

## 附录 C 两个公司的讽刺小品

“我要加入一个俱乐部，并用它来让你就范。”

——Rufus T. Firefly

### Rufus 公司 项目开始

你叫 Bob。日期是 2001 年 1 月 3 日。刚刚渡过了新千年的狂欢，你的头还疼着。你和几个管理人员以及一些同事坐在会议室中。你是一个项目团队的领导。你的上司也在其中，他叫来了归他管的所有的团队领导。他的老板召集了这次会议。

“我们有一个新项目要开发，”你老板的老板（我们叫他 BB）说。他的发尖是那么的高，都擦到天花板了。你的上司的发尖才刚开始长出，他急切地等着有一天他也可以把 Brylcream 护发香波的痕迹沾染在吸音瓦上。BB 描述了他们调查过的新市场的基本情况，以及他们想开发的用来开拓市场的产品。

“到第 4 季度，10 月 1 日时，我们必须完成这个新产品，并使之可用”，BB 要求道。“任何事情都没有它的优先级高，因此要取消你们当前的项目”。

大家的反应出奇的安静。数月的工作就这样完全被丢弃。慢慢地，低沉的反对声开始在会议室中传播。

当 BB 和房间中每个人目光相遇时，他的发尖发出邪恶的绿光。和每个人相视时的阴险目光使每个在场者不寒而栗。显然，他不容许在这个事情上进行讨论。

大家一恢复安静，BB 就说，“我们要马上开始。你们需要多长时间来进行分析呢？”

你举起了手。你的上司试图阻止你，但是他投掷的小东西没能击中你，你没有觉察到他的举动。

“先生，在得到一些需求前，我们无法告诉你分析会花费多长时间。”

“需求文档要在 3 或者 4 周后才能准备好，”BB

### Rupert 工业公司 项目：~Alpha~

你叫 Robert。日期是 2001 年 3 月。在假期中你和家人所度过的轻松时光使你恢复了精神，准备投入工作。你和你的开发团队坐在会议室中。部门的管理者召集了这次会议。

“我们有一些关于一个新项目的想法，”部门管理者（称他为 Russ）说。他是一个容易激动的英国人，他的精力比聚变反应堆还要旺盛。他雄心勃勃并具有紧迫感，但是他了解团队的价值所在。

Russ 描述了公司了解的新市场机遇的基本情况，并把你介绍给 Jay，负责定义用来抓住这个机遇的产品的市场管理者。

和你打过招呼后，Jay 说，“我们希望尽快开始定义我们首次要提供的产品。你和你的团队什么时候能和我谈谈呢？”

你回答道，“本周五我们会完成我们项目的当前一次迭代。在这之间，我们可以抽出几个小时和你谈谈。迭代完成后，我们会从团队中抽出一些人专门投入到你的项目。我们会立即开始招聘一些人来接替他们并为你的团队招聘新人。”

“好极了，”Russ 说，“但是我希望你明白，7 月份我们要在产品展览会上拿出一些东西展示，这很重要。如果我们到时不能拿出一些有意义的东西，我们就会失去机会。”

“我明白，”你回答道。“虽然我还不知道你打算做的是什麼，但是到 7 月时肯定可以拿出一些东西来。我还能马上告诉你那个东西什麼。无论如何，你和 Jay 将完全控制着开发人员的开发方向，所以你可以放心，到 7 月要去展示时，你会拿到可以完成的最重要的东西。”

说，他的发尖由于沮丧而震动着。“假如现在需求文档就在你面前。你需要多长时间进行分析呢？”

大家都屏住呼吸。每个人都环顾着其他人，看看他们是否有一些注意。

“如果分析需要的时间超过了4月1日，那么就会出现。到那时，你们能够完成分析吗？”

你的上司明显地鼓足勇气，突然说道，“先生，我们会找到办法的！”他的发尖增长了3mm，而你的头痛也增加了，需要服两片去痛片才行。

“好。”BB露出微笑。“现在，设计要花费多长时间呢？”

“先生，”你说。你的上司明显脸色苍白。显然，他在担心他那3mm会有危险。“没有分析，是不可能告诉你设计会花费多长时间的。”

