博士研磨

一个博士研究生的回忆录 [1]

原著　Philip J. Guo　(philip@pgbovine.net)
翻译　齐鹏　(pengqi@cs.stanford.edu)

[1]英文原文链接:http://www.pgbovine.net/PhD-memoir/pguo-PhD-grind.pdf。 本翻译基于原文2012年7月16日发布的版本译成。

献给所有热爱创造的人。

目录

Disclaimer

This is an unauthorized Chinese translation of Philip J. Guo's memoir The Ph.D. Grind, and the original author did not have any input on the translation.

Apart from the Translator's Preface, none of the content of this work represents, none should be interpreted as, the opinion of the translator; nor will the translator be responsible for the consequences of any interpretation of this translation.

声明

本作品是Philip J. Guo的回忆录《The Ph.D. Grind》的中文翻译，翻译并没有得到原作者的任何授权，原作者也并没有以任何形式参与到翻译过程中。

除译序外，本作品的一切内容均不代表——且不应被认为是——译者的观点；译者对此翻译的任何理解以及其导致的影响亦不负有任何责任。

序

这本书记述了从2006年到2012年，我在斯坦福大学攻读博士研究生期间六年的求学经历。这本书适合广泛的读者群，其中包括：

- 有志攻读博士研究生[1]的本科生；
- 寻求方向或灵感的在读博士生；
- 希望更深入了解博士研究生的教授；
- 希望聘用和管理拥有博士学位员工的雇主；
- 在充满竞争的创新领域工作、与自我追求和自我激励密不可分的专业人士；
- 对学术研究充满好奇的有一定教育背景的成年人（或者早熟的青少年）。

《博士研磨》与已有的与博士经历相关的文章在写作形式、写作时机和写作基调上都有所不同：

[1]研究生（graduate student）本是指本科之后的高等教育，通常包括硕士研究生和博士研究生两种学位。现代汉语中“研究生”一词经常被滥用，用以单指硕士研究生，这其实是错误的。——译者注

形式 《博士研磨》是一本面向大众的回忆录，而非一本面向在读博士生的“成功指南”。尽管博士生也能在我的经历中学习到经验和教训，但我的目标并不是直接提供建议。对于博士生而言，市面上的“成功指南”和“建议专栏”已经不胜枚举，我也无意画蛇添足。这些文章充满了“持之以恒”和“不积跬步，无以至千里”等等空泛的词汇，但相反，回忆录的形式让我能丰富、具体地叙述发生在我自己身上的故事。

时机 《博士研磨》是我在完成博士学位之后马上着手写作的，而这正是撰写这样一本回忆录的最佳时机。不同于在读博士生的是，我可以在回忆录中对整个博士求学期间的工作进行系统的整理和反思；而相比资历较深的研究者而言，我可能更容易忠于攻读期间的经历，不会引入过多由研究经历带来的有选择性的观点和感受。

基调 尽管保持完全客观是不可能的，但我在写作《博士研磨》过程中仍然试图贯穿一种相对客观的基调。与其他作者不同的是，很多撰写博士相关文章、书籍，或者绘制漫画的人通常属于以下两类之一：

- 成功的教授或者科学家，他们通常给出一些冠冕堂皇的建议，比如他们可能会说：“研究生生活诚然辛苦，但它同时应该是一段美好的知识之旅，你应该享受这个过程、并从中受益.....因为我当年就是这样做的！”
- 或者是苦涩的博士研究生或者辍学博士，他们常常因为自己的经历留下了心理阴影，当提及博士生活的时候会用一种过分夸张、“看破一切”、自我怨恨的腔调：“啊，那时我的世界就是个活生生的地狱，我到底拿我的青春换了什么？！？”

冠冕堂皇的建议可能能激励一些学生，而大倒苦水的呻吟可能能引起另一些处境不佳的学生的共鸣，但作为大众读者而言，他们可能并不能感受到这些极端的情绪。

最后，在我开始讲述自己的故事之前，我希望强调一下，每个博士研究生的经历与他/她所在的学校、院系、研究领域、经费状况等都有巨大的关系。在我的读博生涯中，我感到自己非常幸运，能很大程度上自由自愿地完成自己的学业；我知道很多学生相比而言受到了很多的限制。我的故事只是一个孤立的数据点，所以我所呈现的故事可能并不能泛化、推广到每一个人。然而，我会尽力避免叙述变得过分局限于我个人的情况。

祝阅读愉快！

Philip Guo, 2012年6月

译者序

巫术与火刑

研磨的故事

献给所有热爱创造的人

前言

因为我本科的专业是电子工程和计算机科学，大部分本科同学在毕业之后，都马上投入到了工程性的工作中了。而我最终选择攻读博士学位，究其原因，一方面是受到来自父母潜移默化的影响，另一方面也和我在本科期间对工程性的实习产生的不良印象不无关系。

我父母从未要求过我去攻读博士学位，但我可以看出，终身大学教授[2]是他们最尊重的职业——而博士学位正是成为终身教授的必要条件。为什么终身教授会是他们心目中的理想职业呢？这其实并不是因为他们对纯粹的学术追求有什么不切实际的盲目推崇。尽管我的父母也都到过良好的教育，但他们同时也是非常现实的移民——一份终身教授的职位对他们而言，更大的吸引力在于终身制所带来的工作保障。

我父母的很多朋友都是在企业的工程性职位上供职的中国移民。由于他们在英语技巧和美国文化了解上的不足，他们中的大多数人在职业生涯中的经历都并不顺利，而这一问题往往随着年龄增长而更加突出。在假日聚会上，我经常能听到一个不变的主题：人们的工作因为难以相处的经理而处处不顺、成为年龄歧视和“玻璃天花板”效应[3]的牺牲品、甚至面临大

[2]在美国和加拿大等国家，终身职位（tenure）是指资深学者拥有的，除因正当理由外，免于被解雇的合同权利。与此不同的是任意性职位（at-will），规定劳动双方随时可以以任何理由终止劳动合同，不需要法定的正当理由，也不需要提前通知。大多数非学术研究类（工程性）的工作属于后者。——译者注

[3]玻璃天花板（Glass Ceiling）效应是一个政治术语，用于形容“在企业内升职过程中看

规模裁员和长期失业的风险。尽管我父亲不是一位工程师，而是在高技术产业就职，但他也难逃这种魔咒，在一系列和管理层极其官僚做派的斗争中失败，最终早早地结束了他在公司的工作。那时他还相对很年轻，只有45岁。

我的母亲则是这种不幸潮流中唯一的幸存者。她十分热爱自己在UCLA[4]作为社会学终身教授的工作。和她的大部分中国移民朋友不同的是，她享受终身职务保障，不用向老板汇报工作，可以几乎完全自由地追求她自己的学术兴趣，在她的研究领域也小有名气。亲眼目睹我母亲成功的职业轨迹，和父亲及他们的朋友们在职业生涯上的恶性循环两者之间的巨大反差，这在我高中和大学本科的学习生涯中留下了难以磨灭的印象。

当然，仅仅因为这种非理性的、年少时的恐惧就去选择读博显然是不明智的。为了让自己对在企业的工作生活有所印象，我在本科的每个假期都参加了工程性公司的实习。因为我工作过的办公室都碰巧只有我一个实习生，我被赋予了一种罕有的特权—— 我的工作职责是按照全职初级工程师的标准分配的。尽管在这个过程中我学到了很多技术上的技巧，我仍然觉得这种日复一日的工作极少需要思考且非常无聊；这也可能与我实习过的公司不是一流公司有关。我本科的很多朋友都在微软（Microsoft）和谷歌（Google）等一流公司实习过，并非常喜欢他们的实习经历，他们最终也往往在毕业后和这些公司签下了全职工作合同。

因为我对我的实习经历感到厌倦，而另一方面对本科时作教学助理（助教）和研究助理（助研）的经历比较感兴趣，我当时将未来的职业目标定在了大学教学和学术研究上。等到我在MIT[5]的第三年过半，我已经下定决心在毕业后攻读博士学位，因为这是实现我职业目标的必经之路。我

不到但难以逾越的障碍，通常见于少数人种和女性身上，且与这些人群的资历和成就并无关联”的现象。——译者注

[4]UCLA 是加州大学洛杉矶分校（University of California, Los Angeles）的简称。——译者注

[5]MIT是麻省理工学院（Massachusetts Institute of Technology）的简称。——译者注

决定留在MIT，完成一个五年制本硕连读的项目，以此来在申请博士项目之前积累更多的研究经验，以期能被录取到更多排名顶尖的院系中去。

我找到了一位硕士论文的导师，并且就像很多壮志踌躇的年轻人一样，开始向他阐述自己不甚成熟、遑论完善的研究思路和计划。我的导师很耐心的听我说完了我的想法，但最终仍然成功地说服我进行一些和他的研究兴趣更契合、更主流的研究，而且更重要的是，这些研究项目更契合他的基金项目要求。因为当时我的硕士项目学费一部分来自我的导师从美国政府申请的一个研究基金，我有义务在基金所规定的范围内完成研究工作。因此，我听从了他的建议，并将接下来两年半的时间用于开发一类新的原型工具，用于分析由C和C++语言编写的计算机程序的运行时行为。[6]

虽然我并不是非常热衷于我硕士论文的项目，但事实证明选择一个和导师研究兴趣契合的项目是一个明智的选择：在他有力的指导下，我发表了两篇论文——一篇我被列为第一作者（主要作者），另一篇的位置稍微靠后——并且我的硕士论文获得了系里年度最佳毕业论文奖。这些成就，加上导师在我申请文档中的帮助，为我赢得了几所顶尖计算机科学系博士项目的录取。因为斯坦福是我的首选，在我收到录取的当天晚上，我甚至激动得几乎无法入睡。

我还非常幸运地得到了NSF[7]和NDSEG[8]研究生奖学金的垂青，这两者

[6]运行时行为分析（run-time behavior analysis）是计算机程序编写和维护中一种重要的技术，它可以帮助软件工程师更快、更准确地发现在程序编写时难以注意到的程序缺陷和漏洞（bug），进而完善其功能并提高稳定性。Bug一词的原意是虫子，现代被用于形容软件缺陷。这一说法一个可能的来源，是源于1947年电子管计算机刚刚问世前后。当时在哈佛大学的二号机（Mark II）中，一位操作员仔细排查后发现，二号机当时一些计算谬误的产生，是由于一只飞蛾被困在了一个继电器上。这件事被记录在了当时的实验室日志中，后来常被称为“第一次真正找到的bug（此处为虫子和程序缺陷的双关）”。——译者注

[7]NSF是美国国家自然基金会（National Science Foundation）的简称。NSF是支持美国各大高等院校进行基础自然科学研究的重要基金来源之一。该奖学金的申请只面向美国公民及永久居民开放。——译者注

[8]NDSEG是美国国防科学与工程研究生奖学金（National Defense Science & Engineer-

都只颁发给了大概5%的申请者。这两个奖学金为我免除了攻读博士的六年之中，五年的全部费用，也使我不必完成各种研究基金相关的研究项目。与我不同的是，在我的研究领域中，大部分博士研究生依赖于教授提供的研究经费和院系提供的助教经费。博士研究生的经费包括全额学费，以及大约每个月1,800美元的补贴，用于贴补生活开支。（在我的研究领域几乎没有人自费攻读博士，因为那样在经济上非常不划算。）

因为我已经有一定的研究和写作论文的经验，当我在2006年9月来到斯坦福时，我觉得自己已经为未来艰苦的博士研究做好了充足的准备。然而，那时我完全没有预见到的是，我的博士第一年即将成为我生命中到此为止最为打击信心、令人灰心丧气的一段时间。

---

ing Graduate Fellowship）的简称。该奖学金由美国国防部出资设立，旨在推动国防相关的科学与工程研究的发展，其申请只面向拥有美国国籍的人士开放。——译者注

第一年：碎梦大道

2006年的夏天，在我开始在斯坦福攻读博士学位的几个月前，我考虑了一些我认为自己感兴趣研究的课题。大题而言，我想要创造一些创新性的工具，用来帮助人们在进行计算机编程时提高效率，换言之，*提高程序员生产力*。我之所以对这个方向感兴趣，主要源于我在暑期实习中自己的编程经历：因为日复一日，公司分配给我的工作并不能让我提起太多兴趣，工作中的很多时间，我都坐在自己的格子间里反思，在我工作的这些公司中计算机编程的流程如何低效。那时我认为，如果能投身于旨在降低这种低效性的科学研究，应该是不错的方向。更宽泛地说，我的研究兴趣集中在能让所有类型的计算机使用者更加高效的工作中——而非仅仅聚焦在专业程序员的身上。举例来说，我希望能设计出新的工具，能帮助科学家分析和绘制数据、帮助系统管理员调整服务器配置、或者帮助计算机新手学习使用新的软件。

尽管我当时就有这些模糊、不成型的兴趣，但距离我将这些兴趣转化为真正可以发表的研究项目，并最终形成一篇*博士论文*，还有很多年的差距。对斯坦福计算机科学系的博士研究生而言，通常他们需要发表二到四篇第一作者的学术论文，并将这些论文合并成一篇博士论文——通常是一篇长度与书籍相仿的科技文档。当博士论文通过一个由三位教授组成的*博士论文委员会*批准后，学生就可以毕业，从而获得博士学位了。在我所在的计算机系，通常一个博士研究生需要四到八年毕业，这个年限取决于他们发表文章的效率。

在2006年9月的新生信息会[9]上，系里的教授鼓励所有的博士新生尽快找到自己的导师，所以我和我的同学们一样，把一开始的几个月花在了找教授谈话上，希望尽快找到一个研究方向匹配的导师。对于一个学生的论文委员会来说，导师扮演了最重要的角色，因为他/她对学生能否毕业拥有最终的决定权。在我的研究领域，导师通常还负责通过自己的科研经费为学生提供经费支持，并且指导他们开展课题研究、写作论文。和几位教授谈话后，我发现Dawson[10]和我的研究兴趣和研究风格都似乎最为接近，所以我选择了他作为我的导师。

在我刚去斯坦福的时候，Dawson已经在斯坦福度过了八年，并且刚刚获得终身教授职位；通常，教授在他们工作的前七年如果发表了足够多高水平的研究论文，就可以得到终身职位（终生的工作保障）。Dawson的主要研究兴趣是创造新的工具以自动在复杂、真实的软件中寻找bug（软件代码中的缺陷）。在过去的十年间，Dawson 和他的学生编写了许多这样的工具，相比他们的竞争者而言，他们能找到程序中更多的bug。他们的研究成果十分有效——他们甚至城里了一个成功的创业公司，通过提供基于这类技术的缺陷检测服务而盈利。尽管我对Dawson的研究项目感兴趣，更吸引我的一点则是他的研究哲学和我自己的想法十分接近：他是充满激情的“务实派”—— 相比于单纯为了显得“学术”而去研究理论上“新颖”的课题，他更关注的是得到实实在在、有说服力的结果。

我和Dawson第一次面谈时，他似乎对我的大方向——让计算机的使用和计算机编程变得更高效——只是稍感兴趣。不过，有一点他说的很清楚：他非常希望招收一些新的研究生来帮助他完成一个叫Klee的软件缺陷

[9]信息会（Orientation），或译为迎新会，是在学校或大型组织中常见的，统一为新成员提供常用信息、帮助其尽快适应环境的活动，通常由资深成员主持。

[10]Dawson是这位教授的名字（first name），而非姓（last name），故翻译成Dawson教授是不准确的——原文也没有出现Professor Dawson（Dawson教授）这一称呼。单独使用名字在英语中非常普遍，通常用于熟人之间互相称呼、陌生人之间互相介绍、以及不希望提及全名的场合。文中出现的人名都是单独的名字。为了忠于原文的阅读体验，译者没有对这些名字进行标准化翻译和加工。—— 译者注

检查工具，因为这个项目是他现在的科研经费所支持的。（这个工具有好几个名字，但为了简便，这里就叫它“Klee”。）和其他的教授和高年级博士谈过之后我才意识到，对于新生而言，加入一个已有的、由研究基金支持的项目是一种常规现象，而并不是马上开始进行自己原创的研究项目。我说服了自己，认为自动查找软件缺陷也是一种间接提高程序员工作效率的研究，于是便打定主意，加入Klee项目组。

