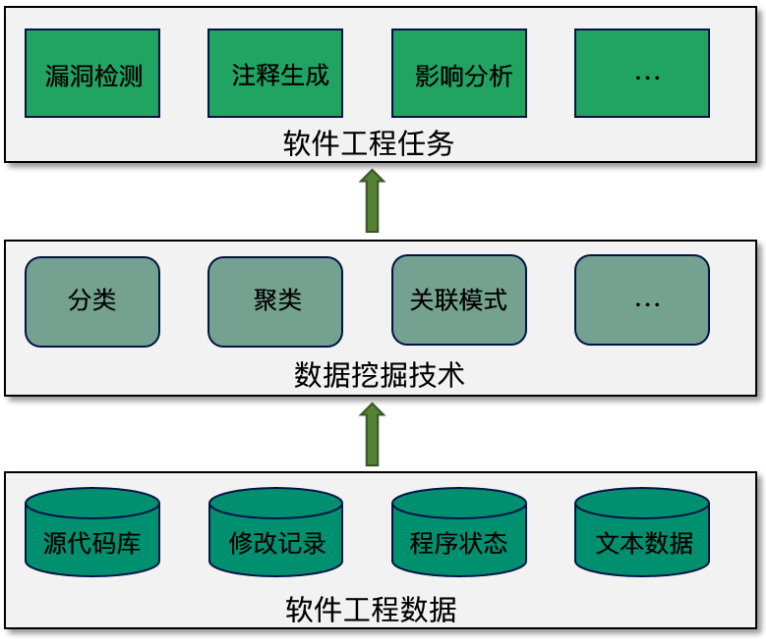


图 1-1 软件工程数据挖掘流程

析领域已经有长达30年的研究历史，但是主流的研究方法更加关注系统中代码实体之间的耦合关系以及代码运行信息，以此来分析可能受影响的范围。我们研究发现，大量代码修改工作诸如需求变更、缺陷修复等，都能在软件存储库（本项目或其他项目）中找到相似的代码修改任务。相似的代码修改任务中的代码修改范围对于开发人员确认修改影响范围有直接的借鉴作用。本文提出一种基于挖掘代码提交（Commit）信息中的修改模式来辅助修改影响分析的方法，通过关键类判定方法，将提交中的关键类等价为当前修改类，利用关键类的修改影响范围辅助分析当前修改的影响范围。

软件代码审查，即让第三方人员对软件系统的代码修改进行审核，是开源和专有软件领域中公认的最佳实践[17, 18]。研究表明，正式代码审核往往会提高软件维护的质量，延长软件生命周期。正式的代码检查流程要求严格的审核标准以确保审核质量的基本水平[19]。在过去的研究中，研究人员已经开发了许多工具和系统来管理软件生命周期的各个方面。对于开发人员而言，软件维护工作中的主要关注点是“如何最大限度的使代码修改提交通过审核人员审核，并最大限度地缩短修改完成时间”。原则上，代码审查是一个透明的过程，旨在评估修改代码的质量。但是，代码审查的执行过程可能受到各种因素的影响，包括技术因素如代码修改难度，也包括非技术因素如项目复杂度，开发人员和审核人员数量等，这些因素很大程度上影响着审查时间和修改的完成

图 1-2 代码提交示例

图 1-3 代码审核示例

时间。然而，很少研究关注代码审查过程中各因素对代码修改完成周期的影响。随着Gerrit等开源代码审查工具的出现，使得代码审查数据（如图1-3 所示[①]）容易被收集，分析和使用。在本文中，我们从Gerrit中收集代码修改的审查数据，提取有效特征，用于预估代码修改的完成周期。