BB的表情难以置信的严厉。“假如你已经做过了分析！”他说，同时还用他那透露着无知的小圆眼注视着。你。“那么，设计会花费你多长时间呢？”

两片去痛片都不能减少疼痛。你的老板，不顾一切的想保住他新增长的发尖，插嘴说道，“嗯，先生，只剩下6个月的时间来完成项目了，设计最好不超过3个月。”

“你能同意，我很高兴，Smithers！”BB面带喜色地说。你的上司放松了一些。他知道他的发尖保住了。过了一会儿，他开始轻轻地哼起Brylcream的广告语。

BB继续说道，“好，4月1日前完成分析，7月1日前完成设计，那么你们有3个月的时间实现项目。这次会议是一个榜样，它表明了我们新的协商和授权程序工作的有多么好。现在，大家可以离开，开始工作了。我期望在下周前，在我的办公桌上看到TQM（全面质量管理）计划以及QIT（质量改进团队）任命情况。哦，别忘了在下个月的质量审计中，你们交叉功能团队要开会并进行报告。”

“忘了去痛片，”当你返回小卧室时，心中想到。“我需要波旁威士忌酒。”

你的上司过来找你，明显带着兴奋，说道，“天哪，多么美妙的一个会议呀。我认为关于这个项目，

Russ满意地点点头。他知道这种工作方式。你的团队总是能让他了解并把握开发情况。对于你的团队会首先着手于最重要的工作并产生出高质量的产品这一点，他极有信心。

~ ~ ~

“那么Robert，”Jay在第一次会面时说道，“你的团队对被分开是怎么看的？”

“我们会怀念在一起工作的日子，”你回答道，“但是，我们中的一些人对于最近的一个项目相当厌倦了，并且期望有一些变化。你那边在做些什么呢？”

Jay微笑着说，“你知道我们的客户当前遇到了很大的麻烦……”接着他用大约半个小时的时间描述了问题以及可能的解决方案。

“好，请稍等一会儿”你说道。“我需要把这搞清楚。”因此，你和Jay谈论了系统可能的工作方式。Jay的一些想法并没有完全成形。你提出了一些可能的方案。他对其中的一些表示赞同。你们继续讨论。

在讨论期间，对于所提出的每个新主题，Jay都编写了相应的用户素材卡。每张卡片都描述了新系统必须要做的事情。卡片堆积在桌子上，展开在你们面前。当你们讨论这些素材时，你和Jay都会指着它们，把它们拿起来，并在上面作一些记录。这些卡片是有效的助记工具，你们使用它们来描述一些刚刚成形的复杂想法。

在会谈结束时，你说，“好，我对你想要什么已经有了一个大体的认识。我和我的团队会对它进行讨论。我想他们会对各种不同的数据库结构以及表示格式进行一些试验。下次我们会面时，就会有一个团队，并且我们会开始确定系统最重要的特性。”

一周后，你新建的团队和Jay会面。他们把现有的用户素材卡在桌子上铺开并开始研究系统的某些细节。

会议非常有生气。Jay把素材按照它们的重要性排好。对于每一个素材都进行了大量的讨论。开发人员关心的是要保持素材足够的小，这样便于估算和测试。所以他们不断地要求Jay把一个素材分成几个小一些的素材。Jay关心的是每个素材都要有一个

我们真的会做出一些震惊世界的事情。”你愤怒得只能点头同意。

“哦，”你的上司继续说道，“我几乎忘了。”他交给你一份 30 页的文档。“记住，SEI 下周要过来做一次评估。这是评估指南。你要把它读一遍，记住它，然后把它撕碎。它告诉你如何回答 SEI 审计师问你的任何问题。它还告诉你在构建过程中可以使用哪些部分内容以及避免使用哪些部分内容。到 6 月份时，我们会被确定为 CMM3 级机构。”