当2006年12月，我准备加入Klee项目组时，Dawson已经在指导五个学生参加这个项目了。项目组的带头人Cristi是第三年的博士生，而正是他和Dawson开发了最初版本的Klee。Dawson、Cristi和其他研究者不久前还发表了第一篇说明Klee系统的论文，并展示了Klee在发现一些新的缺陷上十分有效。那篇论文受到了学术界的好评，所以Dawson希望保持这一势头，继续发表几篇跟进这一项目的文章。值得注意的是，从同一个研究项目中发表多篇论文是可能的（即“跟进论文”），只要这些新的论文有新的原创观点，相比前作的改进和创新，或者相比前作而言在结果上有很大的改善。当时，下一个相关领域的顶级会议的论文提交截止日期是2007年3月，所以Klee 团队有四个月用于基于之前的论文做出创新性的改进，以期发表一篇新论文。

~

在我继续讲述我的故事之前，我想简单地介绍一下学术论文是如何评审和发表的。在计算机科学领域，发表文章最受关注的场合是学术会议。当然，值得指出的是，再很多其他学科中，学术期刊才是最受关注的，而对这些领域而言，“学术会议”往往和计算机科学领域大相径庭。对计算机科学而言，学术会议的论文发表流程大概如下：

1. 每个会议发布一个征稿启事，其中说明了会议所要求的课题范围和一个论文提交的截止日期。

2. 研究者需要在指定的截止日期之前提交自己的论文。通常每个学术会议会收到100到300份论文稿件，每篇论文大概包含30到40页双倍行距[11]的文字。

3. 学术会议的程序委员会（Program Comittee，简称PC）通常由大约20位专家研究者组成，他们负责将论文负责分类，以便审稿。每篇论文通常由三到五个人完成评审，参与评审的人员可能包括PC的成员，或者由PC成员邀请的、来自学术界自愿参与审稿过程的审稿人。论文的评审过程通常需要大约三个月。

4. 当每位PC成员都完成审稿后，整个委员会将开会商议，通过审稿人的反馈决定接收一部分论文稿件，并拒收剩下的稿件。

5. 程序委员会会发出电子邮件通知所有的作者，告知他们论文是否被接收，并将审稿人对他们的论文提出的审稿评价附在电子邮件中。

6. 论文被接收的作者参加学术会议，并关于自己的论文做一个30分钟左右的演讲。学术会议结束后，所有的论文都将被收录在在线的数字图书馆中。[12]

通常，一个备受关注的顶级学术会议的论文接收率在8%到16%之间，而第二级的学术会议大概接收20%到30%的投稿。由于这些接收率相对较低，对于一篇学术论文来说，被拒收、修改并重新提交并不罕见——在论

[11]双倍行距排版时，行间距与文字高度相同，通常用于学术期刊初稿的排版。而计算机科学的学术论文通常采用单倍行距、双栏排版。——译者注

[12]根据研究领域不同，学术会议的举办方式略有不同。在一些领域中，论文会被PC分为演讲展示（oral presentation）和海报展示（poster presentation）两种形式。演讲展示中，通常有5至30分钟供作者在报告厅等场合公开真实自己的研究成果；海报展示时，所有参与的作者则将自己的研究成果展示在一张海报上，并以海报为基础向观众展示自己的研究。通常演讲展示会有更多听众和更大的影响力，在有演讲展示和海报展示之分的学术会议中，PC也会把有限的演讲展示机会分配给他们认为更有影响力的学术研究项目。—— 译者注

文最终被接收之前可能这一过程要重复多次，而这一过程可能会花费数年的时间。（在同一时间，一篇论文只能被提交到一个会议。）

~

当Dawson说他想要提交一篇论文到2007年3月截止提交的顶级会议之后，他向我介绍了当时其他五个学生工作的方向，并让我选择自己感兴趣的工作。我选择了使用Klee 来寻找Linux*驱动程序*中存在的新的程序缺陷。*驱动程序*是指用于帮助操作系统完成其与外置设备（如鼠标或键盘）通信的软件代码。而Linux*操作系统*，和微软Windows系统或者苹果Mac OS系统类似，包含了成千上万这样的驱动程序，用以连接各种各样的外置设备。对于传统的调试方法而言，驱动程序中的软件缺陷十分难以找到，甚至有可能是危险的，因为驱动程序的缺陷可能会导致操作系统死机甚至崩溃。

Dawson认为Klee可以在Linux驱动程序的成千上万行代码中，找到其他自动缺陷分析软件（甚至人）从未找到过的软件缺陷。我记得我当时考虑过，尽管在Linux驱动程序中找到缺陷写在论文里看起来很不错，但我并不是很明白这样的工作能不能算是真正的研究贡献。按照我的理解，我要做的事情是用Klee去寻找程序缺陷——将一个已有的研究应用在实际问题上——而不是采用一种创新的方法来提高Klee的性能。此外，我并不明白，到三月份论文截稿时，我的工作和其他五个学生的工作如何能融合成一篇自洽的论文。尽管如此，我当时相信Dawson 在头脑中有一个高瞻远瞩的思路完成这篇论文。介于我刚刚加入研究项目，我并不想马上开始对这些应该由教授决定的问题开始指手画脚。任务已经摆在眼前，我要考虑的只是尽我所能，完成这个任务。

~

我博士生涯的前四个月被我用于配置Klee以用它来分析上千行的Linux设备驱动程序，以期从中发现新的程序缺陷——这一过程并不

顺利。尽管看起来我的任务并不复杂，但我很快就被淹没在了一些细节问题中，而这些细节对于让Klee能够分析Linux 驱动程序又是必不可少的。我经常会花几个小时设置好Klee所需的复杂的实验环境，以分析某一个驱动程序的程序缺陷；但这种工作又往往以Klee因为其自身的程序缺陷而崩溃告终，让我的努力付诸东流。当我把这些Klee中存在的问题报告给Cristi时，他会竭尽全力解决，但由于Klee本身极其复杂、环节众多，在其中找到并解决程序缺陷绝非易事。我并不是想要专门指责Klee：任何以科研为目的开发的原型软件都会或多或少有一些难以预见的缺陷。我的任务是用Klee去Linux驱动程序代码中寻找缺陷，但讽刺的是，我整个第一个月的工作都变成了在Klee里找缺陷。（Klee不能在自己的代码里自动找到缺陷，这实在是太遗憾了！）随着时间的流逝，我对于手头的工作越来越感到沮丧，我觉得自己被分配的任务就是单纯的苦力，毫无知识含量——我的时间都用在让Klee能正常运行上了。

这是我生命中第一次感受到被手头的工作所淹没的无望。以前，我的暑期实习都相对不难，而且就算一些学校的作业对我来说有些挑战，作业中也总有一个需要找到的正确答案等着我。如果我课上有没听懂的内容，助教或者高年级的学生总可以帮我答疑解惑。就算在本科进行学术研究的时候，我也总能请辅导我的博士学长帮忙，因为我当时处理的问题相对比较简单，而他通常知道问题的解决方法。对于本科的研究助理而言，对学术研究的投入以及期望也相对较低：科研只是我日常生活很小的一部分。如果我在某个科研问题上毫无头绪，我可以选择集中精力在课程作业上，或者干脆和朋友们出去玩。本科毕业也和学术研究毫无关联。然而，作为博士研究生，学术研究是我唯一的工作，除非我在学术研究上能有所成果，否则我将得不到博士学位。我难以把自己的情绪和每天的研究进展分离开来，而在那几个月中，我的研究进展出奇得缓慢。

我现在步入了一个完全陌生的领域，所以我很难再向本科时候那样向别人寻求帮助，因为问题的答案往往是不确定的。因为我是唯一试图将Klee应用于驱动程序代码的人，我的同事们并不能为我提供任何指

导。Dawson偶尔会给我一些较高层面的策略性的建议，但就像所有已经获得终身职位的教授一样，他的角色并不是和他的学生们一起“在战壕里（第一线）参与战斗”。在做出研究成果的过程中，弄清楚所有复杂的细节是我们学生的任务——对我来说，这些细节就是如何找到别人从未发现过的、Linux 驱动程序中的软件缺陷。教授们都喜欢重复这句老生常谈：“如果有人曾经做过这样的工作的话，那就不能叫做学术研究啦！”这是我第一次亲身体会到了这句话的含义。

尽管我每天都觉得工作很无望，但我仍然不断安慰自己：*我只是刚刚开始在这里工作，我应该保持耐心*。我不想在我的导师或者同事面前显得软弱无能，尤其因为我当时是Dawson组里最年轻的学生。所以，我在超过100天的时间里，每天修复Klee不断产生的新问题，不断遇到新的、更加棘手的问题，在我的“寻找Linux驱动程序缺陷之旅”中，拖着沉重的脚步前进。

当时，在我醒着的每时每刻，我不是在工作，就是在思考研究中的问题，抑或是在因为自己在研究中被艰深的技术问题所困扰而感到沮丧。与一般的朝九晚五的工作（比如我的暑期实习）所不同的是，以前每天晚上我可以把工作留在办公室，坐在电视前放松自己；而现在的学术研究在情绪上和心理上都是无休无止的。我晚上几乎没法让大脑停止思考问题，尽情放松休息——后来我发现几乎所有的博士研究生都有类似的困扰。有时，因为我的研究任务繁重得难以想象，我甚至会因为压力而失眠。想要在研究中休息一段时间也几乎不可能，因为在3月份的论文提交截止日期之前，还有很多工作需要完成。

在做这些苦力时，我曾经试图想出一些半自动的方法，可以让每天的研磨不那么辛苦。我和Dawson交流过一些初步的想法，但我们最终的结论是，如果我们想让Klee能在Linux 驱动程序里找到缺陷，这种耗时间的研磨是无法避免的。接下来，在论文提交前的几个月里，我只能继续辛苦工作。

理性而言，我十分理解在科学和工程领域中，通过实验式的研究方法获得研究成果，通常并不光鲜、甚至包含很多辛苦的工作。而博士研究生，尤其是第一、第二年的博士生，通常不得不在这一过程中参与最辛苦、最劳力的工作——这是他们的资金来源所决定的。通常，在一个研究组中，教授和高年级博士生制订高瞻远瞩的研究计划，并将任务布置给低年级的学生，由他们去研磨所有的细节，并让项目实际运转起来。第一、第二年的学生通常很难影响研究组项目的大方向。尽管我完全接受自己在这种“等级制度”下较低的地位，但感性上，我仍然深受打击，因为手头的工作实在是太他妈难了[13]，而且毫无成就感可言。

~

经过两个月的研磨之后，我收获了一些小小的成功。我成功地让Klee能在一些较小的驱动程序上良好工作，并开始找到了几个程序缺陷。为了确定这些缺陷是否真实存在（而非因为Klee的局限而导致的误报），我向开发这些驱动程序Linux的程序开发者发了几封电子邮件，说明了这些可能存在缺陷的地方。几位开发者回复了我邮件，确认了我的确是在他们的代码中找到了真正的缺陷。当我收到这些确认的邮件时十分激动，因为这是我第一次受到外界的认可，尽管它们只能算很小的鼓励。尽管我没有做出什么革新性的研究，我依然收获了一些成就感，因为这些缺陷在没有Klee的帮助下可能非常难以找到。

在这些小的驱动程序里的缺陷得到确认之后，我的士气有所恢复，所以我开始着手分析更大、更复杂的驱动程序。然而，接下来的几周中不断浮现的技术问题逐渐变得难以承受，并几乎把我推到了崩溃的边缘。这里简单总结一下我当时遇到的问题：Klee 实际上只能在不超过大概3000行，由C语言写成的代码中找到缺陷。最小的Linux驱动程序代码大概有100行，所以Klee分析它们游刃有余。而稍微大一些的驱动程序就有大约1000 行代码，而这些代码还与大约一万行到两万行的Linux操作系统

[13]原文此处为“so damn hard”，此处为保持原文语气，进行了直译。

代码密不可分。这就导致问题一下远远超过Klee可以分析解决的范围，因为Klee并不能把这1000 行代码和其他部分“斩断联系”而单独分析。我试过很多方法减少这些外部的连接（术语称为依赖），但是这样做意味着对每一个驱动程序，我将需要花费几天的时间去完成十分复杂的工作，而最终这些工作也只对这一个驱动程序有效。

我和Dawson面谈了一次，向他说明了我面临的问题的复杂程度，也表达了自己对此的消极情绪。对我来说，在每一个新的驱动程序上都要花费几天的时间才能让Klee可以运行，这简直是荒唐至极的工作。这种工作不仅让我疲惫不堪，它本身甚至不能算是学术研究！在我们的论文里，我应该怎么描述我的工作——我花了大概1000小时的苦工，就为了让Klee能分析一堆驱动程序，除此之外我并没有其他可说的了？这根本不是学术贡献，反而听起来太愚蠢了。另一方面，我开始着急，因为距离论文提交截止只有五周了，而Dawson似乎还没有提过我们组的研究论文应该从什么思路着手写作。通常一篇优秀的学术论文需要至少四周时间才能完成，而且我们的情况可能更加复杂，因为我们有六个学生各自负责项目的一部分，我们需要把这些部分融合在一起。

我和Dawson面谈过后几天，他提出了一个改进的思路，让Klee能够克服我所面临的依赖问题。他提出的这个观点叫做约束下运行（UnderConstrained execution，简称UC），而这种方法也许能让Klee将Linux驱动程序和外部的一两万行代码剥离开来，并单独在驱动程序代码中进行分析。他马上就开始和一个高年级的学生开始将UC技术融合在Klee 中；他们把改进后的版本称为Klee-UC。尽管我当时筋疲力竭、接近崩溃，但我仍然非常欣慰看到自己的痛苦挣扎至少激发了Dawson 的灵感，想到了这个有朝一日可能成为学术贡献的新思路。

接下来，Dawson和那名学生在Klee-UC上花了几周的时间。与此同时，他们让我继续用原来的方法，手工在Linux驱动程序中寻找缺陷。他们打算通过用Klee-UC试图找到我用普通Klee 找到的缺陷，来证明其有效性。他们想在论文里阐述的观点是，与其让一个博士生（就是我！）为了

找到每个缺陷而在繁杂的工作上花费几天的时间，不如用Klee-UC来在几分钟之内找到这些缺陷，而且不需要准备任何特定的实验环境。

在近乎疯狂地又研磨了几周之后，我终于让原版的Klee成功分析了937个Linux驱动程序，并找到了55个新的软件缺陷（其中32个收到了开发者的确认）。接着，我还得把刚刚成型的Klee-UC配置好，并准备分析这937个驱动程序。这比之前的配置还要麻烦，因为在我准备用Klee-UC分析驱动程序的同时，Dawson和那个学生仍然在（编程）实现它。谢天谢地，最终Klee-UC的确找到了我找到的缺陷中的大部分，这样我们做了一些学术贡献，也有成果可以在论文里展示了。

但是这里有个巨大的问题。等到我们得到那些不错的结果时，到论文截止日期已经只剩下三天了，而我们都完全没有着手开始写论文。在这么短的时间里，撰写、编辑、完善一篇文章，使得它能被顶级计算机科学学术会议录用是完全不可能的。但我们还是努力这么做了。在提交截止前的72小时里，Dawson和我们五个学生（其中一位现在已经退学了）两个通宵住在办公室，以期完成实验和论文。我们所有学生心里都知道这篇论文不可能被接收，但我们还是在Dawson的带领下，顺从地向目标前进。

最终，我们提交了一篇令人尴尬的文章，里面充满了错误拼写、不通顺的句子片段、没有解释的图表，甚至连结论段都没有。整个是一团乱麻。在那一刻，我难以想象，如果我的研究工作一直是这样毫无章法，我到底要如何从博士项目中毕业。三个月后，不出所料，我们收到的审稿意见极其消极，甚至包括这样的严厉谴责：“程序委员会（PC）认为这篇论文简直太过草率，根本不可能被录用；请不要提交这种甚至无法审阅的论文。”

~

这次惨痛经历之后，我申请了去谷歌实习，因为我急需换一换环境。那份实习和我的研究兴趣毫不相关，但我并不在意。当时我只想逃离斯坦福几个月。

这时是2007年4月份了，距离6月份我的实习开始，还有十周的时间。我不知道我能做些什么，但我想逃离我之前四个月折腾Klee和Linux驱动程序的经历，逃得越远越好。我根本不关心我们是不是会修改并且重新提交那篇论文（最后没有）；我只想逃离那段梦魇。但因为我已经积累了近千小时的使用Klee的经验，而这也是Dawson当时唯一关心的项目，我想这可能是一个开展新课题不错的切入点。于是，我找Dawson谈了谈，和他交流了一些“不寻常”的使用Klee的方法，而不仅仅是局限于寻找软件缺陷。

