课程报告：软件变更影响分析

姓名：王雨欣

学号：DZ1933029

课程：软件维护与演化导论

一、 背景介绍

1. 软件变更影响分析的重要性

在大多数软件生命周期中，软件维护是最昂贵、最困难的阶段[1]。据估计，在软件系统的整个生命周期中，软件维护工作往往超过总生命周期成本的 50%[2]。其中，软件维护最昂贵的两项是对问题或其他明示变更需求的理解，以及对变更后所造成影响的掌握。看起来很小的更改可能会在整个软件系统中引起连锁反应，从而在其他地方带来重大的意外影响。因此，软件开发人员需要一种机制来理解对软件系统的更改将如何影响系统的其余部分，这也即为对软件的变更影响分析（Change Impact Analysis，CIA）。

2. 软件变更影响分析的定义

在过去几十年中，很多研究学者对影响分析（impact analysis）这一术语进行了不同定义。Bohnner 和 Arnold 在 1996 年的工作[3]中，对该术语给出了明确定义：

> *Identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change.*（识别一个变更潜在的影响，或是评估为了实行一个变更，所需要进行的修改。）

这一定义被之后的研究工作者广泛接受。之后的研究工作也都基于该定义而展开。

3. 软件变更影响分析的研究意义

软件变更影响分析的研究意义，主要体现在以下五点：

1) 开发出自动化的分析技术节约了大量人工成本，减少了时间和金钱开销。
2) 能够帮助判断实施更改所需的工作量。
3) 提出应更改的软件工件。
4) 帮助确定测试用例以确保软件被正确地实施更改。
5) 模拟软件变更并分析其影响。部分对软件的变更存在高风险，它们可能会导致

意外的副作用，引入新的错误并导致更多的不稳定。

二、 主要研究内容

目前对软件变更的影响分析，主要在三个方面进行：源代码、形式模型和各种工件。其中，源代码分析方法可分为静态（static）和动态（dynamic）两种。形式模型可进一步分为架构模型和需求模型，以体现软件开发的早期阶段，比如需求捕获和架构推理。对各种工件的分析涵盖了广泛的文档和数据源，例如文档，错误跟踪器和配置文件。下图展示的是软件影响分析的研究范围[3]：

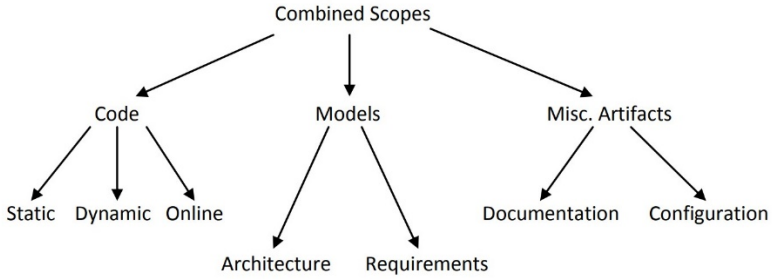

图 1：软件变更影响分析的研究范围

下面，将对源代码、形式模型和工件三个方面的变更影响分析内容做大致介绍。

1. 源代码

目前，有许多方法通过推理代码模块之间的继承关系，方法调用行为以及程序实体之间的其他依赖关系来研究变更的影响。对源代码文件，类包，类，方法，语句和变量进行分析，以预测更改的传播。

对源代码的分析可分为静态和动态两类。静态代码分析从源代码中提取事实，以构建用于评估更改影响的调用图，切片和其他表示形式，无需实际执行源代码。相反，动态方法对代码或编译的二进制文件进行检测，以收集有关方法执行的信息。

过去进行的研究表明，在某些情况下，基于静态分析的技术包括 90％以上的程序，通常会导致不精确的影响集[4]。

2. 形式模型

形式模型又可分为架构模型和需求模型。诸如 UML 组件图之类的架构模型可以在比源代码更抽象的级别上评估体系结构更改。这使得能够在开发的早期阶段和基于模型的开发（MBD）中进行影响分析，这在最近几年变得越来越重要。但是依赖于底层的建模语言，例如详细的 UML 类图，架构分析也可以实现细粒度的结果。粒度的典型级别是系统，子系统，组件和类。

形式化需求是软件开发过程中的重要部分，并且直到确定程序的最终版本前，会进行多次更改。如果需求是用 UML，GRL 或 UCM 等形式化建模语言编码的，那么由于它们结构良好的元模型，因此可以接受形式化的影响分析。

