我和 Google 的故事（2012 年原版）

我和 Google 的故事（2012 年原版）

也许有人看见过我批判 Google 的那篇英文文章。它有一部分片面性，所以被我从英文博客上拿下来了。我一直在反思自己在 Google 的经历，也许现在用自己的母语，我可以得出一个准确一点的结论吧。

也许有人觉得作为一个读了这么多年的 PhD 去给别人做实习生（intern）是一种耻辱，但是我亲眼看到，从一些名校比如 Yale 毕业的 PhD，在 Google 混了好几年，也不过是过着差不多的生活。只不过做了 intern 之后我长了经验，知道了自己的价值，以后不至于落到同样的位置。

这里我就讲述一下我在 Google 的实习经历吧，也许对人有参考作用。

受命于危难

先说说我的项目是怎么开始的吧。当我加入的时候，我的老板 Steve Yegge 的小组试图制造一个跨语言的“服务器”式的编程工具，叫做 Grok。你可以把它想象成 Eclipse, 但是 Grok 的设计目标不只是像 Eclipse 那样检索和分析本机的某一种语言的代码，而是大规模的检索和分析 Google 的所有项目，所有语言，所有代码。这包括 Google 的“四大语言”：C++, Java, JavaScript, Python，一些工具性的语言：Sawzall，protobuf 等，还有一些“build file”和所有第三方的库。Grok 的初期设计目标是一个静态的代码索引服务，只要程序员点击任何一个变量或者函数名，就能“准确”的跳转到它定义的位置。动态的编辑功能稍后也在陆续加入。

这种检索不是像 ctags, etags 那种简单的正则表达式匹配，而是像 Eclipse 和 Visual Studio 那样的准确的“语义检索”，所以它必须真正的理解程序语言的语义。在 Grok 诞生以前，市面上和 Google 内部都没有一个工具能正确的支持所有“四大语言”，所以我不得不说，Steve 的项目比起 Google 的其他更编程语言相关的项目，是相当先进的。

虽然 Grok 的技术含量很高，但是 Google 的管理层对东西的评价并不是看技术含量的，而是看你有多少“影响力”(impact)，说白了也就是有多少用户。Google 当时本来就只有不到一万个程序员，一个“内部编程工具”能有多少“用户”呢？所以 Grok 比起像 CodeSearch 一类利用正则表达式来查询程序的“低端”项目来说，在管理层心目中并不占任何优势。而且由于其它项目界面好看些，用户多些，Grok 随时有被取消的危险，这使得 Steve 心理压力很大。我就是在这个“危难关头”进入他们的小组的。我当然没蠢到会自己进入这样一个组，但是 Steve 在电话面试时把一切都说得很美好的样子。当时小组里只有三个人：Steve 和另外两个组员。

恐惧和疑惑

当我开始的时候，Grok 小组已经初步完成了 Java 和 JavaScript 的检索模块。但是他们的检索并不是从头设计的，而是从 Eclipse (JDT) 和 JSCompiler 里面分别“挖取”了对 Java 和 JavaScript 语义检索的部分，修改之后插入到项目里的。Eclipse 的设计非常的不模块化，以至于项目进行了一年多，大家还在忙着解决它带来的各种 bug。

最开头的时候 Steve 给了我两个选择：检索 C++ 或者是 Python。我觉得 C++ 的设计太繁琐，所以就选择了看起来好一点的 Python。Steve 就让我去找一个好一点的开源的 Python IDE，然后把里面的语义检索部分挖出来插入到项目里面。可是在看过十个左右的“Python IDE”之后，我发现它们没有一个能够正确的“跳转到定义”。分析其原因，是因为这些 IDE 基本上做的是正则表达式匹配，而完全不理解 Python 的语义。所以我对 Steve 说，我要自己从头写一个。但他反对这个提议，因为他觉得这是三个月的时间之内不可能完成的。不但是我不能，而且就算一个小组的高级程序员也不可能完成。就算完成了，他也不想“维护”这些代码。所以他宁愿让我去拿一个不怎么样的开源项目，因为这样“维护”的工作就转嫁到开源项目身上去了。这也许就是 Google 支持开源运动的原因吧？