然而我很快就意识到，我完全没必要把自己局限在Klee上，因为我的经费来自NDSEG 奖学金，而非Dawson的研究经费。而Dawson其他学生的境况则于我不同，他们除了继续在Klee项目中工作之外别无选择，因为他们都是靠Klee的研究经费支持的。所以我决定仍然让Dawson作为我的导师，但离开Klee项目组去从头开始开展我自己的研究项目。

为什么我不早点“单飞”呢？因为尽管我的奖学金理论上给予了我研究方向的自由，但我知道我仍然需要一位导师的支持才能最终毕业。当时，Dawson显然想让他所有的新生都把精力投入到Klee上，所以我就花了四个月的时间在Klee上研磨，塑造了一个“好士兵”的形象，而不是一开始就傲慢地要求做自己的项目。而且，就算我选择了别的导师，我一开始还是会需要花时间在他们的项目上，来证明自己的能力。这种“交学费”的工作根本无法避免。

接下来的十周里，我在一个完全真空的环境思考自己的研究思路，完全没有与人交流。因为我一开始对研究组留下了如此消极的印象，我现在只想自己一个人清静地思考。Dawson对我的消失并没什么意见，因为他并没有在用自己的研究经费支持我。

我把自己完全与世隔绝，心理上基本处于崩溃状态，但我任然试着每天能有所进步。我尝试着每天阅读几篇计算机科学的学术论文，并做好笔记，以帮助我激发自己的灵感。但因为没有良好的引导和研究背景，我最后浪费了很多时间，有时读完了论文却一无所获。我还曾经骑着自行车在

校园里漫无目的地游荡，试图在过程中思考自己的研究思路但常常无功而返。最终，我把时间都浪费在了拖延上，可能甚于我这辈子至今为止的任何一段时光：我看了很多电视节目，睡了很多觉，花了无数个钟头在网上蹉跎。与我的朝九晚五的朋友们不同的是，我没有老板每天监督我的工作，所以我可以无拘无束，让我的大脑自由思考。

尽管我对研究方向的头脑风暴大部分都毫无目标，但我的想法逐渐地开始向一个问题聚拢：*我们如何才能经验性地度量软件的质量呢*？这是在我开始读博之前，受到工程性实习中遇到的低质量的软件而启发，所想到过的一个感兴趣的研究方向。然而，在真空中空想研究思路的问题所在，是我当时没有足够所需的经验来把这些思路转化为现实的项目。因为我还没有准备好，显然这种完全的思想自由在当时更像是一种诅咒。

尽管我对创造新方法来度量软件质量非常感兴趣，我当时也明白这只能是一个模糊的梦，没有足够的形式化的研究方法支持，是不可能被学术界所接受的。如果我当时自己闭门造车研究这个项目，我的下场只能是成为一个胡说八道却又假装内行的人。我永远不可能有机会把这些想法发表在顶级、甚至次一级的学术会议中，而发表不了论文就意味着不能毕业。那时，我已经不再抱有虚无缥缈的梦想，想要成为一名终身教授了：我只想能找到个办法毕业。

在那整整十周的与世隔绝中，我几乎没跟任何人说话——甚至包括朋友和家人。找人抱怨也毫无意义，因为没人能明白我当时经历的一切。我的不读博的朋友们以为我就是“在学校上上课”。而少数几个我在系里新认识的朋友也和我一样，正在因为他们博士第一年里的挣扎感到抑郁——这些挣扎的主要部分，多半就是不得不直面充满挑战、不知从何入手的研究问题带来的震惊，以及无法影响实验室研究方向、只能随波逐流的无奈。我们这一群年轻的计算机科学家来到这里，自愿地投身于复杂而看起来毫无意义的工作中，所收获的却只是我们在公司工作的朋友们四分之一左右的工资。这种现实简直可悲到可笑的地步。但我觉得大家凑在一起互倒苦水也毫无裨益，所以我选择了闭嘴。我选择了不来计算机系的大楼里工

作，因为我害怕碰到同事、同学们。我害怕他们会不可避免地问我我现在研究的课题是什么，而我并没有什么可说的东西。藏身于图书馆或者咖啡店让我感觉更为安逸。

反观当初，这么早脱离实验室单干是一个非常糟糕的选择。和想象中一位孤独的学者坐在路边，一边喝着拿铁咖啡一边在一张笔记纸上涂涂画画这种浪漫化的场景不同的是，真正的学术研究从来就不是在“真空”中完成的。为了创新，一个人需要有坚实的知识、历史、甚至物质基础（比如实验室设备）。在那几周中，我应该追寻的更明智的路线是找更多机会和Dawson谈话，并积极寻求和其他教授以及高年级学生的合作。但当时的我太过崩溃和沮丧，不满于“等级分明”的基于研究组的学术风格——把新来的博士研究生放在最繁杂枯燥的工作中——所以产生了逆反而选择了单飞。

~

在我十周的与世隔绝接近尾声，即将去谷歌暑期实习之前，我给Dawson发了个电子邮件，简单说明了一下我最近读到的一篇让我思考了很多的科技博客。那篇博客让我想到了一个想法，那就是通过开发者在程序代码存在的整段时间里编辑代码的过程，来衡量一个软件项目的质量。令我惊喜的是，Dawson很快给我发了一个简短的回复，说他对这种衡量软件质量的技术也很感兴趣，尤其是如何用这种技术来帮助Klee这样的自动缺陷检测工具。

当知道Dawson对我感兴趣的领域也感兴趣之后，我再次燃起了希望，因为他也许能帮助我把这个想法变得更接近现实。我很快的为这个新想到的经验化软件质量度量的项目记下了一些笔记，并打算从暑期实习回来后开始研究这个课题。这样，在四个月与Klee的痛苦研磨，以及十周漫无目的的找寻方向之后，我博士生涯的第一年画上了一个还算乐观的休止符。

第二年：整装启程

在谷歌的暑期实习对我而言，是一段脱离学术研究的轻松时光。我的工作压力并不大，所以我经常有机会和其他实习生一起闲聊、交朋友。临近夏天结束时，我已经从第一年的痛苦和折磨中痊愈，做好了重新开始博士生活的准备。

在夏季最后几天，我给Dawson写了一封电子邮件，重申了我对追求自己研究兴趣的渴望，但同时强调了从研究中发表论文对我而言的重要性："我从这个暑期以及之前的暑期实习经验，发现自己很难集中精力完成一个博士科研项目，除非我对这个项目有强烈的归属感和科研的热情；所以我非常希望能在令我感到激动的研究方向，和学术界中的教授们广泛认为具有'研究价值'的问题中，找到一个交集，并为之奋斗。"

我准备继续和Dawson一起在经验化软件质量度量的课题上做一番研究，因为我们已经在我第一年结束的时候讨论过这一问题。然而，我当时预感这个项目可能会风险较大，因为这不是Dawson主要的研究兴趣，也不是他主要的专业知识的所在；对他而言，Klee仍然是最高优先级的工作。于是，我想到自己可以去寻找另一个课题，通过同时在两个课题上做研究（并寄希望于至少一个能成功），来冲淡风险。我将在这章稍后介绍我和Dawson合作的主要研究项目，但我想先说说我的另一个项目。

~

就在2007年9月，我准备开始我博士生涯的第二年之前，我去波士顿度假了一周，看望我大学时的朋友。因为我当时离得不远，所以我给几位我本科时就认识的MIT教授发了邮件，希望可以向他们寻求一些指导。他们和我见面的时候，大概都谈到了同样的事情：*要主动去找教授谈话，尽量去找一个共同的兴趣点作为研究课题；不论如何，把自己关起来闭门造车是万万不可取的*。这句简单的建议在我博士接下来的五年里反复得到印证，也最终指引我顺利完成了博士学习。

还在波士顿时，我就马上将这个建议印在心里并付诸行动。我给MIT计算机系一位叫Rob 的教授发了一封*冷邮件*（向从未通信过的人冒昧发送的邮件），并礼貌地请求和他面谈一次。在这封邮件中，我简单地说明了自己是MIT毕业不久的学生，现在在斯坦福攻读博士，并希望能开发出一些让计算机程序开发者能提高工作效率的工具。因为我知道Rob也对这一研究领域有兴趣，我希望他能回复我的邮件，而不是把它标记为垃圾邮件。Rob非常慷慨地在他的办公室和我面谈了一个小时，期间我向他介绍了几个研究项目的想法，希望能从他那里得到反馈。他似乎对这些想法很感兴趣，这让我信心倍增，因为我的想法或多或少得到了一位研究领域内的教授的认可。可惜，因为我不再是MIT的学生，我没有机会和Rob一起开展研究。在我们面谈的最后，Rob推荐我和斯坦福计算机系一位名叫Scott的教授谈一谈，看看我有没有机会成功向他推销这些观点。

回到斯坦福后，我给Scott发了一封冷邮件，请他给我一个机会面谈。去面谈时，我针对想和他沟通的三个具体的研究思路准备了一些笔记，大致格式如下：

1. 问题的定义是什么?

2. 我准备用什么方法解决?

3. 用什么样的实验，才能有说服力地证明我的方法具有良好的效果?

Rob的一位博士学生，也是我的朋友，Greg，教会了我其中第三点——用实验指导思考—— 在提出研究项目思路中的重要性。教授们很乐意让自己的名字出现在发表的论文上，而计算机科学领域的学术论文通常需要强有力的实验支持才能有机会发表。因此，在提出项目的想法时就考虑到实验设计是十分重要的。

尽管我的介绍都没能赢得Scott的全力支持，但他仍然希望可以和我一起开展一个和我大致研究兴趣相关的研究课题。当时，他是一位刚刚来到斯坦福三年的助理教授（尚未获得终身职位），所以他非常希望能多多发表论文，早日拿到终身教职。因为我自己有奖学金，Scott并不需要用自己的经费支持我，所以对他来说也没有什么顾虑。

~

Scott所精通的，是一个称为人机交互（Human-Computer Interaction，简称HCI）的、较为贴近实用的子领域。和其它子领域不同的是，HCI的研究方法以人的需要为核心。一般来说，HCI的研究项目都是这样完成的：

1. 观察人，并弄清楚他们面临的真正问题是什么。
2. 设计一些创新性的工具，以帮助缓解这些问题带来的影响。
3. 用实验来评估这些工具，以检验它们是否真的可以帮助人们解决问题。

因为我希望创造能提高程序开发者工作效率的工具，Scott就先建议我去开发者们真正的工作场所，观察他们的工作，并借此发现他们真正面临的问题。具体来说，Scott非常好奇为什么现代的软件开发者使用多种编程语言，并且严重依赖于在网上搜索、并剪贴代码片段来完成开发。之前几十年的高效开发工具的研究中，都假设了开发者只用一种语言进行开发，

并且认为他们所处的环境中开发者的水平大致相同，而这些假设现在显然都早就过时了。通过观察现代程序开发者的日常活动，也许我就能设计出更贴合他们需求的新工具。

接到Scott给我的定下的大致目标之后，我就马上着手开始寻找可以让我观察的、专业的软件开发者。因为我刚刚在谷歌实习，我试过在那里找到目标，所以我发邮件给了我实习时的经理，他也同意将我的邮件转发给其他同事。我很快就收到了几封婉拒的邮件，因为没人想因为一个非雇员看着他们工作，而惹上知识产权的麻烦。接着我又给一些在斯坦福附近创业的朋友们发了邮件，但很快就发现他们和大公司的程序员一样，也收到种种限制。遗憾的是，他们更难以答应我的请求，因为为了避免竞争者抄袭，他们需要对自己的创业项目保密。而且，他们远比大公司的开发者们更加繁忙，所以也不太希望浪费自己的时间，去帮随便一个研究生完成这种对他们没什么帮助的科研项目。

我当时最后的希望是去谋智（Mozilla）观察软件开发者工作，因为谋智是一个非营利性的软件开发组织，他们的产品包括著名的火狐（Firefox）浏览器。因为谋智的软件项目都是开放源代码[14]的，所以我以为他们应该不怕会有一个局外人来观察他们的程序员工作。我没有试着发冷邮件（其实我都不知道应该要给谁发！），我决定直接开车去谋智的总部，直接登门造访。我冒然和我看到的第一个人打了招呼，并向他介绍了自己。他很热心地告诉了我两位谋智高层领导的电子邮箱，并说明这两位可能会对这样的研究合作感兴趣。我当天就给两个人发了冷邮件，令我感到惊喜的是，其中一位回复称对我的研究很感兴趣。不幸的是，那也是我收到的来自他的最后一封邮件；当我请求继续开展研究时，他就再也没有回复我了。

[14]开放源代码（Open Source），简称“开源”，是一种在自由软件开发者中非常流行的形式。开发者自愿将自己软件的源代码向其他开发者公开，而在一定的通用协议下（通常被称为“证书”），其他开发者可以基于这些软件代码进行改动和再次开发。开放源代码有助于鼓励开发者分享自己的成果，避免重复劳动。当然，尽管软件源代码是开放、可供人任意下载的，有些开源软件也是可以盈利的。—— 译者注

现在反观当时，对于自己试图在工作场所观察别人工作的尝试以失败告终，我并不感到奇怪。毕竟，我并不能为那些被我观察的专业程序员带来任何好处；相反，我只会干扰他们的日常工作。幸运的是，几年后，我有机会观察了一组（非专业）程序员—— 为科学研究而编写程序的研究生们——的工作，他们并不排斥我可能带来的小小打扰，反而非常乐于和我交流他们工作环境中的问题。这些访谈的经历，最终激发了我的灵感，并直接为我的毕业论文做出了很大贡献。

~

在寻找专业程序员的路线上碰了南墙，我就决定回头，开始在斯坦福内部寻找这样的机会了。当我看到一张海报，宣传系里即将举办一年一度的编程竞赛时，我马上给主办者发了一封冷邮件。我向他说明了自己想要观察学生们比赛过程的想法，他欣然接受了。

尽管学生的编程竞赛对说明现实中的软件开发环境而言毫无代表性，但总是聊胜于无。Scott的一个比我高一年的博士学生，Joel，也希望来参加这种观察。Joel和我把整个周六上午四个小时都花费在了观察几个学生参加比赛这件事情上。我们看到的结果很无聊，最终也没能从我们的笔记中整理出太多有用的东西。然而，这个经历促使我们产生了一个想法，那就是进行条件更加可控的*实验室研究*。

这时，Scott已经决定让Joel和我合作进行研究，一起开展一个实验室研究，而不是分散精力在不同的项目上。因为Joel和我的研究兴趣相似，Scott也可以通过让我们合作，减轻自己管理学生的压力。Joel主要负责了实验室研究的设计，我也同时乐得清闲，因为我同时还在和Dawson做另一个课题的研究。

Joel、Scott和我设计了一个时常两个半小时的实验室研究，研究中，一名斯坦福的学生将会被要求从头开始，编程完成一个简单的网络聊天应用。他们可以使用任何他们想用的资源，最主要的就包括在网上搜索现成

的代码，或者已有的教程。我们找到了20名学生参加实验，其中大部分来自Scott当时在讲授的*人机交互导论*课程的学生中。

在接下来的几周里，Joel和我在系里地下室的机房坐了50个小时（20个学生，每人两个半小时），观察参与研究的学生，并同时将学生使用的电脑屏幕录制下来。一开始，观察学生如何工作非常引人入胜，但好景不长，我们很快就发现这个工作变得冗长繁杂，而且总能看到学生们一遍又一遍地做同样的操作、犯同样的错误。之后，我们又大概花了50个小时重放我们录下的视频，并在视频中标记了一些关键事件发生的时间。最终，我们分析自己的笔记，并回放标记好的关键事件，得到了一些关于“当这些学生面对一个简单、但实际工作中可能遇到的任务时，会遇到哪些问题”的深入了解。

我们把实验室研究的成果写成了论文，并将论文提交到了一个顶级的HCI学术会议。三个月后，我们发现论文被录用了，审稿意见非常令人振奋，而论文甚至还被提名了*最佳论文奖*。在这篇论文里，Joel是*第一作者*，而我是第二作者。几乎所有计算机科学领域的论文都是由多位作者合作完成的，而作者的顺序也是至关重要的。名字出现在首位的作者通常是研究项目的负责人（比如Joel），他通常在项目中完成的部分远多于其他位置靠后的作者，所以也最应当受到承认。其他的作者通常都是项目中的助理——通常是年轻的学生（比如我）或者远程的合作者——这些人也做出了足够多的贡献，可以被列为作者之一。博士生们通常把自己的导师（比如Scott）列在最后，因为导师通常会在研究想法形成、项目规划和论文写作方面提供帮助。