3. 各种工件

软件的变更除了发生在源代码或体系结构模型中外，文档、配置文件、错误跟踪器以及类似的软件辅助内容也可能会经常更改。对此类实体的更改也会影响软件，例如更改配置文件时。因此，在这样的实体（大多数是不同类型的文件）之间执行影响分析也已成为研究问题。

由于软件变更大多发生在源代码上，因此接下来将介绍源代码变更的影响分析技术。

三、 代码变更影响分析主要技术

目前，对源代码变更影响分析的技术有以下几种：

1. 调用图

当更改代码中的方法或者过程后，这种改变会直接或间接地去影响其他源代码实体，比如向现有类添加方法会影响整个程序虚拟方法的调用。因此，分析软件系统的调用行为可以帮助评估方法和过程更改的影响。调用图技术提取方法之间的调用信息，并将其用流程图的形式体现出来，以便静态分析源代码。这是一种静态的可追溯性分析技术。

下图所示为一个简单的调用图示例，展示了代码中五个函数 main，scanf，printf，add 和 inc 之间的调用关系。

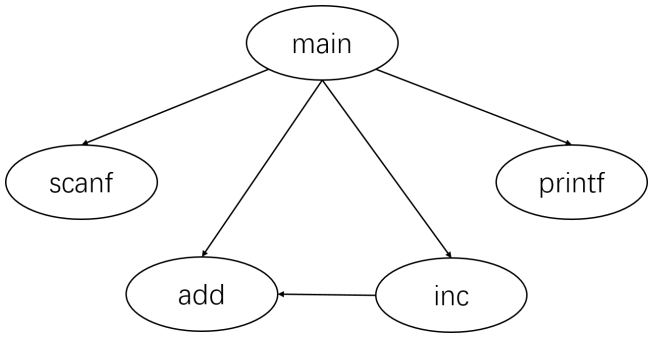

图 2：简单调用图示例

Ryder and Tip 曾提出了一种基于调用图的，针对面向对象编程语言进行变更影响分析的方法[4]。这种方法能够根据调用图分析来判断是哪项变更导致了测试失败。在 Ryder and Tip 提出的方法中，源代码的变更映射成一组原子变更（使用类、方法、域和它们的相互关系作为变更的原子单元），此外还确定了这些原子变更的偏序关系，这些变更间的偏序关系保证程序句法合法。给定一系列原子变更 A 和给定的测试驱动集合 T，执行程序的部分功能。通过此静态方法来确定：

- 测试驱动 T 中挑选被变更 A 影响子集 T’用于回归测试。
- 变更集合 A 的子集 A’可能影响 T 中的某些测试驱动 t。这允许开发者忽略不被测试 t 失败影响的变更。

- 变更 A 的子集可能不影响任何测试用例，这些变更可以立即纳入。
- 覆盖率信息可以作为创建新测试用例的基础。

调用图容易理解且容易构造，其基本原理是传递闭合。但当调用行为较为复杂时，调用图分析存在很大的不准确性，同时调用图也不会捕获由于程序返回而产生的影响传播。

2. 程序切片

程序切片技术分为静态切片和动态切片两种。静态切片技术是指在计算程序切片时使用的是静态的数据流和控制流分析方法。该技术对程序的输入不做任何假设，所做的分析完全以程序的静态信息为依据。静态技术的工作量较大，因为要分析程序所有可能的执行轨迹，所以相对于动态切片技术，静态切片技术一般用于程序理解与软件维护方面。一些知名的静态程序切片工作比如[5]和[6]。

动态切片技术使用的是动态的数据流和控制流分析方法，因此切片的计算过程依赖于程序的具体输入。采用这一技术，每一次的计算工作量较小，但每一次的计算都不尽相同，因此动态切片技术多用于程序调试、测试方面。

3. 执行跟踪

与静态调用图相反，动态执行跟踪仅包含那些在程序执行期间被调用的方法。 与调用图类似，它们允许通过分析在更改的方法之后调用了哪些方法来评估方法更改的影响，因此也可能受到影响。 建立了依靠动态执行数据的方法来克服静态切片（昂贵）和调用图（不精确）的局限性。