1.2 国内外研究现状

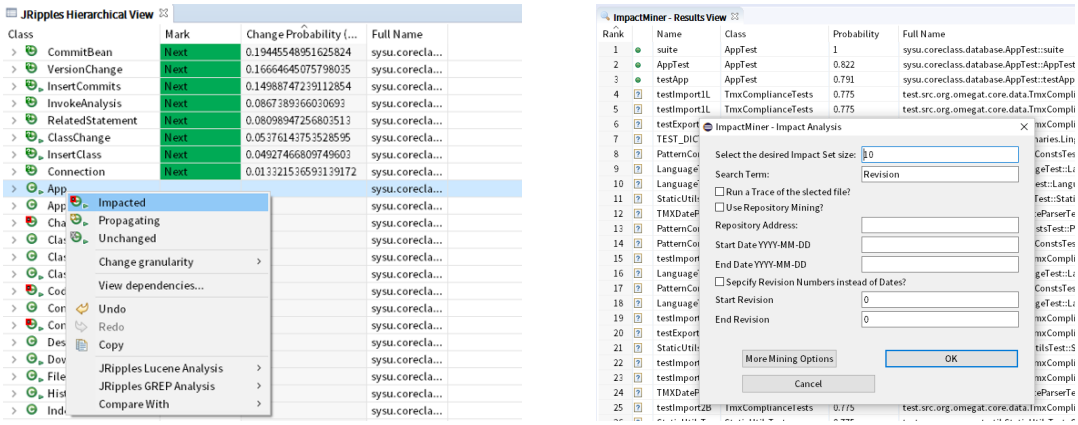
影响分析领域已有多年的研究历史，许多学者通过不同的方法对此问题进

① OPEN DAYLIGHT, https://git.opendaylight.org/gerrit/#/c/62848/1,2018

行研究，积累了大量研究成果。此外，得益于机器学习方法的飞速发展以及开源代码托管和审查工具的广泛使用，数据挖掘在软件维护中的应用也受到更多研究人员的关注。本文将从修改影响分析，代码修改模式挖掘，软件存储库挖掘和的代码修改完成周期预测四个方面介绍国内外研究现状以及与本文研究相关的工作。

1.2.1 修改影响分析

修改影响分析可以帮助开发人员理解代码修改，预测修改的影响范围和修改的潜在代价。代码修改往往会由于系统中各部件的关联性而相互波及，如果没有对受波及的部件做相应的调整，会造成各部件之间程序的不一致性[20, 21]。修改影响分析的目的是提前预估代码修改可能造成的影响范围和程度。Bohner等人[22]将影响分析定义为评估软件变更中所有要修改代码的工作。近年来，对于修改影响分析的研究工作不断增多，研究人员提出了许多影响分析的研究方法和支持工具(如图1-4）。



(a) JRipples① (b) ImpactMiner②

图 1-4 修改影响分析工具

修改影响分析方法主要分为静态影响分析[23, 24, 25]和动态影响分析[26, 27, 28, 29]。静态影响分析通过分析代码之间的语法、语义信息，获取软件部件之间依赖关系，计算修改影响范围。软件项目中某个代码片段的修改通过代码间的依赖关系，可能会传播给其他代码片段，因此分析代码修改的传播机制是静态分析的关键。Breech等人[30]总结了代码修改中的传播机制，并基于这些传播

① JRipples, http://jripples.sourceforge.net
② ImpactMiner, http://www.cs.wm.edu/semeru/ImpactMiner

机制建立影响传播图作为中间表示，通过传播图分析影响范围，使得静态分析的精确度得到很大提高。动态影响分析使用特定测试样例，收集程序运行过程中的数据信息（执行轨迹信息，数据流信息，控制流信息等）计算影响范围。Law 等人[31]提出的PathImpact方法通过收集程序运行时的轨迹信息，构建程序运行路径图，对路径图进行遍历得到修改影响范围。PathImpact方法也是动态分析中使用最广且精度较高的方法。总的来说，静态分析方法由于依赖关系复杂，得到影响范围过大，使得影响分析精度较低；而动态分析方法必须在测试样例运行结束后，才能收集程序运行信息，造成成本过高。针对这个问题，Cai 等人[32]提出了DIVER方法，在PathImpact基础上引入静态依赖图，对程序运行轨迹进行修剪，在较低的成本下得到更高精度影响分析结果。