可是我很清楚的看到，这样一个语义检索，不过是一个抽象解释器 (abstract interpreter)。写解释器是我最在行的，所以我告诉他这是我可以完成的，而且由于设计上的简洁，我的代码的维护代价会比使用一个开源项目小很多。他没有说话。我同时也在进行一些内部培训，看一些视频，折腾 MapReduce 一类的内部工具教程，就这样过了一个星期。我隐约的感觉到 Steve 的不快，因为他不怎么说话了，可是我也没有太在意，仍然傻乎乎的到处凑热闹。到了周五的时候，Steve 下午很早就回家了。另一个组员还待在哪里，不时的叹气。我对她说：“Steve 是不是不高兴了？我知道我说话有点太自信，可能打击到他了。”她好像打满的气球被开了一个洞：“他怎么会被你打击到？你知道他以前做的项目有多厉害吗？他是怕你做不出来。之前有一些 intern 设的目标太高，以至于到最后没有完成他们的项目。”于是她打开 Eclipse，把 JSCompiler 的代码给我看。“你知道我们以前一个类似的项目 JSCompiler，花了多少时间才完成吗？一个小组的人，四年的时间！”她打开其中一个文件，也就是处理符号表的那个模块，说：“看这一个文件就有 9000 多行代码。你三个月能写出这么多代码吗？”我翻了一下白眼，搞笑似地说：“啊～ 怎么可能有 9000 多行？这些人真的知道怎

么写这种代码吗……”

后来具体的对话我忘记了，但是她确实给了我一些压力，再加上 Steve 那个闷声子，真是不好受。所以那个周末我没有出去玩，我下载了一个 Jython，把它的 parser 文件 (ANTLR) 拿出来。然后自己设计了一个更简单的 AST 数据结构，把这个 parser 生成的 AST 转换成我的结构。然后就开始在上面写一个抽象解释器。由于 Java 的限制，我想出了一个更简洁的用 Java 实现解释器的方法，从而避免了使用繁琐的 visitor pattern。一个周末之后，我做出了一个基本的原型。当然因为 Python 语言的复杂性，有很多细节的东西到后来才完全的实现。

等到星期一的时候，我告诉 Steve 我做了一个原型出来，而且因为我拿了 Jython 的 parser，我们以后可以用这个理由把这代码 merge 回 Jython，给他们提供功能，让他们帮我们维护代码，对两方都有好处。他居然一点也不高兴，把我叫到一个白板前面，板着脸说：“来，给我讲一下你打算怎么做。”我就画了一个 AST 的类关系图，在里面每个类插入一个叫 interp 的方法，然后指出这个东西就是一个抽象解释器。最后他豁然开朗了一样，说：“好。我相信你知道你在干什么了。就这样做吧。”

陌路

在 Google 的整个夏天我都觉得跟其他人没有共同语言。我感兴趣的东西，他们一点都不了解。我觉得不以为然的一些东西，却被他们捧上了天。比如，有一次几个人在谈论一个 Google 的“牛人”，说他做了一个多么了不起的项目，很快就升为了 Staff Software Engineer （“Staff”是比“Senior”高一级的职位，Steve 就是个 Staff）。我去看了一下这项目，发现不过就是 JUnit 的“C++ 版本”。JUnit 这东西技术含量本来就是相当低的，做这样一个东西就能当“Staff”，那我岂不是轻而易举就可以成为“Principal”了？哈哈。我心里这样想，但是没有说出来。一个 Staff 就如此，谈到 Google 的两个创始人的时候，有些人就简直是黑白不分了。对他们的各种武断的甚至愚蠢的做法，居然都津津乐道。创始人在他们眼里俨然就跟皇帝一样，他们做什么都是对的。这种浮夸和互相吹捧之风，恐怕是在其它公司也少见的。Google 要求员工们保持一种“Googley”的态度，原来就是这样的态度，过度“正面”和“积极”。美国所崇尚的“个人主义”和“批判性思维”，我在 Google 还真的没有见到过。