因为我不是论文的第一作者，这篇论文也就对我的毕业论文没有什么帮助；然而，这个经历教会了我许多关于如何开展研究、以及如何写作学术论文的知识。更重要的是，看到这个研究项目最终发表了一篇顶级会议的论文，我感到很有成就感——这和我博士第一年被拒收的Klee的论文形成了鲜明的对比。

Joel后来继续沿着这条研究路线走了下去，把我们的论文转化成了他博士论文的第一部分。在接下来的几年中，针对在一开始的实验室研究中发现的程序开发者们面临的困难，他又创造了几个能帮助克服这些困难的工具，并发表了论文描述这些工具的原理。同时，Scott也没有积极招我入麾下，我也就没有考虑过更换导师的事。当时看来，似乎我和Joel的兴趣太过接近了，而且Joel已经在他的子领域小有成就了。所以，我继续将研究重点放在和Dawson一起研究的课题上，毕竟他仍然是我的导师。然而在这一边，研究进展就没这么顺利了。

~

读者可以回忆一下，在我博士第一年接近尾声时，我开始在一个被称为*经验化软件质量衡量*的子领域寻找研究灵感——具体来说，就是通过软件开发的历史记录，来衡量软件的质量。正好Dawson也对这一课题感兴趣，所以在我博士第二年里，我们就继续在这一问题上开展研究。他的主要兴趣还是在于发明像Klee这样的自动缺陷检测工具，但他也在软件质量方面有一些兴趣。对他而言，让他感兴趣的研究课题包括：

- 如果给定一个大型软件项目，其中包含上千万行代码，如何区分哪些部分对项目而言更加重要，而哪些部分不那么重要?
- 有哪些因素影响了一些部分更容易出现软件缺陷? 比如，是最近刚刚被新手修改过的程序代码更容易包含更多缺陷呢，还是短时间内被很多人修改过的程序更容易出问题?
- 如果一个自动查找缺陷的工具找到了1000个可能的问题，哪些会是其中更重要的呢? 程序开发者可能并没有时间和经历处理全部1000个缺陷，所以他们必须根据一个重要度的评估，来优先完成重要的部分。

我通过分析和Linux系统核心这个软件项目相关的数据集，开始调查这些问题。我选择了Linux作为研究对象，因为它是当时最大且影响最广

泛的开源软件项目，有数以万计的程序开发者在二十年间贡献了几千万行的程序代码。Linux完整的*版本控制历史*能在网上免费得到，所以这成为了我主要的数据来源。项目的版本控制历史记录了在项目的整个生存周期中，所有代码文件中出现过的所有改动，包括改动出现的时间，更重要的是，改动是谁做出的。为了将这些项目历史和软件缺陷对应起来，我从Dawson 的软件缺陷检测公司拿到了一份数据集，其中包含了公司的一个产品在Linux 中发现的2000个缺陷。当然，这不会是Linux里所有的缺陷，但这是我能接触到的数据集中，唯一能为我提供了足够的信息，来研究我们的学术问题的了。

那时，我每天的工作流程就是写程序来从Linux的版本控制历史以及那2000个缺陷报告中提取、和清洗数据[15]，并将它们转化成便于分析的格式，并进行分析。为了能对这些数据有更深层次的认识，我自学了量化数据分析、统计学和一些数据可视化的技巧。在研究过程中，我细致地将自己的实验进展记录在了一本*研究笔记*中，用来记录哪些尝试并没有给我想要的结果。大概每周，我都会和Dawson面谈一次，汇报自己的研究进展。我们的会面通常是我想他展示我从分析中得到的图表或者表格，而他则会提供一些较高层次的评价，比如：“嘿，这张图的这个部分看起来不太对劲。这是为什么呢？试试这样分割你的数据，并做一些更深入的分析。”很多年后，我意识到这其实是一种对于很多学术领域中的*计算研究者们*而言，再平常不过的工作流程；而在我的博士论文中，我也创造了一些工具，来消除这一过程中的一些效率不高的环节。不过那时我还没有这么长远的计划；我所想的只是做出一些有趣的发现，并把他们写成论文发表。

~

[15]数据清洗（Data cleansing）在科研项目中通常被广泛应用，尤其是当数据来自实际的项目，或数据产生、采集的过程没有严格监控时。数据清洗通常包括丢弃数据中对研究没有帮助的部分，或者丢弃因格式、内容、完整性等问题不能被用于研究的“脏数据”。此处作者应该指前者，即将和研究相关的数据分离出来的过程。——译者注

Dawson和我在将我们的成果发表的道路上并非一帆风顺。在那一年里，我们提交了两篇论文，最后都以被拒收场。这个研究项目后来又经过了一年，才转化成为一篇长度较短的、第二级学术会议中的论文得以发表，论文影响力微乎其微，也没有“作数”成为我博士论文的一部分。不过到那时，我已经不太计较这些了，因为我已经在开展其他研究项目了。

我们的论文在发表上遇到了这么多障碍，究其原因是因为我们不是经验化软件质量衡量（有时也被称为经验化软件工程）这一子领域的“圈内人”，而我们的论文恰恰属于这一子领域。在我们步入这个方向之前，已经有几十所大学的研究小组在进行类似的研究了。Dawson和我在这种竞争中完全处于劣势，因为我们的竞争对手包括很多专于此道的教授和科学家，而他们手下通常有很多博士生为他们产生研究成果。这些人对发表论文如饥似渴，因为他们中很多人仍然是年轻的教授，渴望借此得到终身职位。他们也更擅长于一些能让论文更容易被接收发表的技巧，包括统计方法、相关文献的引用，以及“营销技巧”等。最重要的是，他们中很多人在相关的学术会议中担当了外部审稿人或者程序委员会成员等职务，所以他们很清楚如果一篇论文想要被这个领域的学术会议所接收发表，应该需要怎么撰写。

读者可以回忆一下，我介绍过每篇论文都会由三到五位自愿参与的领域内专家——通常都是领域内的教授或者科研工作者——进行同行评审[16]来决定是否可以发表。如果审稿人认为文章值得发表，文章就会被接收并发表；否则，作者就需要修改论文，之后再重新投稿。同行评审的目的在于保证所有被发表的论文都有一定的质量保障，这一保障的水平则由

[16]同行评审（Peer-review）是一种广泛用于计算机科学学术会议的评审模式，其中参与审稿的审稿人可能是领域内的教授、专家、研究者或在读博士生（本科生和硕士生较为罕见），这些人可能也是学术会议中的投稿人（审稿人不能审阅和自己有利益相关性的——比如自己撰写的——稿件）。“同行（Peer）”一词即用于强调这种审稿人和投稿人之间的平等性。与此不同的是，很多学术期刊采用专业审稿人审稿的制度，审稿人往往是领域内名至实归的教授或者科研工作者，一般认为他们在领域内的经验、学识等比一般投稿作者都略高一筹。——译者注

学术圈来决定。这种检阅的机制是必要的，因为总要有一些公认的标准，来过滤掉一些不靠谱的言论。然而，同行评审这种制度本身存在其缺陷，尽管它努力使得评审结果更加客观，审稿人也仍然是人，还是不可避免地有自己的品味和偏见。

因为学术会议通常只接受不超过百分之二十的论文，如果一篇论文给审稿人以不好的第一印象，可能就会被拒收。Dawson和我都不是经验化软件度量领域的专家，所以我们并不能以一种审稿人所期待的方式，去“推销”自己的论文。所以，我们经常因为一些负面的评审意见受到打击，比如：“结果，我发现我就是对作者展示的结果不太信任。我的疑惑来自于两个方面：一方面我不相信他们对度量方法的认知是否合理；另一方面他们也没有使用一些有效的统计方法验证自己的成果。”在发表学术论文这个残酷的世界中，仅仅对一个课题感兴趣远远不足以让自己的论文得以成功发表；想要自己的论文可以发表，就得非常清楚作为论文审稿人的、自己领域中年长的同事们的偏好。简而言之，我们的数据集没有别人的好，技巧没有别人的细致，我们的成果和展示方式也和这一领域中有经验的研究者所期待的形式似乎格格不入。

相比而言，我和Scott、Joel发表的论文就更加成功，因为Scott是HCI领域的专家，并且在这一领域（尤其是我们发表文章的学术会议中）已经发表和评审了很多论文。当然，作为内行人并不能让审稿人在评审我们的论文时，对标准有任何的放松，因为那就太不公平了。但是，Scott仍然可以利用自己的经验，让我们的项目从动机到成果，都以一种审稿人所期待的形式展现出来，从而论文被接受的概率也大大上升。

~

在我博士第二年接近尾声时（2008年6月），我因为仍然缺乏有说服力的成果，又看到在经验化软件度量领域的论文如雨后春笋般地冒出来，开始变得越来越沮丧。我仍然没能将自己的发现发表成论文，也意识到自己和圈内的老手根本没法竞争：因为Dawson和我都不知道审稿人期待看到

的是什么，我开始明白继续在这个领域试图发表论文，只能是一路逆水行舟。又因为发表论文是毕业的先决条件，我必须尽快找到一个新的项目，否则我就不能拿到自己的博士学位了。在我准备在斯坦福开始自己博士生涯的第三年时，我非常绝望，以致不愿放过任何一棵能为我带来论文发表的救命稻草。也就是那时，我选择了回到博士第一年曾经让我痛苦不堪的项目——Klee——当中去。

第三年：噩梦未止

在2008年中，我的博士第三年拉开帷幕时，我作为项目组最年轻的学生，重新回到了Klee项目中。当时，项目里只剩下两个人了——两位Klee的创造者，Dawson和Cristi （Cristi这是已经是他最高年级的博士生了）。Klee项目组的其他成员都已经离开了这个项目。

对于回到项目中来，我的心情很复杂。一方面，我第一年不堪回首的经历让我对项目本身和项目组的工作状态都感到抵触。另一方面，Cristi和Dawson都对这个项目充满激情，他们也都希望发表更多的跟进论文。因为他们在软件缺陷检测这一领域已经身经百战，我觉得和他们合作似乎更有机会能够发表论文。我的另一个选择，则是继续在我第二年开展的经验化软件度量项目上进行研究。虽然我仍然对那个项目很感兴趣，我清楚地知道对于我和Dawson两个外行人来说，在那一领域发表论文并非易事。因为我的目标是发表论文，取得博士学位，我决定收起自己的自尊和自大，并重新投身于Klee项目中。我给Dawson发了一封邮件，说到：“我大体的计划就是与你和Cristi 合作，尽量用Klee做出一些脚踏实地的研究，争取内能投稿几篇论文。我认为对我来说，为学术做出贡献以及每天做出一点令人满意的成果的最佳选择，就是尽可能利用Klee，并和你们的研究兴趣以及目标保持一致。”

~

Cristi和Dawson希望我能实验一种被称为交叉检测的、运行Klee的新方法，这种方法能让Klee找到同一个软件的不同版本之间的差异。在接下来的四个月中（2008 年7 月到10 月），我每天的研磨和我第一年用Klee寻找Linux驱动程序缺陷的任务别无二致，只不过我现在学会了更好地劳逸结合，让自己不至于过度劳累。但就像我第一年的情况一样，我在用Klee做很多繁重的苦工，而实质上并没有用什么方法改进Klee，以让它能找到新的软件缺陷。我每天的工作就是配置好一系列Klee的运行环境，用10个小时运行Klee让它完成对软件的一系列检测，第二天早上再来收集、分析、可视化、并弄明白结果的含义，对Klee的设置进行适当的调整，然后再开始新的一轮10小时的实验。

就像很多复杂的软件工具一样，Klee有很多可以设置的选项。而因为它是一个由很多学生合作完成的用于学术研究的原型系统，它的选项中有很多都没有清晰的文档说明其功能。这就导致我因为误解了一些选项间复杂的相互作用，进而浪费了很多时间在调整这些选项上面。当时我的研究笔记上随处可见这样的脏话："妈的，我觉得我的问题是没发现Klee需要某些调用参数（比如-emit-all-errors），而目标程序也需要一些参数（比如-sym-args），如果把这些搞混了，你也不知道会发生什么，因为Klee在运行目标程序时采用的参数列表和你想想的不一样！"

在那几个月间，Cristi和Dawson时不时地提起他们想提交一篇关于Klee交叉检测的论文，所以我对于这个看似具体的目标充满动力。在我工作的同时，我完成了一篇论文的初稿，并在里面填上了我的实验结果和一些笔记。然而，令我惊讶的是，Christi 和Dawson似乎都不急于完善这篇论文并让它得以发表。我当时还做不到在没有他们帮助的情况下，在这一领域发表一篇像样的论文，因为我还只是一个帮忙完成苦工的研究助理：真正的研究意义以及有说服力的"营销"仍然离不开他们。后来，我们完全没有提交论文，所以我四个月的工作再一次付诸东流，就像我第一年在Klee上的研磨一样。

因为我自己对领域内的技术不够精通，加之缺乏年长的同事的指导，这个项目很快就流产了。尽管Cristi在指导我交叉检测的思路、以及调试Klee的各种奇怪问题时十分耐心，但可以看出他的精力并没有完全投入在这个项目上。因为当时他已经即将完成博士学习了，他的主要精力都用在了申请助理教授职位上。教学职位的申请通常需要花费几个月的时间，投入十足的精力准备，而尽管这样也仍然有很多申请者无法获得职位。每年，每所大学的院系至多会招收一到两名新的终身制导向[17]的新教授，而上百位能力超群的高年级博士生、博士后（博士毕业后暂时留在实验室进行研究的学者）、以及从事研究的学者会竞争这些令人垂涎的空缺。申请学术职位是一种充满压力、耗尽一个人所有时间和精力的过程。所以，Cristi并没有动力花几百个小时的时间再去投稿一篇论文，因为就算论文被接收了，对于他的职位申请而言也已经太晚了。

事后回想，我能明白这个项目因为参与者动机不同注定失败，而当时的我却缺乏预知这种失败的智慧。我当时只是单纯地想要和年长的博士生以及富有经验的教授一起在他们熟知的领域发表论文，才回到了Klee项目中和Cristi、Dawson一起合作。我之所以这么打算，是因为同样的策略在前一年收效显著，我帮助Joel（年长的博士生）和Scott （富有经验的教授）完成了他们的HCI项目，而那个项目最终让我们发表了一篇在一流学术会议中被提名最佳论文奖的论文。

这边的状况又有什么不同呢？简而言之，Cristi和Dawson都没有发表论文的渴望。此前，他们已经一起发表了很多篇Klee相关的论文，而和我一起完成一篇交叉检测的论文，对他们而言是“锦上添花”，但绝非雪中送炭。Cristi已经是博士最后一年了，所以并不需要再发表任何新论文就可以毕业了；Dawson已经获得了全职职位，所以他也不急于发表论文。相比而言，Joel当时正处于博士生涯的中段，急需发表用于博士论文的第一篇论

[17]终身制导向教授（Tenure-track professor）职位是区别于研究导向教授（Research professor）职位的一种称呼，后者和学校的工作合同通常是任意性（at-will）的，相应也不需完成教学的任务。只有终身制导向的助理教授才有机会获得终身职位。—— 译者注

文；而Scott还是助理教授，他需要发表足够多的论文才能获得终身职位。这两次结果完全不同的经历让我明白了，在和可能的合作者合作研究之前，深入了解他们对项目的动力和动机的重要性。

~

因为Cristi在忙于申请教职，交叉检测的项目完全没有进展，我决定自己起头，试图完成一个和Klee相关的项目，而不再继续作为交叉检测的实验助手。和Dawson讨论之后，他建议我试着改进一下Klee的核心组件——它的*搜索算法*。Klee通过在软件的可执行代码中像搜索“迷宫”一样查找软件缺陷，所以改进搜索算法可能能让Klee 找到更多的缺陷。

这是第一次，我开始在Klee上进行创新——改进其搜索算法——而不是为了用Klee找到软件里的缺陷而做苦工。对于衡量我的工作的有效性而言，一种办法是测试Klee在一系列测试软件程序中的*覆盖率*（即测试Klee的搜索算法能覆盖代码迷宫中多大的部分）。Dawson的目标很简单：取得一个比最近一篇Klee论文中报告的结果更高的覆盖率。当时，在89个测试程序上，Klee已经能达到91%的覆盖率了（完美覆盖是100%），而我的任务就是尽可能提升这一比率。每天，我都会尝试修改一下搜索算法，然后在89 个测试程序上运行一遍（大概需要十小时的时间），然后第二天早上再查看覆盖率的数据，然后再改动程序、运行测试，如此往复。

