信 息 科 学 与 技 术 学 院

毕 业 设 计 开 题 报 告

Java 反射机制研究与应用

学　　号: 20112799

姓　　名: 林从羽

班　　级: 2011 级软件 5 班

专　　业: 软件工程

指导老师: 胡晓鹏

职　　称: 副教授

2015 年 03 月 31 日

开题报告的内容应包括

（1） 本课题的目的、意义。

（2） 本课题国内外研究现状（国内外文献综述，给出参考文献）。

（3） 本课题设计任务与要求。

（4） 拟采取的技术路线与试验方案。

（5） 预期成果（包括预期能够完成的设计或者理论研究成果，拟提交的软件、硬件、仿真程序等）。

（6） 设计进度安排。

注：（1）开题报告工作由毕业设计指导小组组织实施，目的是帮助学生执行选题论证。

（2）毕业设计指导小组的论证意见以“通过”、“不通过”结论。通过者按计划开展论文工作。不通过者，整改后重新提交开题报告，如果仍然不通过则取消毕业设计资格。

1. 本课题的目的、意义

反射(reflection)是指一个运行期程序具有检查自身信息和外部环境的信息，并基于这些信息来改变自己行为方式的能力[12]。Java 是纯面向对象的语言，系统中存在的所有一切都是对象，Java 程序在执行之前需要被编译成平台无关的字节码。Java 虚拟机希望尽可能多的类型检查能在程序运行之前完成，换句话说，编译器应当在编译期间尽最大努力完成可能的类型检查[14]。这体现了静态语言强类型的优势，同时也限制了程序的动态性：所有的类型信息都必须在编译时确定，系统无法对运行时发现的类型进行动态的判断处理。对此，Java 引入了很多特性来加强对动态性的支持，比如 transient 关键字、泛型、注解等，但并不成完整的体系。Java 对动态性真正完整且自成体系的支持，是它的反射机制。

反射的概念自 1982 年被提出以后经过了不断发展，如今已经具备比较完备的体系，也在很多编程语言上有了不同的实现。相比之下，目前国内对于 Java 反射机制的研究仍然停留在使用方法的层面上，很少有深入到底层和源码来剖析 Java 反射机制的，从体系和架构上来研究的则更加稀少。反射更多作为软件开发的“黑箱子”，为人熟知但少为人所深入理解。但正如侯捷先生所言，“练从难处练，用从易处用”[1]，我们在学习一项技术的时候，不仅要学习它的使用方法，还要掌握它本质的原理，达到举一反三的理解。学习原理不仅不是浪费应用开发的时间，相反，“熟知技术应用背后的原理，能使人以少胜多，进而降低学习新事物的成本，摆脱面对前沿技术的被动态度”[2]。本论文正是尝试以目前使用最广的 Oracle 虚拟机 Java HotSpot 为例，从源码上对 Java 的反射机制进行研究，并根据现有的反射理论框架总结提炼 Java 反射机制的架构和体系。另外，基于 HotSpot 对反射的具体实现，本文针对软件开发对通用性和拓展性日益增长的需求提出了应对方法，即在元数据层面上来拓展语言对应用的支持。

Java 反射机制有很多应用，特别是在框架开发的过程中，Java 反射机制有其重要的地位。软件的开发框架多是为了解决某一方面或领域的特定问题提出的解决方案，使用框架的开发者往往需求各异，这就要求框架要有很高的通用性和拓展性。通用性方面，Java 团队在 Java1.5 的发布中引入的泛型已经提供了优雅的解决方案。拓展性方面的需求则更加复杂，有时我们需要根据系统运行时发现的字符串或字段来生成相应的类、对象或代码模板，这种能力是类似 C++这种传统面向对象语言所无法提供的。反射机制正是在元层级别为提高应用软件的高拓展性提供的解决方案，它包含注解、字段反射、方法反射等特性。很多开源框架都使用了反射来达到其自身的高拓展性，如 Spring 中的 BeanFactory，它根据 xml 配置和注解来动态注入依赖；Spring MVC 中的 DispatcherServlet，它根据@Controller、@Service、@RequestMapping