根据使用技术的不同，修改影响分析还可以划分为基于静态语法依赖关系的影响分析[33, 34]，基于耦合关系的影响分析[35, 36]，以及基于软件存储库挖掘的影响分析[37, 38]等。基于静态语法依赖关系的影响分析是最常用的影响分析技术，通常在代码层次建立依赖图作为中间件，将依赖图中各代码实体之间的可达性作为影响传播的依据。代码层次获得的信息更加丰富，分析结果也更加精确。基于耦合关系的影响分析，计算代码实体之间的结构耦合、概念耦合等，通过耦合关系的强弱计算修改影响范围。基于软件存储库挖掘的影响分析则是根据软件存储库中的历史修改信息，挖掘历史修改中的修改影响范围，将历史修改的影响范围作为当前修改的影响范围。

Li等人[39]的研究表明，基于挖掘软件存储库的影响分析技术越来越受关注。挖掘软件存储库可以发现软件各部件之间重要的历史依赖关系，例如类之间，函数之间或者文档文件。软件维护者可以使用历史依赖关系分析历史变更中，修改影响是如何传播的，而不是仅依赖于传统的静态或动态代码依赖关系。例如，对一段将数据写入文件的代码进行修改，可能需要对从这个文件读取数据的代码进行相应的调整，但是由于两段代码之间不存在数据和控制流信息，传统静态和动态分析方法将无法捕获这些重要的依赖信息。因此，在传统静态或动态分析技术的基础上，挖掘软件存储库是影响分析方法的良好补充。已有的挖掘软件存储库的方法，仅根据当前修改代码实体，挖掘历史修改记录中该代码修改所产生的影响范围，没有涉及具体的修改内容，即对同一代码做不同的修改，通过历史修改记录所获得影响范围是相同的。本文针对这一问题，提

出一种新的历史数据挖掘思路。通过计算当前修改需求、修改代码与历史修改提交中修改描述、修改代码的相似度，得到与当前修改相似的修改提交信息，再使用关键类判定方法判断提交中核心修改的类，将关键类等价为当前修改的类，用提交中关键类的修改模式辅助确认当前修改的影响范围。

1.2.2 代码修改模式挖掘

软件维护是一个长期的过程，了解软件演化进程中的变更模式可以帮助开发人员更高效的完成维护工作。代码修改模式挖掘目的是挖掘历史修改信息中代码实体间是否在修改过程中有关联关系。代码修改中最常见的是同步修改模式，例如，如果从历史信息发现代码实体e_1总是与代码实体e_2同时进行修改，当开发人员再次对e_1进行修改时，可预估e_2需做相应的修改。根据代码同步修改模式，开发人员可以在面对代码修改需求时，预测需要共同修改的代码实体。Bouktif等人[21]定义了同步修改模式的一般概念，即描述在小时间范围内共同变化的代码实体，并使用模式识别中的动态时间扭曲技术对历史修改数据进行分组，提取相似的修改模式。Ying等人[40]通过基于频率计数的频繁模式挖掘技术从源代码更改历史中挖掘同步修改的源代码。Zimmermann等人[41]利用历史修改代码中的关联规则挖掘出代码实体的同步修改模式，并实现了一个原型系统ROSE[①]，为开发人员预测需共同变化的代码，预防因不完整修改而导致的错误，并且能检测出用传统程序分析方法无法得到的耦合关系。Ajienka等人[42]的研究表明，频繁同步修改的代码实体之间存在很强的耦合关系，挖掘语义耦合关系有助于识别修改模式。

除了同步修改模式外，代码修改中还存在许多其他修改模式。Jaafar等人[43]的研究中提出了两种新的代码修改模式，代码异步修改模式和代码移相修改模式。代码异步修改模式指的是在大时间区间内共同修改的代码；代码移相修改模式指的是频繁在相同时间间隔进行修改的代码。Stephane等人[44]通过聚类的方法，对一定时间周期内完成相似修改的代码进行归类，划分不同的修改模式。与Stephane等人[44]类似，Fluri等人[45]利用层次聚类实现了修改模式的半自动挖掘。本文通过挖掘历史修改提交中的修改模式，辅助当前修改的影响分析。