这时已经是我博士第三年的一半了，我和很多同期的同学一样，都陷入了一种“炼狱”般的状态，我们都不太会每天到办公室工作了。我们还因为自己所研磨的问题非常抽象和过分专业而饱受孤独的折磨，因为我们研究的问题可能除了我们之外很少有人理解或者感兴趣。我们的导师和高年级同事会给出一些较高层面的建议，但他们几乎从不坐下来和我们一起仔细研究项目中每一个恼人的细节。

和我们在公司上班的朝九晚五的同学们不同的是，我们没有马上做出任何实物的压力—— 我们没有项目截止日期的催促，也没有中层管理人员

需要取悦。对于系里大部分学生而言，他们偶尔休息一天其实并没有人会在意，所以以此类推，为什么不干脆休息两天、休息一周，甚至休息一个月呢？所以，博士生在第三年左右退出博士项目并不是什么应该令人惊讶的事情。

为了防止拖延症，我不知疲倦地工作，逼自己保持自律，并保持工作日规律地工作。我试着对自己进行“微观管理”，每天给自己制定较小、容易完成的目标来激励自己，期望着最终能收获一个积极的结果。但结果是，因为很难看到自己每天的进展，我很难保持自己在工作中充满动力。

只有自律是不够的——在三个月的对Klee搜索算法的调试之后，我仍然没有得到任何令人振奋的结果。因为Klee在我们的测试程序上已经达到了91%的覆盖率，把这个数字仅仅提高几个百分点，比如到94%，都极其困难。雪上加霜的是，这些微乎其微的进展写在论文里也不会引人注目。我们的论文主旨用一句话来说将是：“我们以某种方式改进了Klee的搜索算法，使得它的平均覆盖率从91%提升到了94%。”这种成果在审稿人眼中很难谈得上令人激动，甚至可能连有趣都谈不上；这只是一种常见的、无趣的在已有的项目上进行的增量式的工作。不出意外，Dawson也并没有兴趣投稿这样的一篇无趣的论文。

如果我当时能用一种更有趣且有效的方法改进Klee的搜索算法，可能Dawson会更感兴趣投稿这样一篇论文。然而在2009年1月，我经过三个月的研磨徒劳无果后，我开始觉得自己日复一日的缓慢努力永远不可能达到一个符合Dawson期待的突破性进展。我不喜欢被称作一个经常放弃的人，但我意识到Klee搜索算法这个项目只有死路一条，所以我还是放弃了。

现在回想当初，我在一个悲剧性的事实中找到了对当时选择的宽慰：在我离开Klee搜索算法的项目之后，Dawson的博士生中有两个人先后对这个问题展开了研究，结果两个人都没能在过去的三年中就这一问题发表任何论文。我并不觉得自己能比那两个学生做的更好，所以如果我继续在那

个课题上努力，可能我也会像这样卡在一个长达三年的炼狱之中。

~

尽管在Klee项目上屡战屡败，我还是想要在它上面做出一点成果，毕竟这是Dawson当时唯一关心的项目。我开始讨厌Klee了，但因为我已经在这个项目上消耗了上千小时的时间，我希望能最终收获一些实质的成果来证明自己的努力没有付诸东流。这时已经是我博士第三年的一半，我非常急于发表一篇第一作者的论文作为我博士论文的一部分；我开始感到自己有些落后于身边的同学们，因为他们中有些人已经发表了自己第一篇可以贡献于博士论文的学术论文。我天真地以为Klee是我通往博士学位道路上“阻力最小的方向”，因为它和我导师的研究兴趣比较贴合。

这时，一个名叫Peter的第一年的博士生加入了Dawson的实验室，并且在寻找研究项目。我和Dawson商量了一下和Peter合作的事情，考虑是两个人一起工作可能能比我一个人更容易做出较好的成果。Dawson对此表示赞同，并建议Peter和我一起在Klee 中重新实现*约束下运行*的功能（简称“Klee-UC”）。读者可能还记得，在我博士第一年时，Dawson和他的另一位博士生一起实现了第一个版本的Klee-UC。他们做出了一份粗略的初稿，投稿了一篇在三天之内粗制滥造出来的论文（我仍然清晰地记得当时的惨痛教训），而在那个学生从博士项目中退学之后不久，项目也就停滞了。所以，两年之后的现在，轮到Peter和我来重新实现这个想法，并尽量让它能够做出足够好的成果发表一篇论文了。

我带着我能鼓起的所有勇气和乐观精神接受了这个新任务，并努力忘掉我和Klee并不愉快的过去。我对自己说，如果我能发表Klee相关的论文，那一定就是这个Klee-UC 的项目没错了。我全心全意地相信Dawson的Klee-UC创意在学术研究的角度看来是新颖有趣的，所以如果Peter和我能够足够努力实现它，并用它找到一些更加重要的软件缺陷，我们就能发表一篇很好的论文。而且，我还可以重复利用我之前在Linux驱

动程序的实验中大部分的程序和设置。然后，我还幻想着一篇关于Klee-UC的很好的论文可以一扫之前我在Klee上几千小时苦工的阴霾，让我扬眉吐气。毕竟，还是我博士第一年用Klee 查找Linux驱动程序缺陷时的经历，让Dawson想到了Klee-UC这个点子。这样的话，如果我能以第一作者身份完成这篇论文，把想法变成现实，也算是一个合情合理的交代（在我所在的领域，尽管有时候研究想法是教授产生的，他们也会让自己的学生作为论文的第一作者）。我甚至期待这个项目会成为我博士论文的第一部分，并从此为我最终博士毕业扫清障碍。

在接下来的两个月之间（2009年2月到3月），Peter和我全心全意地投入到了变成实现Klee-UC之中。我们每天一起在办公室编程，过程十分有趣；这对于很多博士生所过的孤独的生活而言是一个很好的调剂。然而不久之后，Dawson似乎就开始对我们缓慢的进展感到不满。Peter和我都认为我们的进展还算不错，但是Dawson似乎对我们的工作很不满意，所以他很快就打消了写一篇论文赶在下一个截止日期之前提交的打算。

当时，我完全不理解Dawson为什么对我们如此缺乏耐心，但现在我能完全理解他当初的感受了。他当时心中有一个十分清晰的Klee-UC的想法，并且希望几个有天分、并且勤奋努力的研究生能把他的想法变成现实。如果Dawson仍然是一个博士生，他肯定用不了几个月就能让Klee-UC正常工作，然后自己单打独斗就能写出一篇顶级论文。当他还是个学生的时候，他的发表文章的记录就已经非常惊人了，也正是因此，他才能得到像斯坦福这样的顶尖大学的教授职位。然而，现在他需要完成一些教授需要完成的任务，比如教学、论文/程序委员会工作、论文审稿等等，所以他不能再像之前一样投入上千小时，专注于把一个想法变成一篇可以发表的论文了。就像所有劳力密集型的研究领域的教授一样，Dawson需要一些学生的帮助才能把他的想法变成现实。

我认为Dawson期望Peter和我能比实际进展更快得到能用于论文发表的成果，所以对于他来说，我们似乎对于给我们的任务不是能力不足，就是不够用心。作为一个顶尖大学的教授而言，Dawson所有的学生恐怕都不

如博士时代的他自己有能力，这着实是一个不幸的事实。而这一事实的原因很简单：在顶尖大学的博士毕业生中，大概75个学生中只有一个能有幸在斯坦福这样的大学任教（而对于普通大学的博士毕业生中这个数字可能是200比1）。所以其实并不出乎意料的是，Peter和我都不是这样出类拔萃的人才。如果Dawson能和他自己的年轻版的克隆人一起研究，估计研究进展会快很多！

尽管Peter和我在那两个月中在这个项目中投入了很多精力，但我们能清楚地意识到Dawson对我们并不满意。Peter对此感到很失望，于是他之后更换了导师，并最终从博士项目中退学了。我的合作者离开之后，我对Klee的最后一丝幻想也随之破灭，所以我也很快离开了Klee项目，而且再也没回来过。

~

Peter和我离开Klee项目的两年之后，Dawson终于找到了一个博士生，可以帮助他把他Klee-UC的想法最终变成现实。2011年，Dawson和这位新学生发表了一篇非常优秀的论文，其中包含了Klee-UC和交叉检测的思想。最终，通过四个博士生长达五年的努力，Dawson的Klee-UC的想法终于变成了一个可以发表论文的研究项目。在这四个学生中，只有一个“幸存”了下来——我离开了Klee项目组，另外两个博士生则直接退学了。从每个学生的角度看来，成功的概率实在是过分渺茫了。

然而，从一个教授的角度来说，Klee-UC是一个令人激动的成功！因为Dawson已经获得了终身职位，所以他的工作并没有什么风险。事实上，终身教授制的主要意义之一就在于，获得终身教职之后，教授们就可以有机会尝试更加大胆的项目思路。不过，这个制度的阴暗面就是，往往教授们会让学生去风险较大的项目上研磨，而这些项目的成功率往往并不高。学生也并不能拒绝教授的要求，因为他们都是由导师的科研经费支持的。幸运的是，我有外源奖学金的赞助，所以对我来说离开Klee项目要容易很多。

我并无意专门把Dawson和Klee拿出来，作为批评的典型。在大多数劳力密集型的科学与工程学的研究领域中，这种在终身教授和博士生之间学术动机的鸿沟都是普遍存在的。通常的情况都是，首先，教授获得一些研究经费，并产生一系列高瞻远瞩的研究思路（比如Klee-UC或者交叉检测）。然后，教授用研究经费将一些学生招入麾下，指导他们实现自己的研究思路，通常（但并不一定）作为这些学生的毕业论文的主要工作。离开了这些上千小时的学生的工作时间，这些研究思路永远不可能转化为现实的成果，更遑论发表论文了。

有时，教授们可能需要经历好几届的学生在项目上的失败甚至退学，直到某一届学生最终将项目引向成功。这一过程有时需要两年，有时需要五年，有时甚至需要十年的时间。很多项目常常持续的时间都超过了博士生的“生命周期”。但只要最终这些项目的想法得到实现并形成论文得以发表，项目就算是成功了。教授对此感到很满意，院系也感到很满意，出资资助研究项目的机构也表示满意，最终将项目引向成功的学生也会感到满意。但在这之前，一路上“伤亡”的学生呢？终身教授通常可以接受几年的损失在临时的失败上，但对一个博士生来说，这一系列的失望可能导致的是学术事业夭折在萌芽中，甚至心理健康都可能岌岌可危。

~

2009年5月，在我博士第三年接近尾声的时候，我参加了Cristi的论文答辩。论文答辩是博士生获得博士学位之前最终的“神圣仪式”：博士生就自己博士期间和博士论文相关的研究做出一个一小时的口头汇报，并需要回答一个教授小组提出的尖锐的问题。Dawson通常都很安静和内敛，而这次当他向观众介绍Cristi、并激动地说到和他一起合作创造出Klee是一段令人愉快的时光时，他脸上的骄傲和自豪清晰可见。他对Cristi的夸奖并非空穴来风：Cristi在博士阶段的工作无可指摘，而且他在Klee 中发展出的思想，创立了软件缺陷检测领域中的一个新的子研究领域（称作混合式实际/符号化程序运行）。

看过Cristi的论文答辩展示之后，我更清晰地意识到自己几乎不可能从Klee中继续做出一整个博士论文的工作了，这也进一步增强了我离开Klee项目的决心。Cristi闪耀的成功，让Dawson年轻的学生们发表论文和最终毕业变得更加困难了。从无到有的Klee的创造的工作已经完成；所剩下的大多是在此基础上一些添加和改进，或是将Klee应用在一些新的软件上，比如Linux驱动程序软件。尽管这些工作也可以最终成为可以发表的论文，甚至成为博士论文的一部分，Dawson已经不像其它刚刚进入我们研究领域的竞争者一样急于发表论文了。

因为Klee（以及一些2005年到2008年的相关项目）创建了一个全新的子领域，很多助理教授和年轻的学者很快就加入到了这个大潮中，并迅速发表了很多篇论文描述针对这些项目的改进，以期获得终身教职或者职位晋升。这就像一个学术圈的淘金潮，而Cristi、Dawson以及其它一些少数的先驱者引领了这个潮流。因为Dawson已经获得了终身教职，并创造了包括Klee在内的很多著名的项目，他并不需要随波逐流，也没有强烈的欲望发表很多论文来填满自己的简历了。

这一现象的结果是，和那些年轻的研究者一起开展研究的博士生更容易发表论文并去的博士学位，而与此同时Dawson的学生似乎都没那么顺利。在我离开Klee项目后的三年中，来自世界各地的很多研究小组发表了上百篇和Klee思路类似的论文。令人欣慰的是，其中有15篇论文描述了对Klee本身的改进——作为我们开放Klee源代码，以促进这方面研究进展的结果。同时，Dawson的博士生中有五位在Klee上下过苦工；而到目前为止，只有一位成功地发表了一篇论文（即Klee-UC）。

让然觉得讽刺和不幸的是，现在作为学术会议程序委员会成员和论文审稿人的都是Dawson的直接竞争者，尽管他参与创立了这一研究领域，他自己的论文也更难以得到发表了。因为Dawson已经有几年没有积极发表论文了，他已经对于发表论文所需要的修辞技巧、新诞生的术语、以及符合审稿人口味的一些“营销技巧”有些生疏，所以在顶级会议发表论文已经变得困难了。此外，他的竞争者们发表越多的文章，审稿人对论文的要

求也就越加苛刻，他和他的学生在发表论文时也就越需要说明自己研究思路的与这些论文的区别与联系，最终导致他们的论文更容易被拒收，收获更多令人失望的结果。这很讽刺，因为归根结底，如果不是因为十年前Dawson的远见，这些挑剔的审稿人根本不可能在这一领域开展研究，更别说拒收他的论文了！

我考虑了一下，留在Klee项目中唯一的优势就只剩下Dawson发自内心热爱这个项目了。尽管可能我最后不能在这个项目上发表论文，也许他也能最终给我一个“怜悯毕业”。但因为我之前在Klee上的经历并不顺利，我没有勇气在Klee上再研磨不知道多少年的时间，仅仅指望一个“怜悯毕业”的结果。

这时，我已经度过了我博士生涯三年的时光，但仍然对我最终要怎么完成一篇博士论文毫无概念。我对未来没有什么计划，但我清楚地知道，自己肯定不可能再回到Klee 项目中来了。

插曲

我博士第三年结束后，我马上就动身去了华盛顿州的西雅图，去哪里的微软研究院总部实习。那个夏天成了我生命中最有趣和最有成效的夏天之一：我实习时参加的研究项目让我发表了三篇顶级论文，而且更重要的是，这个项目为我的博士论文奠定了基石。

现在，微软研究院已经是一个进行高水平学术研究的企业研究所了。在更多的公司中，研究所往往将精力投入在研发工程中，以期改善公司即将推出的产品。然而，微软研究院（简称MSR）的主要任务则是进行科学与工程方面的基础科研，并以在相关领域发表顶尖学术论文作为目标。

如果做比喻的话，MSR可能最像是一个没有学生的、巨大的研究型大学。全职的研究员就像这里的教授，而区别是他们并没有教学或者指导学生的义务，所以他们可以将自己的几乎全部精力投入到研究当中。可能他们最享受的职业优势，则是不需要去申请科研经费，而对教授们来说这是一种周而复始、极其消耗时间精力的工作。因为微软已经是一个利润可观的公司，所以它可以每年将成百上千万美元的盈利用于支持学术研究（发表论文）。微软的打算是，由其麾下的研究员研发的一些知识产权可能作为新产品灵感的来源，而且它也希望能拥有计算机科学界最顶尖的研究者能作为自己的顾问。正因为如此，微软公司给予了研究者们所有他们所需的资源，让他们完成最好的学术研究。

在MSR获得全职研究员的难度不亚于在一个知名大学获得教职的难度。尽管技术上来说MSR的研究员们并没有终身职位，他们的工作保障很优越，只要他们能保证不断发表学术论文。因为在计算机科学领域，很多研究领域都是劳力密集型的，所以研究员们经常招募博士生作为暑期实习生，帮他们完成自己的研究计划。这对于双方来说都是个不错的交易：研究员们可以请博士生们来帮助处理一些研究中的苦工，而博士生们同时也能有机会和学校以外的著名研究者一起，发表顶级的学术论文，甚至之后得到工作的推荐信。在过去的十年间，计算机科学领域顶级学术会议的论文中，有相当一部分都是由MSR 的研究员和他们的实习生完成的。