[1] http://jjhou.boolan.com，侯捷官网。
[2] http://blog.csdn.net/peopleware/article/details/19798, 刘天北，《人月神话》一书译者。

等注解来分发、调用用户的 servlet，放松了对用户的限制，如用户必须继承 HttpRequestServlet 这个类，必须使用 doGet、doPost 的方法命名等；Junit4.x 框架中利用@Test 注解来标注并用以发现用户所有的测试方法，避免了 3.x 框架中对用户不必要的限制，如测试方法命名必须以 test 开头，等。

本论文的独特工作，一方面在于采用了研究源码的手段，真实地还原了 Java 反射机制底层的实现原理；另一方面在于通过源码的微观视角和体系架构的宏观视角，全面研究了 Java 的反射架构所对外提供的功能接口（也称元对象协议），并以此出发定性定量分析了 Java 反射机制的优点和缺点。无论是分析源码的方法，还是对 Java 反射机制在元对象协议中具体地位的分析，在目前都是独一无二的工作。

本论文的意义，一方面在于通过对 Java 反射机制的源码剖析和体系架构的提炼，为读者提供 Java 反射从具体实现到体系架构的整体图景，使读者对反射这个工具既知其然，又知其所以然。正所谓“源码之前，了无秘密”[3]；另一方面在于通过现今软件开发的发展趋势来理解反射的意义和发展，并通过反射的视角提出应对未来对软件更高拓展性要求的方案。

2. 本课题国内外研究现状（国内外文献综述，给出参考文献）

反射这个概念是 1982 年 B.C.Smith 博士在其博士论文[1]中首次提出的。文中提到一个具有反射能力的系统应该具备以下几个基本特性：一是系统必须持有一份关于自身的信息描述；二是系统与其描述之间必须能方便地进行连接交互；三是系统必须隔离、安全地执行反射任务。

1984 年，Daniel P.Friedman 和 Mitchell Wand 在其论文[2]中针对 B.C.Smith 博士提到的连接和安全问题提出分层观点，并给出了在 CLOS 系统下对反射的具体实现。这个元层和基本层分离的方案，引出了在元层“开放实现”的观念，也成为很多语言和系统实现反射时采用的模型，如 SOM、SmallTalk、CLOS、Java 等。

1987 年，P.Maes 在其论文[3]和[4]中对反射的关键内容进行了归纳和总结，提出了反射的理论框架和实现事宜，标志着对反射理论和实现进行深入研究的开始。

1989 年，Ira R.Forman 和 Scott H.Danforth 在[5]中描述了他们在 IBM 的 SOM ToolKit3.0 中实现的一个反射模型，以及构建一个反射系统会涉及的静态层面和动态层面。静态层面涉及存储数据的数据表、方法表、继承体系和元类循环等问题，动态层面涉及了方法拦截(intercession)、动态调用(dynamic invocation)、方法决议次序(method resolution order)、元类继承约束(inheritance of metaclass constraints)等。此外，I.R.Forman 和 S.H.Danforth 还发表了关于这个模型的一些论文，[6]是一个概述，[7]中描述了 SOM 中的元类继承的问题，[8]中分析并解决了元类系统构建中的元类继承约束问题，[9]中描述了元类中的 Before/After 元类的组合和应用。

[3] http://jjhou.boolan.com，侯捷官网。

1991 年，George Kiczales 在[10]中描述了 CLOS 系统的元对象协议(metaobject protocol)的设计和实现。

1996 年，Stanley B.Lippman 出版著作[11]介绍了 C++的对象模型，虽然 C++仅有 RTTI 的能力，严格而言并非反射，但这个对象模型是研究基于类和对象的反射模型的基础。

Ira R.Forman 的[12]是首部详细介绍 Java 反射的书籍，其涉及 Java 反射的各个方面，包括使用场景、字段（基本类型、数组类型和接口类型）反射、方法反射、动态调用、异常链信息反射、类加载机制、设计模式、性能分析以及反射的发展与未来等。