经典的执行跟踪技术比如 PathImpact[7]和 CoverageImpact[8]。在 PathImpact 中，提出了一种基于 whole path profiling 的动态变更分析技术。该技术分析程序结构并建立控制调用图，如果程序 P 发生了变化，所有在 P 之后调用的程序以及所有在 P 返回之后仍然处于堆栈中的程序，即被认为可能受到 P 的变化影响。该方法利用静态切片和自动插桩技术动态获取程序运行时的数据，并收集这些数据，最后从这些数据中找到可能受到影响的部分。

4. 显示规则

通过引入专家知识，可制定严格的变更影响规则。这些规则确定当某个实体发生更改时，哪些其他实体必须更改。例如，如果某接口被更改，也必须更改实现该接口的所有类。

相关工作[9]目的即是通过软件变更管理框架为开发人员提供支持，该框架已集成到软件工程环境中，通过进行影响分析，来进行正向工程和再工程。其所提出的影响分析方法通过使用变更管理功能来扩展软件工程环境的内核，从而将自动化技术与有指导的用户干预相结合。分析过程大致如下：首先，该方法从程序中提取出依赖图，然后应用一组传播规则来确定受影响的实体。然后将获得的可能受影响的实体集呈现给开发人员，然后由开发人员决定实施哪一个。

5. 信息检索

信息检索（IR）是近年来影响分析中热门的技术路线之一。其通过利用自然语言相关技术，在不同文档中搜索相似术语，以推断它们之间的关系。软件系统中的模块（或类）以多种方式关联，其中最明显且探索最多的一种关系是基于数据和控件的依赖关系。但是，这些类也可以在概念上（或在文本上）相关，它们可能有助于实现类似的域概念。

给定文本更改请求（例如，错误报告），使用潜在语义索引建立索引的源代码的单个快照，用于估计影响集，并从源代码中导出概念上的耦合。

6. 概率模型

马尔可夫链和贝叶斯信念网络（BBN）等概率模型可以基于经过深入研究的数学模型和定理，对变化的传播进行建模，从而计算相关实体受更改影响的可能性。

7. 历史挖掘

历史挖掘是一种可能性分析技术，它从经常一起更改的软件存储库中挖掘集群或实体模式，因为对集群中一个实体的更改也可能会影响该集群中的所有其他实体。一些相关工作比如[10]和[11]。

关于静态和动态变更影响分析的区别，总结来说，静态变更影响分析技术通常是通过分析程序（或其更改历史存储库）的语法和语义或演化相关性来执行的。结果影响集通常有很多假正例，其许多要素并未真正受到影响。静态分析可进一步分为结构静态分析，文本分析和历史分析。结构静态分析即分析程序的结构依存关系并构建依存关系图；文本分析基于对源代码中的注释和/或标识符的分析，提取了一些概念上的依赖性（耦合），这些耦合措施为传统的结构耦合措施提供了新的视角；历史分析是通过从软件存储库中的软件的多个演化版本中提取信息来执行的。

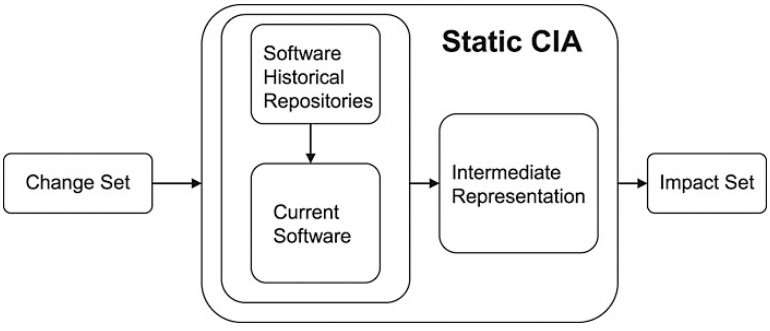

图 3:静态变更影响分析的大致流程

而动态分析考虑一些特定的输入，并依靠对程序执行过程中收集的信息（例如执行跟

踪信息，覆盖范围信息和执行关系信息）的分析来计算影响集。动态变更影响分析技术的过程如图所示在图 4 中，它以一组测试数据作为输入开始，并收集执行信息以进行分析。与静态分析相比，影响集倾向于更精确。但是，动态变更影响分析技术的成本比静态变更影响分析的成本高，这是因为在程序执行过程中进行了昂贵的依赖分析，而且其影响集通常还包含一些假阴性。动态变更影响分析可以在线或离线执行。离线变更影响分析在程序完成执行之后执行，而在线变更影响分析使用程序执行时收集的信息执行所有分析。