另一些时候，我会遇到一些对某种语言或者技术有宗教情绪的人。有一次一个工程师坐到我面前，像是在面试我一样，正儿八经的开始自我介绍，后来我们就谈到 C++。我说 C++ 设计实在是太繁琐了，其实很多简单的语言效率并不比 C++ 低，C++ 最近其实在向其它高级语言学一些东西…… 后来这人就不说话了。那天以后我就发现跟他打招呼他都不理了。后来我才发现，在 Google 是不可以指出某种语言，特别是 C++ 的缺点的。C++ 在 Google 的势力之大，连 Java 都只能算二流货色。

最让我受不了的其实是 Google 的气氛。总体感觉就是过度“和谐”，没有人说真话，以至于你不知道什么好，什么不好。很多文档，视频，活动都挂着“Google Confidential”的标签。等你去看了，却发现相当幼稚，其实是众所皆知的东西，没有什么机密可言。可是大部分的实习生们却有一种受宠若惊的感觉，或者假装有这种感觉。每个星期五，都会有一个“TGIF”，两个创始人会像主持人一样组织一个大会。本来无可非议，但是总感觉气氛过于群情激昂了，有点像文化大革命时候念红包书的感觉。好不容易大家聚在一起，总是在听新闻发布，不然就是谈工作。真正大家一起玩的 party，却非常稀少。所以一些别的公司的人都在疑惑，Google 的员工到底有没有下班的时间。

我就是这样度过在 Google 的每一天，以至于后来我都不怎么在饭桌上吃饭了。自己把饭端到靠墙的吧台去吃，或者坐在“冰激凌吧”跟里面的厨师聊天，省得遇到一些高谈阔论的人无语。我发现自己跟打扫卫生的大妈小妹们也谈得来，她们也喜欢跟我说话。后来我发现有这种感觉的不只是我，另外两个比较厉害的博士生也懒的在那边吃饭了。其中一个说他一个星期就把自己的项目做完了，然后假装仍然在做，免得又被增加任务。

皇帝的织布工

虽然 Steve “允许”我自己做一个 Python 分析器，但是这却不是没有压力的。这种感觉就像是“皇帝的新装”里的织布工一样。我扬言自己会做出精美绝伦的布料，皇帝的大臣们却看不见，所以他们就相当的小心。总是对我很敬畏的样子，有时会来问候一下，做得怎么样了。可是一旦扯到深入的话题，却又怕被看穿其实他们不懂很多东西。因为我的教授们研究 Scheme，所以 Steve 有时候也会很激动的表扬 Scheme，或者类似 Scheme 的语言比如 Clojure。这种奉承真的让我受不了，生搬来的术语都是错乱的。为什么程序语言总是引起这种宗教的态度，不是抵制就是膜拜？

有一次一个 Staff Software Engineer 来访。看我在做这个 Python 分析器，很鄙夷的样子，说：“你做那个东西干什么。Python 本来是没有类型的，怎么推导得出类型来？我倒希望 Java 的类型推导做得更好一些，不用手写很多类型。”显然他不知道什么是类型推导，他也不知道如何把 Java 的类型推导做得更好。很多人把自己的命运寄托在语言的设计者身上，自己没有能力去改进它们，所以他们才会对程序语言顶礼膜拜。

压力

直到有一天，我才发现 Steve 为什么这么紧张。那天有另一个“分舵”的 director 来访。他给我们做了一个关于“创新”（innovation）的演讲。基本内容就是说，技术上的创新，如果吸引不到用户，那就不算什么创新，拉得到用户的东西才叫创新。