当初夏我抵达MSR总部时，整个地方都充满了活力，有上百位博士生在和他们的经理会面，并准备开展工作。因为我们只能在那里实习三个月，我们的经历都为我们精心策划了研究项目，以期在实习结束时能发表论文。我们中的大部分人都能用自己暑期实习的工作投稿一篇论文，而其中一部分最终能够得到发表。当然，学术研究本身是有风险的，所以也有一些实习生被分配到的项目并不能产生可以发表的论文。尽管如此，几乎每个人都很享受在这里的时间——我们的工资比平时在学校的津贴几乎翻了两番，微软还会组织有趣的出游活动增进大家的交流，我们还有机会聆听由一流的学者做出的激动人心的学术讲座。

可能MSR实习带来的最深远的影响，就是我们在实习中形成的友谊。在那个夏天，我有幸结识了一些我同辈中最聪明、最充满灵感的年轻的计算机科学家。比如，和我同一个办公室的三位实习生之一的一位女生即将开始她在MIT的博士项目，而她在本科学习期间发表的顶级论文可能已经超过了大多数博士生在博士期间所能发表的数量。另外一位是来自加州大学伯克利分校的博士生，在工作日白天的实习工作之外，他还将每天晚上和周末都用于有全国各地的合作者参与的另一个研究项目。他们都很可能将来成为获得诸多荣誉的教授、研究领袖、或是高科技行业的创业新秀，所以我感到非常荣幸能和这些人共事了一个夏天。

~

关于我如何获得进入MSR实习的机会，这个过程则是一个对充分结合自己的研究成果、并积极发展专业内的人际关系，其重要性的明证。很多博士生都会通过某种人际关系获得实习机会（以及之后的全职工作），而我也不例外。

在我博士第二年，和Scott、Joel在他们的HCI项目上开展研究时，就申请过一次MSR 的实习。我通过常规渠道申请，在网上递交了自己的简历，但我的申请很快就被拒绝了，因为实习岗位被发表论文更多、且通常有更多内部人际关系的博士生们占去了。

一年之后，在我博士第三年，一位MSR的研究员注意到了我和Scott、Joel合作的HCI 项目（论文发表在一个该领域的学术会议上），所以他给我发了一封电子邮件，问我是否有兴趣在一个大致相关的项目上，跟着他做实习。他专门找到了我，是因为在我MIT本科的时候，我的研究导师曾经将我介绍给他过，所以他对上了号。

我对他的邀请深感荣幸，但我回复了他我已经不在HCI方向做研究了；那时，我已经回到了Klee项目中，继续进行着软件缺陷检测的研究。不过，我表达了自己对去MSR 开展经验化软件质量度量课题的强烈兴趣，因为我博士第二年都在和Dawson做这类的工作。他马上就把我的邮件转发给了他的同事Tom，而Tom是经验化软件度量领域正在冉冉升起的新星。通过邮件简单地自我介绍之后，我将自己和Dawson撰写的关于Linux缺陷报告度量的论文发给了Tom。Tom很喜欢我的论文，于是便决定招聘我作为他的实习生。在我博士第二年中我曾经读过好几篇Tom的学术论文，所以我对于他招募我为他工作，感到十分激动。

如果我只是像上百位其他申请者一样，盲目地在网上递交自己的申请，可能我永远也没机会吸引Tom的注意。很多其他的实习生也是通过某种人际关系得到实习职位的，不过对他们而言，可能是导师直接将他们推荐给了在MSR研究方向相关的学者。有趣的是，不是Dawson，而是我本科一位研究导师（我们的项目大概是在六年前）为我提供了这种我当时亟需

的人际关系。这位导师后来还为我的第一份全职工作，介绍了一位关键的人[18]。从这个经历，我明白了被有影响力的人推荐的重要性；在一个充满挑战的领域，仅仅做得好是远远不够的。

~

Tom在较高层次上为我的实习研究制定了计划，并鼓励我完成一个可以实现但雄心勃勃的目标——在夏天结束之前投稿一篇顶级学术会议的论文。我的项目是使用量化的方法研究在软件缺陷报告生成之后，缺陷被修复，被分配给其它开发者，或者被认为已经修复、然而后来发现还有问题等，这些现象中背后的人为因素。为了深入研究这个问题，我需要写程序分析软件缺陷数据库，并对比微软内部的员工个人信息。我当时非常擅长做这类的数据挖掘和分析的工作，因为我整个博士第二年都在和Dawson一起，就Linux 的缺陷报告和版本控制历史数据做着类似的分析。

每天下午五点，Tom在下班之前都会来看看我的进展。尽管每天的进度检查可能让人压力很大，但实际上我发现这些进度检查非常有帮助，以为Tom态度和蔼，对我的工作也并不抱有批判的态度。每天都能得到及时的反馈，让我能保证专心在工作上，也更有工作的动力。这种确定、短期的目标和持续、有效的反馈的组合，使得我实习期间的工作日比之前三年博士生涯中任何一段时间都更加高效。最让我开心的是，我每天只在正常的工作时间工作（早九点到晚六点）。我并不能将工作带回住处，以为我需要的数据都只能从微软内部才能访问，所以我每天晚上都有机会出去放松，而不需要担心自己是不是要做更多的工作；而在学校时，我常常在担心自己是不是工作得够多，因为几乎醒着的每一分每一秒我都能工作。

由于Tom发表过也评审过很多软件度量方向的论文，他是绝对的“局内人”，他知道什么样的结果和写作方式在这一子领域更受欢迎。在夏天结

[18]原文是“provide a crucial introduction”，也可能是指介绍了一个机构/组织等，但从上下文无从推断。这里译者采用了更符合中文语言习惯的翻译。——译者注

束、我们提交论文的时候，Tom 非常熟练地在相关研究的基础之上说明了我们研究工作的贡献，阐明了我们论文的结果新颖、有影响力的原因，并尽量完善了论文的叙述。三个月后，我就听说了我们研究软件缺陷修复的论文被一个顶级会议收录的好消息，而那年这个会议的收录率只有14%。

但Tom没有就此收手！因为他刚刚入职MSR不久，他非常渴望能够发表更多的跟进论文，以建立自己在公司的名望。在接下来的几年中，我们用我2009年暑期实习时得到的实验结果又发表了两篇顶级会议的论文，一篇是关于缺陷报告负责人变更的，另一篇是关于缺陷报告重新开放的（这篇论文获得了最佳论文奖）。

~

我在MSR和Tom开展经验化软件度量研究所收获的成功（发表了三篇顶级论文），对我来说是一个令人满意的拯救——将我救出了博士第二年在这一领域失败的低谷（投稿了两篇论文被拒，最后只发表了一篇次级会议的短文）。因为我在这两次截然不同的经验之间并不是变得聪明了很多，我将我实习项目的成功归功于两个因素：微软和Tom。

首先，作为一个MSR的实习生，我可以访问到包含微软软件缺陷和人员资料在内的大量内部数据。如果是作为外部人员，我是永远不可能获得这些数据的访问权限的。MSR所拥有的庞大的数据资源使得他们的研究员们（比如Tom）可以更快地做出开辟性的研究，而他们的竞争者因为没有相应的数据，可能永远都做不到这一点。相比而言，当我和Dawson 一起做研究的时候，我能获得的Linux数据集规模小了很多，质量也差强人意，因为开源软件项目往往不如一个世界一流的软件公司在维护数据记录上更为用心。

此外，Tom在其中的贡献也不可或缺：因为他在经验化软件度量领域是老手，所以他知道应该如何作为一个技术导师来指导我，他还清楚论文需要怎样的精心调整才能尽可能最大化被接收的概率。与此相反的

是，Dawson在这一领域只是一位充满研究热情的局外人，所以对他而言，他并没有在这一领域开展研究的动力，也不能良好地指导我完成研究项目（尽管他在另一领域——软件缺陷检测——是世界知名的学者）。

在我博士第二年时，因为Dawson和我在经验化软件度量这一领域发表论文困难重重而倍感煎熬，因为我们的竞争对手都是这一领域的专家和老手。现在当我终于和这些专家之一一起发表论文，站在了成功的这一边时，这种感觉令我由衷地欣喜。

~

尽管我在暑期经历了一段美妙的“插曲”，当秋天回到斯坦福时，我仍然对博士论文的项目毫无头绪。我只知道我不想再Klee项目上继续消耗下去了，但我对于什么样的研究工作使我感兴趣——而且更重要的是，可以发表论文——仍然一无所知。

我考虑过将我在实习做的研究工作扩展成我的毕业论文。然而，我最终得出的结论是，我回到斯坦福之后就很难在这个问题上发表更多论文了，因为那样我就无法访问微软的数据了。最好的打算似乎是，我能在微软完成自己博士论文的所有相关工作。然而这显然可能性不大，因为我没听说过之前有学生这样做过。

作为最后的努力，我尝试联系了我在谷歌实习时的经理，以寻求重新回到谷歌实习，利用他们关于软件缺陷的内部数据集继续经验化软件度量的研究。他似乎对此没什么意见，但我后来没有继续跟进这个计划：因为他自己也不是学术研究者，而除了他之外，我似乎也想不到谁会愿意帮我完成这个研究计划。所以，我最终决定放弃经验化软件度量这个方向，所以在MSR发表的三篇论文最终也没有为我博士毕业做出贡献。然而，这些经历对于我提升研究水平和论文写作水平，无疑产生了积极的影响。

因为求毕业心切，我在暑期实习期间，将很多天晚上和周末用于阅读学术论文，或者是在咖啡店展开头脑风暴。在某一阶段，我甚至产生了一

个想法，那就是创造一个和Klee 类似的博士论文项目，但不使用到Klee这个软件本身。这个计划可以一方面让我远离Klee的深渊，而另一方面也保持Dawson对我研究项目的兴趣。不幸的是，我没能从已经发表的论文中找到我自己原创的、有足够创新性的思路。

接着，2009年7月24日，就在我的暑期实习刚刚过半的时候，灵感突然降临了。在MSR 的办公室编写程序分析数据的时候，我开始想到了一个想法，而这个想法最终转化为了组成我博士论文的第一个项目。我发狂似地写下了一页接一页的笔记，并让我的朋友Robert帮我确认我的想法不是异想天开。当时，我完全没有预料到这个想法会不会被更广泛的学术圈认可，但对我而言，这至少是我回到斯坦福、开始我博士第四年时，一个明确的前进方向。

第四年：卷土重来

在我整个博士第二年、和2009年在MSR暑期实习期间进行经验化软件度量研究的过程中，我一直在使用一种叫Python的非常流行的编程语言[19]，进行数据处理、分析、和可视化。具体来说，为了我的程序可以运行，我需要在我的电脑上存储很多程序所需的数据；为了我的程序在数据分析过程中，经过每一次很小的改动之后可以运行得更快，我不得不把我的程序改得过分复杂。当这种想法在我的头脑中慢慢地酝酿一个夏天之后，在七月的一个安静的下午，我突然萌生了这样一个想法：如果我能以某种创新的方式修改Python 的运行环境（术语称为*解释器*），也许我就能消除我每天面对的很多低效的工作，从而提高使用Python进行计算的研究者的工作效率。我将我对Python解释器的改进想法称为IncPy，这来自于增量式Python（**Inc**remental **Py**thon）的缩写。

~

就像所有务实的研究者一样，我想到IncPy想法之后的第一件事（其实是从一开始的激动平静下来之后的第一件事），就是仔细地在网上搜索，试图确定是不是有人曾经做过相关研究。谢天谢地，没人曾经做出过我计划完成的工作，但的确有一些之前的研究项目和我的想法有点接近。不过这不是大问题；没有什么研究创意完全是原创的，所以学术界永远存在着

[19]Python是一种解释型的编程语言，语法灵活且简洁明了，非常适合用于编写需要很快完成的数据处理、分析程序，也因此被很多研究者、数据分析师等采用。

很多相似、相关的研究项目。不过，如果想最终发表论文，我需要令人信服地说明IncPy和这些已经存在的项目有何不同。在几天之内，我就简单地规划出了一个初步的项目计划，包括一些说明IncPy为什么独特、新颖、值得研究的论述。

因为我还有一个多月才离开MSR，我充分利用了我周围的环境——有很多聪明的同事都在像我一样，为他们的研究编写着类似的数据处理程序。我和几位同事一起喝咖啡谈过话，我简单地问了问他们在工作中进行数据分析时的习惯，以及他们工作中遇到的低效问题。我还向他们简单阐述了自己的IncPy想法，并和他们讨论了如果要让这个想法更加实用、以及更加适合作为学术研究问题，项目的哪些方面还需要微调。这些早期的谈话增进了我的信心，让我知道我正前进在正确的方向上，因为其他人和我一样对数据分析中的低效感到失望，并且渴望看到一个IncPy这样的解决方案。

那个夏天接下来的时间里，我将晚上和周末都用于在咖啡店里不断完善我逐渐成型的IncPy想法，强化它的“营销宣传”，并且从我的MSR同事们中获得反馈。我还将我的想法的初稿发给了Dawson，但那时我并不太关心他对这个想法感不感兴趣，因为这将会成为我自己的研究项目。我不是在征求他的同意；我只是想告诉他这是我秋天回到斯坦福之后想要开展的研究工作。

~

我的博士生涯在我在斯坦福的第四年重新启程，我挥别与过去三年的联系并翻开新的一页。不再在已建立的研究项目上开展工作，不再去尝试制定计划使得自己与教授和高年级学生的研究兴趣相一致，也不再去担心什么样的研究才是学术圈乐于看到的。我现在拼命地只想把我IncPy的新想法实现出来，形成一篇可发表的文章，并使之成为我毕业论文的第一部分。

虽然充满了新的热情与激情，我还是有点忧惧的，因为我是脱离了原有的工作来做一件新的事，并且还未得到教授的支持，不知道是否会成功。我们系大多数博士生都会做他们的导师或者教授感兴趣的项目，毕竟有了教授的帮助和专业知识会更容易发表论文。然而，即使没有教授撑腰，我还是预感IncPy会成为一个能发表出来的想法；在过去三年中的研究失败中我积累了不少经验教训，从而对哪些想法可能会成功有了一个更好的直觉。对这种直觉的坚信不疑成为了我博士生涯的转折点。

还有一点比较务实的事是，我依然保持Dawson作为我的导师，因为他同意我做自己的项目而且还能时不时给我的工作一点高层面的回馈。由于我仍然有外源奖学金来作自我基金支持，我也就没有义务像Dawson的其他学生那样必须继续在Klee项目中工作。我和Dawson的关系在我博士的接下来几年变得有些远了。我们碰面讨论项目可能总共才十余次；我大多时候还是自己工作或者寻找其他合作者。但我却不想正式地更换导师，因为就算那样的话，我到了一个新的研究组里还是得再一次“交学费”。况且，我也不知道系里还有哪个教授对我IncPy的想法感兴趣，不然的话我可能还是会认真考虑一下换导师的事情的。

为了最终能够毕业，我需要找到三位同意阅读并且会批准通过我的毕业论文的教授，来组成一个博士论文委员会。大多数学生会直接请导师帮他们选择这个委员会的成员，毕竟他们一直在做导师指定的“官方”项目。但由于我单飞了，没有和Dawson在Klee项目上开展工作，我就不能享有这个特权了。我努力地寻找我们系或者其他相关院系（如统计系，生物信息学系）的教授来做我的委员会成员。我给一些教授发了冷邮件，并试图通过他们现在带的学生联系到他们。我甚至还做了一个幻灯片准备向这些潜在的委员会成员展示我毕业论文的初步想法，但不幸的是，没有教授有兴趣和我面谈。然而，我没有选择放弃并回到有院系教授支持的传统项目上去，而是冒险在IncPy的征途上继续前进，并且觉得委员会成员的事一定会车到山前必有路。

~

受第二年时与Scott和Joel在人机交互工作上的启发，我2009年秋天一回到斯坦福，就开始问了一些写Python程序来在研究中分析数据的同学。我的目标是发现他们与编程相关的低效问题是什么，以及如何能用IncPy消除这些低效问题。我的一些朋友还帮我在他们的实验室组会中安排了一个报告，让我能介绍当时还只是半成品的关于IncPy的想法。在初始阶段的这些采访和汇报，对我新想法的提出和“营销技巧”的提升很有帮助。我也特别感谢那些当我除了几页粗制滥造的幻灯片而外别无其他东西展示时，还能帮助我让我的项目起步的朋友们。