[①] ROSE: http://www.st.cs.uni-sb.de/softevo/

1.2.3 代码修改完成周期预测

开发人员在完成代码修改后需要经过第三方审查人员的检查，审查人员根据代码修改情况提出修复建议，以便在代码集成之前识别和修复缺陷。很多研究人员发现，代码审查环节中的许多因素会对代码修改的完成周期产生影响。Patanamon等人[46]研究发现查找合适的代码审查人员是代码修改任务中的关键步骤，不合适的审查人员将严重提高代码维护工作的成本。同时，Patanamon等人[46]还提出一种代码审查人员的推荐方法，该方法根据修改代码的文件路径的相似性来推荐合适的代码审查人员，位于相似文件路径中的文件将由类似的代码审查人员进行管理和审查。与Patanamon等人[46]类似，Oleksii等人[47]的实证研究表明开发人员和审查人员的个人因素会很大程度影响代码修改提交的审查通过率。Baysal等人[48]研究发现诸多非技术因素会影响代码修改的审查通过率，包括：代码修改量，修改提交时间，修改需求的优先次序，代码修改人员和代码审查人员等。他们的方法从WebKit[①] 项目的问题跟踪和代码审查系统中提取信息，验证各因素的重要性。本文使用机器学习算法从开源代码审查软件中提取审查信息，根据代码审查中各元素预测代码修改的完成周期。由于没有与代码审查周期预测直接相关的工作，接下来主要介绍机器学习在软件维护研究中的应用。

随着机器学习算法的飞速发展，研究人员已经将机器学习应用于软件维护领域的各个研究工作中。Murali等人[49]提出一个贝叶斯框架，从代码语料库中学习程序规范，使用这些规范检测可能存在缺陷的程序行为。该方法主要的观点是将语料库中的所有规范与实现这些规范的程序语法相关联。Mills等人[50]通过二分类方法实现可追踪性链接恢复，能够自动将所有潜在链接集合中的每个链接分类为有效或者无效。Rath等人[51]利用修改提交的相关信息训练分类器，以识别修改提交所针对的修改问题。Karim等人[52] 提出从软件度量指标中提取特征并使用支持向量机和随机森林建立模型来预测软件故障的方法，他们将软件度量指标划分为静态代码度量指标和过程度量，从静态代码度量指标中提取代码行数，循环复杂度以及对象耦合等特征；从过程度量中提取源代码历史变化等特征。Shimonaka等人[53]提出利用机器学习方法从源代码中识别自动生成的代码，该方法认为通过朴素贝叶斯和支持向量机模型从源代码中学习代码的

[①] WebKit, https://webkit.org/

语法信息，可以预测代码是否为自动生成。Nguyen等人[54]提出一种自动映射不同编程语言之间API的方法，该方法从不同编程语言的原代码库中学习API的关联关系。Mario 等人[55]从项目源代码以来的API中提取特征，使用机器学习算法实现对软件项目的自动分类。

1.2.4 软件存储库挖掘

软件工程中对软件存储库的挖掘已经有很长的历史，开发人员通过软件存储库数据可以理解软件项目的演化历史，从而更好的完成软件维护和更新工作。软件存储库挖掘的一个方向是对源代码的挖掘。Michail等人[56]运用数据挖掘技术检测在不同程序中如何复用代码。了解代码的复用模式可以有效减少开发人员工作量。Lin等人[57]利用频繁挖掘技术在源代码中提取编程规则，他们的研究表明，违反编程规则的代码可能存在缺陷。Holmes等人[58]通过挖掘源代码库中代码的结构上下文，问开发人员展示相关API的用法。类似地，Bruch 等人[59]提出从代码库中学习从而提升IDE中代码的补全效果。

软件存储库挖掘的另一个研究方向是从版本控制系统中挖掘历史修改信息。Uddin等人[60]通过挖掘客户端的历史修改数据，研究客户端的演变过程与API 的使用关系，能根据进度为开发人员推荐API。Canfora等人[61, 62]和Thummalapenta等人[63]通过版本控制器定期提取软件项目状态，用来研究软件演化过程中各软件实体（如代码行，缺陷，代码克隆等）的演变规则。Zaidman等人[64]通过挖掘软件演化数据，研究工程代码和测试代码在软件演化中如何共同发展。Nguyen 等人[65]从Java 项目的修改历史中提取方法级别的修改规则，并研究这些规则在项目内和跨项目的可复用性。与Nguyen等人[65]类似，本文从多个开源项目的历史修改提交中，挖掘与当前修改最相似的提交，提取相似提交的修改规则，辅助当前修改的影响分析。