Oracle 公司有三份与反射内容相关的官方文档：[13]对 Java 语言的语义进行了细致的说明；[14]对 class 文件和虚拟机的架构和指令集做了规格性的说明，是 Java 反射所采用数据模型的基本架构；[15]对 java 的本地接口(native interface)的规格和使用做出了详细的说明。

国内方面值得注意的有[16]和[17]两篇论文的研究。前者提出可以通过在系统设计阶段区分系统的功能性需求和非功能性需求的方法来设计具有反射能力的系统，并对元层技术、元对象协议、反射理论和反射的分类等做出了归纳和总结；后者提出了一个分布式元对象协议的实现，并指出在元层开放实现与在基本层封装变化的思想一脉相承。

[18]介绍了虚拟机的基本架构和运行调优，[19]中描述了 HotSpot JVM 的架构，介绍了如何在虚拟机层级表示 java 层级的类和对象，如何组织数据等。

[20]是对 JVM 全面细致的剖析，包括语言处理器、即时编译器等。侯捷在 2004 年发表的[21]也是一篇对 Java 反射机制应用的概述性文章。

参考文献：

[1] Brian C.Smith. “Reflection and Semantics in a Procedural Language”[J], Ph.D. thesis, Massachusetts Institute of Technology, LCS TR-272, 1982.

[2] Daniel P.Friedman. “Reification: Reflection without Metaphysics”[J], *Conference Record of the 1984 ACM Symposium on LISP and Functional Programming*, 1984, 348-355.

[3] Pattie Maes. “Issues in Computational Reflection” [M], Meta-Level Elsevier Science,

[4] Pattie Maes. “Concepts and Experiments in Computational Reflection”[J], *OOPSLA’87 Conference Proceedings*, October 1987, 147-155.

[5] Ira R.Forman, Scott H.Danforth. “Putting Metaclasses to Work” [M], Addison-Wesley, 1998.

[6] Danforth, S.H., I.R.Forman. “Reflections on Metaclass Programming in SOM”[J], *OOPSLA’94 Conference Proceedings*, October 1994, 440-452.

[7] Danforth, S.H., I.R.Forman. “Derived Metaclasses in SOM”[J], *Proceedings of the 1994 Conference on Technology of Object-Oriented Languages and Systems*, April 1994.

[8] Forman, I.R., S.H.Danforth, “Inheritance of Metaclass Constraints in SOM”[J], *Reflection’96*, 1996.

[9] Forman, I.R., S.H.Danforth, H.H.Madduri. “Composition of Before/After Metaclasses in SOM”[J], *OOPSLA’94 Conference Proceedings*, October 1994, 427-439.
[10] Kiczales, G., J. des Rivieres, D.G.Bobrow. “The Art of Metaobject Protocol” [M], The MIT Press, 1991.
[11] Stanley B.Lippman. “Inside the C++ Object Model”[M], Addison-Wesley, 1996.
[12] Ira R.Forman, Nate Forman. “Java Reflection in Action” [M], Manning Publications Co., 2005.
[13] Gosling, J., B. Joy, G. Steele, G. Bracha, A. Buckley. “The Java® Language Specification” [M] (SE7 ed), Oracle America, Inc., 2013.
[14] Lindholm T., F. Yellin, G. Bracha, A. Buckley. “The Java® Virtual Machine Specification” [M] (SE7 ed), Oracle America, Inc., 2013.
[15] Sheng Liang. “The Java™ Native Interface: Programmer’s Guide and Specification”[M], Addison-Wesley, 1999.
[16] 曹玲. 基于元层技术的反射分析方法研究[D]. 硕士论文. 河海大学, 2003 年 3 月.
[17] 李冰. 反射的分布式元对象协议（MOP）的设计与实现[D]. 硕士论文. 天津大学, 2004 年 1 月.
[18] 周志明. “深入理解 Java 虚拟机：JVM 高级特性与最佳实践”[M], 机械工业出版社. 2011 年.
[19] 陈涛. “HotSpot 实战”[M]. 人民邮电出版社. 2014 年 3 月.
[20] 莫枢. “Java Program in Action”. http://elastos.org/elorg_files/FreeBooks/java/JVM%E5%88%86%E4%BA%AB20100621.pdf . 2010 年.
[21] 侯捷. “Java 反射机制”[DB/OL]. http://jjhou.boolan.com/javatwo-2004-reflection.pdf. 2004 年.