我越来越感到振奋，因为我发现很多基于计算的领域，例如机器学习、制药学、生物工程、生物信息学、神经科学，还有海洋工程，都在使用相似的方法进行数据分析，并且能从类似IncPy这样的工具中受益。经过了几个星期的调研采访和后续的计划细化以后，我自信我有了足够的卖点来令人信服地“推销”出这个项目，形成一篇论文投稿出去。我想下的论点是：各种领域的众多计算型研究者每天都在与编程工作流中常见的低效问题抗争，而IncPy正好提供了针对这些低效问题的一个完全自动的解决方案，这是之前没有人做到的。这个初期形成的卖点最终也演化成了我整个博士论文的主题。

有了一个总体的策略之后，我准备好了开始一个长达几千小时的辛苦工作——研磨——这也是把IncPy从想法变成现实的原型工具所必经的工作。在那个夏天快要结束的时候，我扮演了一个“教授的角色”，草拟出了高层面的设计提纲，作了报告，细化了概念性的想法。现在我准备好了转成“学生的角色”，在接下来的一年经过辛苦的研磨从而实现出IncPy。

~

这时我也清楚有了一个不错的想法只是做出可发表的研究成果的必要条件，而绝非充分条件。年轻的研究者——通常是博士生——必须在这里那里花费成百上千小时，在细节处挥洒汗水，最终才能让想法结出果实。在计算机科学的研究中，苦力劳动最主要的形式就是编写计算机程序来搭

建、测试和评估新的基于软件的原型工具和技术。我在过去十年中花了近万小时在各种形式的编程工作上，这些工作有的在课堂上、有的在业余生活中、有的在实验室里，还有的在工业界的实习中，这些让我对实现一个类似IncPy这样的研究创意所必须的、高强度的、复杂的编程要求有了准备。

实现IncPy包含了一些我从未完成过的编程中的脏活累活。如果我在过去十年没有经历过这么多小时的如火炼真金般的训练，那我也绝不会去尝试这么一个劳动密集型的项目。不用怀疑，肯定也有其他人已经发现了我所注意到的计算型研究工作流中的这些低效问题，他们本要成为我的竞争者，但将我与他们区分开来的要素是，他们没有创造出像IncPy这样完全自动的解决方案所需要的深厚编程功底。他们可能最多也就能做出一个半自动的解决方案而已，而实质上这样的工作还不足以在优秀的会议上发表论文。

虽然我不太喜欢我研究生初期和本科期间的研究项目，但我从这些经历中获得的技术上的技巧以及判断力，使得我现在能将自己真正关心的想法实现出来变得可能。在接下来的三年中（2009年到2011年），我不知疲倦地做出了五款帮助计算型研究者的新原型工具（IncPy是其中的第一个），发表了多篇[20]介绍这些工具的第一作者论文，接着将这些工作组合在一起形成了一篇我深感自豪的博士论文。那三年——我博生阶段的后半程——是我迄今为止最具创造力和最多产的一段时期，与我博生阶段的前半程形成了鲜明的对比。

大量生产的愤怒让我疯狂地主动工作。我变得刚毅、果决、高度专注。每当我回想起我前三年博士生活所忍受的低效、失败、沮丧所带来的煎熬时，我心中就会产生一团怒火，然后逼迫自己工作得更努力；我被这种强迫性的鞭策所驱动，来弥补假想的浪费掉的时间。诚然，早些年的光阴实际上并没有浪费；没有那些挣扎，我也不会获得创造出组成我博士论

[20]原文是“published one or more first-author papers"，这里没有直译，而是根据上下文翻译成了“多篇”。—— 译者注

文的这五个项目的灵感和能力。

~

然而，2009年9月，在我开始我第四年博士生涯的时候，我还不知道有什么会在未来等着我，也绝没有像五个项目、230页的毕业论文这样的宏伟计划。我甚至都不知道有没有教授会认真地看待我未经检验的想法，并同意担任我的论文评委。我想做的就只是将IncPy 实现出来，尽可能让其发表，然后到时候再来考虑接下来一步做什么。

当我正准备开始实现（编写程序）IncPy的时候，一个出人意料的好运从天而降。当时，我还不知道这么一件小事竟能带来一系列的好运，为我铺开了一条通往毕业的大道。有一天午饭的时候，我的一个朋友Robert给我说他正计划把他一个新研究项目的论文投稿到一个研讨会[21]，这个研讨会的论文截止日期是两个半月之后。

按常理来说，我对这件事应该不会有太多想法，有两个原因：首先，Robert的研究主题（一个叫做*数据来源*的子领域）和IncPy并不相关，所以他准备投稿其实和我没有一点关系。第二，在我们系，一篇发表的*研讨会论文*对于博士论文来说并不“作数”。研讨会论文的目的是宣传初期的研究想法，它并不像会议论文那样要经过严格的审稿。会议论文的接收率一般在8%到30%，而研讨会论文的接收率却有60% 到80%。我们系的大多数教授都不主张学生往研讨会投稿，因为论文一旦被接收（这也是很容易的一件事），教授就得从他们的经费中拿出大约1500美元，来承担学生参加研讨会并作报告产生的车旅费、住宿费和注册费。送一个学生去参加研讨会的费用已经几乎和参加一个会议的费用相当了，但发表会议论文对学生和教授来说却又更有声望。因此，顶级的计算机科学教授会强烈建议学生发表更多的优质会议的论文，同时也要避免投稿到研讨会。

[21]研讨会:相比起学术会议（conference），研讨会（workshop）的规模较小，更倾向于集中讨论某一个特定的话题。——译者注

我问了问Robert为什么他的导师会建议他把初期的想法投稿到一个研讨会，而不是再深入挖掘一下，日后再投稿到一个学术会议。他说部分原因也是出于方便，因为这个研讨会就在圣何塞举行。他们整个组都准备去参加这个研讨会，圣何塞离斯坦福也就20英里的路程，所以他也想在那做一个论文报告。

出于好奇，我浏览了一下那个研讨会的主页，看一看组织者们对哪些话题感兴趣。虽然这是一个关于数据来源（Robert的研究领域）的研讨会，它的话题列表里却包含了这么一点："高效/增量式再计算"。我心里暗想：嗯，IncPy为Python程序员提供了高效的增量式再计算，所以如果我以合适的方式推广我的论文，也许还正好契合这个研讨会！由于我不需要长距离奔波就能参加这个会，所以如果我的论文被接收，我也就不用忐忑地去找Dawson让他帮我报销注册费了。我发邮件给Dawson告诉他我准备往这个研讨会投稿的计划，他不冷不热地给我回复了我。不出所料，他对研讨会提不起多大的兴趣，但是觉得我要想投稿过去也可以，因为他很敬重这个研讨会的程序委员会的联合主席，一名名叫Margo的哈佛教授（他后来也扮演了一个至关重要的角色，帮助我得以毕业）。

时间也正合适：我还有两个半月来加班加点地实现出第一个能够工作的IncPy原型,然后写一篇研讨会论文并投稿。由于我知道研讨会的接收门槛比会议论文的低很多，所以我的压力也没那么大。我只需要让一个基本的IncPy原型能够合理地运行，并且生动地讨论一下这个经验就行了。我发现这种找到并为自己设定短期目标的策略有着奇效，也让我能够在我后来的博士生涯中一直保持专注。没有自我设定的截止日期，很容易就会陷入倦怠期并且患上拖延症。

我的论文最终被接收了，还获得了不错的审稿意见，我在2010年2月参加了那个研讨会并做了个半小时的报告。值得一提的是，我的论文还很荣幸地得到了程序委员会联合主席Margo 的赞扬，他还点出了我的论文是如何与他的学生Elaine正起步的一个基于Python 的新项目产生关联的。Elaine没有来参加这次研讨会，不过Margo给了我Elaine 的邮件地址并

建议我们联系一下。

起初，我担心Elaine会成为我的直接竞争对手，还可能在我之前发表会议论文，这会使得我发表一篇关于IncPy的会议论文变得更为困难：因为在学术界成为第一人是很被珍视的，一旦其他研究者在你之前发表了一篇类似你的想法的论文，你再想把你的想法发表出来就会变得更难了。但是经过几次试探性的邮件接触和视频聊天之后，我发现她并没有把这个工作作为会议论文投稿的打算，这让我放松了一些。而且，她的工具也没有全自动的能力，而这正是把IncPy和其他类似工作区分开来的特点。当我们意识到彼此不是竞争对手后，我们立马成为了朋友。我很高兴在接下来的几年里，我还和Elane保持了联系。后来我和Margo重新取得联系，让我能够奔向博士征途的终点，部分也要归功与Elane。

~

即使做这样一个有关IncPy的研讨会论文报告有利于获得反馈，尤其还能和Margo会面，但那篇论文对我的博士论文而言并不算做是“真正”发表的论文。我知道我仍然还需要将这个工作发表在我们系教授所认可的会议上。研讨会论文和会议论文最大的区别就是一篇会议论文必须有一个有说服力的*实验评估*来体现论文中所描述的工具或者技术的高效性。一篇论文的评估章节可以是多种形式，包括从对运行时间的测量，到用户行为的控制性实验研究。由于很多研究者都提出了相似的想法，所以审稿人会仔细地审查这些想法是如何被实现出来的，并实验性地评估它们，从而决定哪些论文可以接收，哪些论文应该拒绝。

虽然在IncPy项目开始时，我就知道展示一个令人信服的实验评估是很困难的，因为我想说明的论点——IncPy可以提高计算型研究者的生产力——是一个主观且模糊的概念。在读了一些同样也提出生产力的提升这一论点的论文之后，我设计了一个可分为两部分的评估策略：

1. **案例研究**：从各种计算型研究者处收集一些用Python语言编写的程

序，并进行模拟，看看如果研究者们采用IncPy而不是常规的Python后，生产力可以提升到什么程度。

2. **部署：**找一些研究者来让他们在日常工作中使用IncPy而不是常规的Python，之后再让他们报告是否IncPy提升了他们的生产力。

下一个相关会议的论文提交截止日期在七个月之后，我计划在那之前准备好论文。我花了很多时间来寻找用于案例研究的程序样本和用于部署的潜在用户。没有这些，就没有评估，也就没有会议论文，也没有博士论文，更别说毕业了。（当然，我还是把我大部分醒着的时间花在了恼人的写代码、调试，以及测试上，从而来实现IncPy。）

我学着扮演一个一半收购者一半乞求者的角色，坚持不懈地询问同学是否有Python程序可以拿来给我用作案例研究，或者更好的是，是否愿意安装并使用IncPy，以此作为日常工作的基础，并向我提供他们的使用感受。如我所料，我得到的回答多半是否定的，但我还是礼貌地询问了他们能否给推荐其他一些我可以联系联系的人。我也到许多研究组的组会上做了不少报告来招徕大家对IncPy的兴趣。历经了几个月的"乞求"之后，我获得了来自不同领域的近十个研究者的Python程序，足够我进行案例研究了。我感激当时助我脱离困境的每一个人，他们除了来自一个不相识博士生的好话外，没有得到任何东西。

~

虽然案例研究对我论文中评估的那一个章节来说已经足够了，但我真心希望的还是部署——让真正的研究者使用IncPy。这不仅是因为我觉得通过部署能够做一个更强有力的评估，而且更重要的是，我发自内心地相信IncPy可以提升计算型研究者们每日工作的生产力。大多数研究型原型工具的搭建都是为了证明一个新颖的想法，但之后就被抛弃了，但我想让IncPy成为一个实用且能被沿用下去的工具。我并不是为了发表论文才

凭空想出IncPy这个点子的，创造它的灵感来自于我研究时碰到的真真正正的实际的编程问题，所以我希望人们也能够在实际中使用它。

尽管我追求理想主义，但我也理解为什么几乎没有研究型的原型工具成功地得到部署使用。这背后的原因是因为人们往往不愿意去尝试一个全新的工具，除非他们看到这个工具有立竿见影的巨大好处，而且没有一点缺点；研究者们却通常没有足够的时间和资源来让他们的原型工具运行得足够好，从而满足这些严苛的条件。特别的是，我的目标用户对使用常规的Python很满意—即使它存在一些低效问题，他们也不愿意冒险转移阵地到IncPy上来。为了说服某个人试一试使用IncPy，我就必须保证IncPy在任何情况下都比常规的Python表现得要好。尽管这在理论上是对的，但实际上IncPy 只是一个仅由一个学生维护的研究型原型工具，所以它一定会存在很多漏洞。相反，官方版的Python是一个有二十余年历史的项目，而且同时还有几百个经验丰富的程序员在维护它，所以它更加稳定和可靠。只要有一个人发现了IncPy的一个漏洞，大家立马就会觉得IncPy是靠不住的，转而回去使用常规的Python。我知道胜算不站在我这边，但我不在意。

在斯坦福寻找人来安装和使用IncPy未果后，我转而在附近的大学来寻找愿意第一个吃螃蟹的人。2010年3月，我给UC Davis[22]的几个博士生发了封冷邮件，他们都是和我同在MSR实习的一位同学的朋友。感谢他们的利他主义精神，在我这个研究生亟需援助时拉了我一把，还邀请我去宣传我的IncPy。几个和蔼的教授甚至还同意和我会面，包括一个我在数据来源那个研讨会上碰到的教授。尽管我得到了不少有帮助的反馈，我还是没能找到人作我案例研究或者部署的实验对象。

[22]UC Davis是加州大学戴维斯分校（University of California, Davis）的简称。——译者注

那天晚上我待在了UC Davis，打算第二天早上再回斯坦福。突然，我有了个冲动的想法——给Fernando发封冷邮件，他是UC Berkeley[23]的一个研究科学家，非常热心于将Python投入到计算型研究中。几个月前，一个来听我作报告的研究生同学给我发了封邮件，里面有一个Fernando的一篇博客的链接，我当时就记下了一个备忘录，准备在将来某个时候联系一下Fernando。 现在似乎正好是一个合适的时机：伯克利正好处于UC Davis和斯坦福之间，我回家时可以顺便在他办公室门口停一下。我给Fernando发了冷邮件，询问他明天早上是否有时间和我聊一聊。这是一个不太会成功的尝试，不过他最终还是答应了和我见一面。我和Fernando欢快地聊了一个小时；有这么一位资深的高级科学家能支持我IncPy的想法，我感到非常高兴。

第一次与Fernando会面最重要的一个收获就是他邀请我接下来这个月回伯克利做一个有关IncPy的报告。他说我的听众中会有使用Python进行计算型研究实验的神经科学家。在我那个一小时报告的过程中，我被三个研究员稍稍吓了一跳，他们老是打断我，反复纠缠一些IncPy在实际中表现如何的细节问题。一开始，我觉得这几个人实在是太过学究了，不过报告后他们过来向我表达了试用IncPy的强烈兴趣。他们抱怨的所遭受的低效问题正好是我创造IncPy的初衷！看来他们并不是在我报告时添乱；他们是真真正正想了解其中的细节，以此来评估将IncPy部署到他们实验室的机器上去是否真的可行。

有可能会拥有第一批用户让我很激动。我和他们邮件交流了几次，还专门驱车去伯克利协助他们安装和设置IncPy。我的第一封邮件采用了一种谨慎乐观的语气：

> 非常感谢您有兴趣成为我IncPy解释器的第一个用户！我非常有热情将IncPy带到一个您能在日常工作流中使用的状态。我觉得最大的问题应该是安装、设置、配置中恼人的工作，我会解决

[23]UC Berkeley是加州大学伯克利分校（University of California, Berkeley）的简称，出于习惯，下文将简称其为“伯克利”。—— 译者注

这些问题从而给您最流畅的用户体验。

在我尝试给伯克利这些神经科学家的电脑上安装IncPy之前，我已经搭建和测试了几个月，所以我很自信也能给他们安装成功。然而，安装之后没一会儿，我们就发现IncPy 与这些科学家天天都在使用的很多第三方Python插件（称为*拓展模块*）并不相容。我自己测试的时候，只测试了一些基本用例，而没有考虑到这些扩展模块。这给了我一个关于现实部署的深刻教训：失败可能以你没有料到的形式出现，而一旦给用户留下一个不好的第一印象，就全完了！就像绝大多数研究者只在象牙塔里闭门造车做原型工具一样，我绝不会想到这个不可预见且不起眼的拓展模块问题，会让我在第一次部署的尝试中出局。