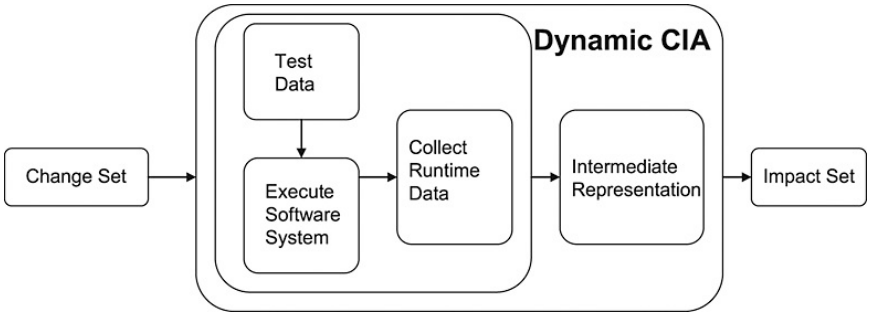

图 4: 动态变更影响分析的大致流程

四、 报告总结

本文对软件变更影响分析的研究主题进行了较为综述性的介绍。首先介绍了变更影响分析的研究背景，包括该研究的重要性、相关定义和研究意义；然后对变更影响分析的主要研究内容进行了介绍；之后对最为关注的源代码变更影响分析的相关技术分类进行了介绍。

通过本次的个人课程报告，本人学习到许多未曾接触过的专业知识，对不熟悉的研究领域多了一些认知。此报告大部分内容是对软件变更影响分析的相关方法技术进行了综述性介绍，希望以此为知识面基础，之后在专业层次上的进行更深度的技术学习。

五、 参考文献

1. Wei Li and Sallie Henry, “An Empirical Study of Maintenance Activities in Two Object Oriented Systems,” Journal of Software Maintenance, Research and Practice, Volume 7, No. 2 March-April 1995, pp.131-147.
2. M. Lee, A. J. Offutt, and R. T. Alexander, “Algorithmic analysis of the impacts of changes to object-oriented software,” in Proceedings of the 34th International Conference on Technology of Object- Oriented Languages and Systems (TOOLS 34), Santa Barbara, CA, USA, July 2000, pp. 61–70.
3. S. A. Bohner and R. S. Arnold, “Software Change Impact Analysis”. Los Alamitos, CA, USA: IEEE Computer Society Publications Tutorial Series, 1996.
4. B. Ryder and F. Tip, “Change impact analysis for object-oriented programs,” in Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering (PASTE ’01), Snowbird, Utah, USA, June 2001, pp. 46–53.
5. D. Binkley and M. Harman. “A large-scale empirical study of forward and backward static slice size and context sensitivity”, in Proceedings of the International Conference on Software Maintenance (ICSM), pages 44–53, 2003.
6. D. Binkley, M. Harman and J. Krinke, “Empirical study of optimization techniques for massive slicing”, in ACM Transactions on Programming Languages and Systems (TOPLAS), 30(1), pages 3:1–3:33, 2007.
7. J. Law and G. Rothermel, “Whole program path-based dynamic impact analysis”, in Proceedings of the International Conference on Software Engineering (ICSE), pages 308–318, 2003.
8. A. Orso, T. Apiwattanapong and M. J, “Harrold. Leveraging field data for impact analysis and regression testing”, in Proceedings of the Symposium on Foundations of Software Engineering (ESEC/FSE), pages 128–137, 2003.
9. J. Han, “Supporting impact analysis and change propagation in software engineering environments,” Monash University, Peninsula School of Computing & Information Technology, McMahons Road, Frankston, Victoria 3199, Australia, Tech. Rep. 96-09, October 1996.
10. B. Fluri, H. C. Gall, and M. Pinzger, “Fine-grained analysis of change couplings,” in Proceeding of the Fifth IEEE International Workshop on Source Code Analysis and Manipulation 2005, November 2005, pp. 66–74.
11. B. Fluri and H. C. Gall, “Classifying change types for qualifying change couplings,” in Proceeding of the 14th IEEE International Conference on Program Comprehension (ICPC 2006), Athens, 2006, pp. 35–45.