3. 本课题设计任务与要求

本论文设计主要分五个部分：

第一部分，背景和现状。

第二部分，反射模型的基本理论和体系。本部分主要展开 Brian C.Smith 博士提到的反射系统的三部分，包括自我描述(representation)，元层和基本层的连接(casual connection)和安全性(security)三个方面。数据的自我描述部分将讲述一个一般的数据描述模型，以及 Java 具体的数据模型：class 文件；元层和基本层的连接部分将涉及一个元对象协议的基本内容；安全性部分将描述 Java 的类加载器机制、SecurityManager 执行的安全检查等。

第三部分，机制研究。本部分以目前较为普及的 Oracle JVM: HotSpot 的源码来研究 Java 反射对论文第二部分所提之反射模型的实现情况，并从源码中还原出 HotSpot 虚拟机实现的

反射模型和体系。然后，与其他语言和模型（如 Smalltalk, CLOS, SOM, COM 等）所实现的反射机制进行简要的技术层面的对比，并分析其他不支持（如 C/C++, Pascal）或不需要反射（如动态语言）特性的语言对软件开发的支持，从开发生产的眼光来看软件开发对反射的需求。

第四部分，应用研究。本部分将结合实际的 Java 开源框架架构和源码研究 Java 反射的实际应用，会涉及设计模式、AOP(面向切面编程)、IoC(控制反转)等内容。另外，从性能、拓展性、代码清晰度等各个方面研究 Java 反射的应用场合，提出不足部分的改进方案。

第五部分，改进和验证。

第六部分，总结。

4. 拟采取的技术路线与试验方案

本论文技术重点主要在有效分析 JVM HotSpot 源码上，因此搭建虚拟机运行调试环境很有必要。本文拟在 Linux 系操作系统 Ubuntu 14.10 版本上编译 openjdk，借助 HSDB 工具和 Java 自带的 Visual VM 1.3.8 来抓取 HotSpot 运行时的数据快照和可视化内存布局。

5. 预期成果（包括预期能够完成的设计或者理论研究成果，拟提交的软件、硬件、仿真程序等）

英文文献 1 万字翻译稿 1 份。

毕业设计 Java 反射机制研究与应用论文 1 篇。

《本科毕业设计（论文）指导纪要》1 份。

6. 设计进度安排

周次	时间	任务
第1周	2015-03-09－2015-03-15	仔细阅读毕业设计任务书，制定本学期毕业设计工作计划。
第2周－第3周	2015-03-16－2015-03-29	明确课题研究现状，确定翻译篇目，完成开题报告。
第4周－第5周	2015-03-30－2015-04-12	文献阅读，完成1万字论文翻译。
第6周	2015-04-13—2015-04-19	校对并修改外文翻译。
第7周－第9周	2015-04-20－2015-05-10	按毕业设计任务书和开题报告制定计划进行研究。
第10周	2015-05-11－2015-05-17	准备中期答辩资料，包括论文中期报告，指导纪要等。
第11周－第15周	2015-05-18－2015-06-16	完成研究及所有成果。学术检测，校级抽样答辩。
第15周	2015-06-17－2015-06-21	校级论文抽样答辩，毕业设计论文全面答辩。
第16周	2015-06-23－2015-06-26	提交研究成果，指导纪要，外文翻译，答辩记录表。论文存档。

指导教师意见

指导教师签字：_____

年　　月　　日

毕业设计指导小组意见

（签字）_____

_____

_____

_____

年　　月　　日

注 1、以上各项空格不够可以续页。

2、此表作为附件装入毕业设计（论文）资料袋存档。