但我并没有放弃。我花了接下来几个星期的时间重新设计和实现了IncPy中一个关键部分的代码，使它能够与任何Python拓展模块一起完美工作。我又给伯克利那几个神经科学家们发了封邮件，希望再给我一次机会，但我没有得到任何回复。唉，我曾经有一个绝佳的机会，可我却挥霍了它。

~

这个惨痛的经历驱使我不断地从实际的角度出发来不断完善IncPy：一边修补了几个精细的漏洞，一边我还制作了一个IncPy的项目网站，上面包含一个短短的演示视频，文档以及新手教程。做这些东西的几百个小时对我原本的研究没有丝毫贡献，但这对我获得实际用户的事例，从而在将来投稿时完成论文的评估章节来说却是必须的。

几个月以后，来自世界不同地方的三个陌生人通过这个网站发现了IncPy，他们下载了IncPy，并将其用来提升他们研究中一些编程工作的速度。虽然三个人只是一个可怜的用户总数，但也总比零好，而大多数研究型原型工具用户数就是零。IncPy已经达到了一个打磨得还不错的状态，我很满意——这多亏了我从伯克利回来之后的做的改进——这几个陌

生人现在可以不需要任何人工指导就能使用IncPy了。这三个人发邮件告诉了我IncPy是如何在他们的一些工作中起到帮助的。这些事迹虽然并不是IncPy高效性的强有力证据，但也聊胜于无。

2010年9月，在我的第四年博士生涯接近尾声的时候，我把我IncPy的论文投稿到了一个顶级会议，论文的评估章节包含了案例研究和三个简要的部署实例。这个会议头一年的接收率只有14%。所以我知道我的论文很有可能被拒收，既因为它极低的接收率，也因为我的IncPy并不能很好的契合任何一个传统的子领域。尽管这样，先以顶级会议为目标，之后有必要再重新投稿到二级会议仍是一个明智的选择，毕竟顶级会议的论文发表在将来毕业的时候会占到更大的权重。

我仍然还没拥有一条通往毕业的康庄大道，但我至少能开始自我管理自己的研究内容，而不是跟着别人一起做他们的研究项目了。我在过去这一年还是挺开心的，我把脑海中灵光一现的IncPy想法变成了一个半实用的工具，还让三个下载它的陌生人从中受益。虽然这只是一个很小的成就，而且对我的毕业没有丝毫帮助——教授们更看重理论上的新颖性，而非实际中的部署情况——但在我博士的第四年末尾时，这多少还是有一点让人满意的。

第五年：峥嵘岁月

第五年（2010年9月）刚开始的时候，我还没有任何东西可以放到我的（还未存在的）博士论文里去。到这时，我的大多是同学都以第一作者身份发表了至少一篇博士论文水准的会议论文。而我却一篇都还没有（IncPy的那篇论文当时还处于评审阶段），因此我很担心我总共需要七八年才能顺利毕业。

然而在接下来的十二个月中，我发表了四篇会议论文和一篇研讨会论文（均为第一作者），这也铺平了我通往毕业的道路。毫无疑问，我的第五年是博士生涯中最高产的一年，我不懈地保持着专注。

~

2010年的夏天进行到一半的时候，IncPy项目正按部就班的进行着，我也正准备着在九月的截止日期前投一篇论文。但我知道对于博士论文来说，IncPy还不算是一个实质性的工作。所以除了继续努力准备投稿而外，我也花了一下时间想了想下一个项目要做什么。

我希望可以把我的头脑风暴说成是受纯粹的学术精神所激励，但事实上我是受日益增长的恐惧所驱使：我担心我不能在一个能接受的时间内顺利毕业，所以我逼迫我自己去想一些有希望发表出来的新想法。我也知道一篇论文被接收发表可能会花两到三年，因此我要是想六年之内毕业，就得在今年提交几篇论文，并祈祷其中有至少两篇会被接收。我感到了一种

紧迫感，因为我的外源奖学金只持续到这一年结束。等我的经费一到期，要么我就得去从教授那里获取经费支持并面对一些强制性的限制（例如再一次在Klee上投入工作），要么我就得长期地担任助教并延期毕业。时间真的不多了啊。

2010年7月29日，几乎就是我想出IncPy的初始创意正好一年之后，我又有了一个相关的想法，同样也是受计算型研究者分析数据时碰到的实际问题的启发。我注意到由于研究者们经常以一种专门的"马虎"的方式写程序，他们的程序就常常由于一些低级错误而崩溃，没能产生任何一点分析结果，让人垂头丧气。我的想法就是通过改变Python 语言的运行时环境（解释器），我可以消除掉这种崩溃，让这些马虎的程序能产生一部分的结果，而不是什么结果都没有。我把这个Python解释器的修改版命名为"SlopPy"，代表**Slop**py **Py**thon。

虽然SlopPy和IncPy是非常不同的两个想法，但我都是通过改变Python解释器的行为来实现它们的。过去一年为了做IncPy，我花了近一千个小时来研究（修改）Python 的解释器，这让我有信心能够轻易地实现出SlopPy。只用了两个月，我就做出了一个能工作的基本版原型工具，跑了一些初始的实验，并提交了一篇论文到一个二级会议。我以那个会议为目标，既是因为它的截止时间比较合适，也是因为我觉得SlopPy 这个想法还不足够"大"来让它被顶级会议接收。

~

到了2010年10月的时候，我有两篇论文都处于评审中。这时，我已经放弃了找一份大学教授工作的想法，毕竟我连一篇博士论文能拿来用的论文都还没发出来；有竞争力的计算机教职候选人在博士生涯的这个时候都已经发了很多篇受赞誉的第一作者论文了。所以除非奇迹发生，要不然我是找不到一个一流大学的研究工作了，因此我把完成工作够得上毕业作为了我的目标，都不再去担心我的简历字面上好不好看了。

我收到的是奇迹的反面：我IncPy和SlopPy的论文都被拒收了。我感到有点失落但也并不震惊，因为在这之前我已经习惯了论文被拒。对我论文的批评意见都是合情合理的，所以解决这些问题可以强化我之后的投稿。

最值得注意的是，我介绍IncPy时采用了一种不太明智地包装策略，这使得我的论文被安排给一些来自对我的研究理念不太“友好”的子领域的学者来审阅。理论上来说，专业论文应该只基于它的内容来评判，但实际上，审稿人有他们自己不同的主观品味和理念差异。所以我彻底地重写了开头用来自我推销的部分，希望获得更多友善的审稿人的赞同，我还将它重新投稿到了一个二级会议，以此来进一步增加它被接收的机率。我的计划奏效了，2011年年初，我IncPy的那篇会议论文的第二次投稿被接收了，尽管评审意见有点冷淡。

后来我又修改并重新提交了SlopPy的那篇论文到一个研讨会，这个研讨会正好和我要作报告介绍IncPy的那个会议在一起举行。这个策略很有效，因为想让一篇论文被一个研讨会接收比被一个会议接收实在是容易太多了。而且，Dawson也不用花额外的钱让我去参加这个研讨会，因为我本来就是要去同时举行的那个会议上做IncPy的报告的。SlopPy的论文和我料想中的一样被接收了，尽管它不“作数”，只能算作我博士论文外的而外成果，但至少也比没有论文发表好；我希望能把这篇论文吸收到我的博士论文中去作一个小章节，以此作为那些来自会议论文的更重要章节的补充。

~

2010年10月，我刚把IncPy和SlopPy两篇论文投出去的时候，我问了Dawson我要毕业的话需要达到什么条件。不出所料，他回答我说我需要发表一些论文来证明我工作的合理性。不过他还有一个更为具体的建议：另一个能到达博士论文水准的项目，这个项目能够把我对Python的兴趣以及他所喜爱的类似Klee的想法结合起来，从而做出一个能够自动

查找Python程序中漏洞的工具。由于我对回到任何形式的Klee项目都没兴趣，我没理会他的建议，而是继续思考IncPy和SlopPy还有没有什么可拓展的东西，可以形成一篇后续论文提交出去。

这时，我博士论文主题的初期构想已经开始在我脑中萌芽：IncPy和SlopPy都是*提升计算型研究者生产力的软件工具*。因此，思考接下来一个项目要做什么的时候，我仍接着去寻找计算型研究者在工作中会碰到什么样的问题，然后设计新的工具来解决这些问题。

具体点说，我注意到研究者们每天跑计算实验时，都要编辑并执行他们的Python程序几十上百次；在有一个重大发现之前，他们往往要重复这个过程几周或者几个月。我觉得把每一次程序执行之间的变化记录下来并进行比较，会对调试程序和获得思路有帮助。为了便于这种比较，我打算拓展IncPy，使之能够记录下当一个Python程序执行时，哪部分代码和数据被存取了的细节，从而维护一个计算型实验的详尽历史记录。我也认为如果研究者之间能够共享这种*实验历史记录*的话那就太棒了，这样他们就能够从中知道哪些实验尝试是奏效的，哪些是不行的。

我的直觉告诉我按这个思路发展出来的想法肯定是新颖的，而且能够发表出来，但我还没有形成一个清晰的研究亮点；我的思路还都是模糊的一团。我觉得思维有点被卡住了，于是我又寻求和Fernando见一次面，就是我博士第四年在伯克利一个实验室的最后作报告介绍IncPy时见过一次面的那个教授。Fernando把与我的会面排入了日程，我们这个小时的交谈为我的下一个项目种下了种子。

~

我一把我拓展IncPy来记录基于Python的实验历史的想法告诉Fernando，他立马饶有兴致地给我介绍起了一个我从没听过但却很吸引我的话题：*可重复性研究*。

实验科学的根基之一就是任一个人的研究发现应该能够让同行验证、比较，并且以此为基础继续发展。在过去的十年中，来自不同领域越来越多的科学家都在通过编写计算机程序来分析数据和获得科学发现。每年都有几千篇充斥着以数据、图像、表格为支撑的定性结果发表出来。然而，现代科学中大家都心知肚明的一个令人愧疚的事实就是，想重复或者验证他人的发现几乎是不可能的，因为他们的原始代码和用来产生结果的数据集几乎很难得到。这造成的结果就是，大量论文中包含的触目惊心的错误——有的的确是无心之失，但有的却是公然造假——变得无法被检验，有的甚至变成了导致人员丧命的科学论断。最近几年，诸如Fernando这样的改革派科学家们正竭尽全力唤起大家对计算科学中可重复性研究重要性的重视。

为什么可重复性在实际中如此难以实现呢？一部分有很强竞争力的科学家故意不公开他们的代码和数据，以此来避免潜在的竞争，但大多数人还是愿意在别人有需要时分享给他们代码和数据的。然而，主要的技术壁垒还是仅仅获得了别人的代码和数据，还是不足以重新运行和产生他们的实验。每个人的代码都需要一个高度特殊化的环境来运行，任意两台计算机上的环境——即使它们有相同的操作系统——还是会有一些细微且不兼容的差别。所以，如果你把你的代码和数据发给你的同行们，他们还是有可能没法重新运行你的实验。

Fernando很喜欢我IncPy的记录实验历史的想法，因为这样可以记录下一个实验最初出现时，有关其软件环境的信息。之后使用Python的研究者就可以把他们的代码、数据和环境发给想要重复他实验的同行。会谈出来以后我感到非常振奋，我找到了我想法的一个具体应用。可重复性研究的这个动机似乎也足够吸引人，可以组成我第二篇基于IncPy论文的故事情节，还能成为我博士论文的一部分。

正当我把更多的细节草记下来的时候，一个极为清晰的念头瞬间袭来：为什么只把实验记录限制在Python程序上呢？有了我头脑中的一些想法，我可以做出一个工具，能够轻松重复任何语言写的计算型实验。

头脑还十分混乱的我草拟了一个计划，我准备设计一个新工具，名字就叫“CDE”，也就是**C**ode, **D**ata, and **E**nvironment的意思。

~

当我把我的想法告诉Dawson时，他非常支持，还鼓励我再想得大胆一点：*为什么只把*CDE*瞄准在科学家的代码上？为何不做一个针对所有种类软件程序的多用途工具包？*真是字字珠玑啊。一大批软件的创建人和贡献者——不只是科学家——都遇到过让别人运行自己的程序时，由于同样的环境不兼容问题带来的烦恼，这一现象被亲切地称为“依赖地狱”。依赖地域问题在基于Linux的操作系统中尤为广泛，因为用户使用的不同变种的linux之间存在大范围的不完全兼容现象。在一个用户的Linux计算机上能运行的程序不一定也能在另一个用户的稍有不同的Linux计算机上运行。只要把我的原始想法稍加改进，CDE就能帮用户打包他们的程序，从而让其他用户能够成功运行，而不再用担心环境不一致的问题。我感到十分激动，因为CDE有可能缓解Linux 里存在了十余年的依赖地狱问题。

按惯例要进行实际调研，于是我在网上搜索了相关工作，看看有没有什么研究型原型工具和生产质量工具有着类似的功能。让我欣慰的是，已有的工作并不算多，而且CDE在这个宽松的竞争环境中有两方面显得尤为鹤立鸡群：首先，我将CDE设计得比类似的工具更容易上手使用。作为用户而言，你要新建一个自我完备的代码、数据和运行环境的包，就只需要运行你希望打包的程序就可以了。因此，如果你在Linux计算机上运行了一套程序，那么CDE就能让其他人在他们各自的计算机上都能返回到相同的这个程序，而不需要其他的安装或者配置。第二，CDE所采取的的技术机制——一种叫*系统调用重定向*的技术——使它在各种各样复杂的实际情况中，比其他相关的工具更为可靠。

这时，CDE还只是以一堆笔记和设计提纲的形式存在，但当我意识到它比所有现存的工具概念上都更简单，更易于使用，且更为可靠时，我还是感受到了它的巨大潜力。我身体里的一部分感到震惊和激动：*居然之前*

没有人实现过这个东西？！回想起来这个想法这么显而易见！一个令我惧怕的之前没人做类似CDE工具的原因可能就是，要搞清楚所有的细节并让其在实际中高效工作几乎是不可能的。可能CDE也是一种论文上看起来不错但却实际中不可行的创意。我于是觉得除了自己亲自实现一遍CDE这条路而外，没有其他更好的办法把这件事弄清楚了。

在横跨2010年10月到11月那紧张的三个星期中，我都在为创建出CDE的第一个版本而高度研磨。正如我所怀疑的那样，虽然CDE背后的研究创意是很直截了当的，但要让CDE在真正的Linux程序上工作，还有很多编程相关的脏活累活要干。那几个星期我与CDE生活甚至呼吸都是在一起，把我生活中的其他事都抛之脑后。我夜以继日地编写程序，还常常在梦里思考我需要解决的代码中错综复杂的细节问题。每天早上，我刚醒来就跳起来编程，生怕今天我会遇到一个难以逾越的障碍，最终证明让CDE在实际中工作是不可能的。不过这一天迟迟没有到来，我也离我的第一个里程碑越来越近：证明CDE是如何能让我在两台Linux计算机之间毫不麻烦地传递一个复杂的科学程序，并重复一个实验的。

在经过三周靠咖啡驱动的高强度研磨之后，我感到狂喜，我终于让CDE在我的一个科学程序演示样本上正常工作了。这时，我知道如果我继续测试并不断改进代码，那么CDE有在多种实际的Linux程序上起作用的潜力。我制作了一个十分钟的介绍CDE的演示视频，搭建了一个包含视频和CDE下载拷贝的网站，并把网址发给了我的几个朋友。我不知道的是，有一个朋友还在著名的计算机Geek论坛Slashdot上帮我发了这么一个广告:

> 斯坦福的研究院Philip Guo开发了一个名叫CDE的工具，可以自动将Linux程序极及其依赖（包括系统级的库，字体等等）打包，并能直接在另一台Linux计算机上运行，并不需要做设置库和程序版本，或者解决版本依赖地狱问题这些工作。他已经上传了二进制文件、源代码和视频截图。看起来应该对集群/云部署和程序共享有很大帮助。

还不到24小时，Slashdot论坛上的那个帖子就有几百个回复了，我也开始收到几十封邮件，都是来自世界各地下载并试用了CDE的Linux狂热粉。其中有一些珍贵的来信，比如：“我就是想说你简直太酷炫了！这个创意能够实际起作用是在是令我折服。我会在我们墨西哥提华纳的Linux社区里推荐使用这个工具。”这些来自实际用户的未经修饰的、随感而发的赞誉，比起来自研究同行们对我之前创意或者论文的表扬，对我而言意味着更多。

~

第六年：尘埃落定

结语

后记