那天下午，这个 director 来到我们的办公室。表情严肃的“审问”Steve：“你说你每天有 5000 个用户。可是 Google 总共还不到 10000 个程序员。你是怎么算的？你把接受你的服务的那些下游项目的用户全都算进去了吧！”唉，想不到大名鼎鼎的 Steve Yegge 在这种皇帝的钦差大臣面前也只能唯唯诺诺。

我可以说，这个 Python 的东西，虽然不费我很多力气，但却是 Google 里很少有人可以做出来的。就算 Python 的创造者 Guido van Rossum 恐怕也玄。所以实际上我的这个东西在很大程度上拯救了这个濒临灭亡的项目，因为一旦 Grok 支持所有的“Google 语言”，就会有很多人注意到这个东西，从而会有“影响力”。这确实是后来发生的事，我走了之后，Grok 开始通过 API 给很多项目提供服务，包括 CodeSearch。

Google 给我的那点工资，其实是根本买不起这样的软件的。你可以参考一下像 CodeSonar 之类“静态分析”软件的价格，一份基本上就是我三个月的工资。由于我上学想找点外快，让他们捡了一个便宜。可是这种“上级领导”的压力居然也间接的传到了我身上，而且是以一种非常不尊重的方式。这种感觉就是，你做得再多再出色，你相对于 Google 的“大拿”们，什么都不算。这也许就是 Google 为什么雇佣 Dennis Ritchie, Brian Kernighan, Ken Thompson, Rob Pike, Guido van Rossum 等大牛吧。因为它就可以说：“看我们 Google 有这些顶尖牛人，你算个什么，要不断努力！”Steve 不止一次的对我说：“你要为 Google 做出杰出的贡献啊！Google 的东西总是最好的，你要做出 Google 一贯的品质来。你知道 Python 的创造者 Guido 也是 Google 的员工吗？我一定会在他面前给你美言几句。”这种语气，我好像在几十年前的中国听说过呢？“你要为祖国做出杰出的贡献！”他也许以为我会受宠若惊，可是我心里却不是个滋味。

总之他们就是用这种奉承，利诱，竞争，加威胁的方式，想方设法让我多做事情。可是我心里想的是，Google 老爹，您就给了那么点钱，您想买多少东西啊？本来这系统能做出来就不错了，一个组员却一直催着我写 test。她根本不明白，一个程序并不是写了测试就会是个好程序。这个程序经过我多次的大规模修改甚至推翻重来，即使一早写了测试，那些测试也会很快作废。这种大公司给人灌输的“test-driven”编程方式，在这种创造性的程序设计里是根本就是行不通的。要写出这样一个系统，必须全神贯注，深入到语言的本质。而去写测试，往往会打乱原来的思路，所以测试应该是最后完成之后才写的。当我最后完成这个系统，可以大规模的处理 Python 代码的时候，却听见从她的桌上传来一声沉闷的咆哮：“WRITE–THE–TESTS—”这真的非常的 Googley!

结果

最后我顺利完成了整个项目。现在它每天都会把 Google 所有的 Python 代码索引一遍。很多内部工具比如 CodeSearch 里面的 Python 文件上的链接，都是这东西做出来的。我所有的代码加起来才 4000 行。处理符号表的模块只有 600 行。我怎么也想不通为什么 JSCompiler 会有 9000 行来处理这么简单的东西，但是也许这就是为什么 JSCompiler 花费了四年时间。

总结

所以你看到了，这就是我对 Google 的印象。有好几次我都看到很不错的工程师被 Google 雇佣了之后就销声匿迹了一样，为 Google “默默奉献”一生，不再有自己的发明创造。我感觉 Google 就是一个埋没人才的机器，而它的“创造性”的名声，却让越来越多的人才被埋没。主动找上门的人才被埋没了不说，还吞并其它公司，并且对他们施行同样的“Google 文化”。其实我本来挺喜欢一个公司叫 ITA Software，后来也被 Google 收购了，据说连“文化”也变了，所以我也不再想为 ITA Software 工作了。唉，Google，别再吞并我喜欢的公司了，改造一下你自己吧。