1.3 本文的研究内容与主要贡献点

本文依托海量的代码修改提交数据以及代码修改审查数据，结合机器学习方法，针对代码修改影响分析以及代码修改完成周期进行深入研究。本文的主要研究内容如下所述：

(1) 针对代码修改影响分析，本文提出一种基于历史修改模式的影响分析辅

助方法。本文从开源项目中获取代码提交历史数据构建提交语料库，通过计算当前修改的自然语言描述与提交注释信息以之间以及当前修改前后代码与提交中修改前后代码之间的相似度匹配最相似的提交。通过关键类判定方法判定提交中的关键类，将关键类等价为当前修改类，以关键类在提交中修改模式指导当前修改的影响分析。最后通过传统影响分析方法获取当前修改的初始影响集，结合18种程序实体间的耦合关系，分析提交中关键类与其他类的耦合关系以及当前修改类与初始影响集中其他类的耦合关系，根据耦合关系的相似度将关键类的修改模式映射回当前修改，对初始影响集中的类进行重排序得到最终修改集。最后，通过实验对比初始影响集与最终影响集的影响分析效果。

(2) 针对修改完成周期的预测，本文提出一种基于可判别性特征的修改完成周期预测方法。本文从开源代码审查软件中获取海量代码审查历史数据，提取可判别特征，以代码修改审查轮次预估代码修改完成周期，并结合机器学习方法建立代码完成周期预测的模型。从代码审查人员以及代码修改人员信息中提取非技术维度的影响特征，从提交注释文本以及修改前后代码中提取技术维度的影响特征，结合多维度的特征训练机器学习模型。

本文的主要创新点如下：

(1) 在修改影响分析方面，本文引入开源项目中的相似修改信息对传统影响分析方法的结果进行优化，效果得到提升。与其他挖掘项目历史修改信息的方法不同的是，本文从多个开源项目中基于修改内容相似度匹配历史提交，而传统方法仅从当前修改类的项目修改历史中检索与当前类共同修改的类集，不涉及具体修改内容。实验结果表明，本文提出的影响分析辅助方法，在多个开源项目上都能提升传统影响分析工具的精确率和召回率。

(2) 在修改完成周期预测方面，本文创新地提出通过挖掘代码审查信息中的可判别特征，预测代码修改周期的方法。实验结果表明，代码审查信息中存在大量影响代码修改周期的因素，本文基于这些因素，结合机器学习模型，完成了对代码修改周期较好的预估结果。

1.4 本文的论文结构与章节安排

本文共分为五章，章节内容安排如下：

第一章主要阐述了代码修改影响分析和代码修改完成周期预测在软件维护过程中的重要性，概述了代码修改影响分析、代码修改模式挖掘、代码修改周期预测以及软件存储库挖掘的国内外研究现状，以及介绍了本文的研究内容和主要贡献点。

第二章提出了基于历史修改模式的影响分析辅助方法。该章主要介绍代码修改提交语料库的构建、相似提交的筛选、历史修改模式的映射以及影响集的优化等。

第三章提出了基于可判别性特征的修改周期预测方法。该章主要介绍代码审查数据的处理、特征的提取、算法选择以及模型优化和评估等。

第四章的主要内容是介绍本文提出的修改影响分析辅助方法和代码修改周期预测方法的系统设计和实现，分别介绍了系统实现方法、主要功能以及系统展示。

第五章对本文的工作进行总结。主要总结本文提出的方法，并分析这些方法存在的不足与局限，最后，指出在将来的研究工作中如何进行改进。

1.5 本章小结

本章首先介绍了代码修改影响分析和代码修改完成周期预测在软件维护过程中的重要性，并对与本文相关的技术和研究领域进行了概述，分别介绍了代码修改影响分析、代码修改模式挖掘、代码修改周期预测以及软件存储库挖掘的国内外研究现状。最后概述了本文的主要研究内容以及本文的贡献点，并对本文的章节安排作了简要的介绍。