\* \* \*

你和你的同事开始对新项目进行分析。这很困难，因为你们没有需求。但是，从 BB 在那个决定命运的早上所做的 10 分钟介绍中，你们对于产品应该做什么有了一些认识。

公司的过程要求，开始时要编写一份用例文档。你和你的团队开始列举用例并绘制椭圆图以及参与者标识图。

团队中爆发起来了争论。对于某些用例之间是用《extends》还是《includes》关系连接起来，大家有不同的意见。虽然不同的模型都被创建出来，但是没有人知道如何去评价它们。争论在继续着，明显拖延了进度。

一周后，有人找到一个网站：iceberg.com，上面推荐完全不要使用《extends》和《includes》，应该用《precedes》和《uses》来代替它们。该网站上由 Don Sengroix 所写的文档描述了一个叫做 Stalwart 分析的方法，该方法声称可以逐步地把用例转换成设计图。

使用这个新方案，更多不同的用例模型被创建出来；但是，同样，大家对如何评价它们仍无法达成一致。争论仍在继续。

用例会议越来越多的是被情绪而不是理性所驱动。要不是因为没有需求，你早就会因为没有任何进展而心烦意乱了。

在 2 月 15 日拿到了需求文档。接着，20 日、25 日及此后的每周都有需求文档到来，每次新版本的需求都和前面的有冲突。虽然编写需求文档的市场人员负责此事，但是显然他们没有得到一致的意见。

清晰的商业价值和优先级，倘若他对素材进行了分割，他就会保证这一点。

素材堆积在桌子上。Jay 在编写它们，不过，在需要时开发人员会在上面写上注释。没有人试图去捕获所谈论的每一件事情。素材卡不必捕获所有的东西；它们只不过在会谈中起提示作用。

当开发人员对素材较满意时，他们就开始编写对它们的估算。这些估算很粗糙并且只是一种预算，但是它们却使 Jay 对素材的代价有了一个概念。

会谈结束时，明显还有许多素材可以讨论。同样，也清楚地知道已经明确了最重要的素材，实现这些素材需要几个月。Jay 结束了会议，他带走了素材卡并承诺明天上午会拿出一份关于第一次发布的方案。

~ ~ ~

第二天早上，你又召集了会议。Jay 挑选了 5 个素材卡，把它们摆放在桌子上。

“根据你们的估算，这些素材卡代表着大约 50 个点的工作量。上一个项目的最后一次迭代在 3 周内完成了 50 个点。如果我们可以 3 周内完成这 5 个素材，我们就能够把它们演示给 Russ 看。这样，他就会对我们的进度感到特别满意。”

Jay 在催促着。你从他脸上腩腆的表情可以看出他也知道这一点。你回答道，“Jay，这是一个新团队，从事的是一个新项目。期望我们和前一个团队具有同样的开发速度有点专横了。不过昨天中午我和团队谈过，事实上，我们都同意把最初的速度设定为每 3 周 50 个点。所以在这个事情上你非常幸运。”

“还记得，”你继续说，“现在，有关素材的估算以及设定的速度都是推测出来的。在做计划时我们了解得会多一些，在实现时了解得还要再多一些。”

Jay 透过他的眼镜看着你，好像要说，“在这里到底谁是上司？”接着他笑着说，“好的，不必担心，现在我已经知道了规则（drill）。”

Jay 接着把另外 15 张素材卡放到桌子上。他说，

同时，许多团队成员又提出了一些新的不同的用例模板。每个人都以自己特有的方式来延缓进度。争论愈发激烈了。

3月1日，过程监控人员 Percival Putrignence 成功地把所有不同的用例形式和模板合成为一个单一的包含一切的形式。仅仅空白表格就有15页之多。他把所有不同模板中出现的每一个不同的地方都包含进去。同时，他还提供了一份159页的文档，描述如何填写用例表格。所有当前的用例都要按照新的标准重写。

令你大为惊奇的是，现在要回答“当用户敲击回车键时，系统应该做什么？”这个问题，需要填写15页的表格和问答题。

公司的过程（由 L.E.Ott 制订，他是“全盘分析：软件工程进步辩证法”的知名作者）坚决要求你们必须找出所有的首要用例，87%的次要用例以及36.274%的第3级用例之后，才能算完成分析并进入设计阶段。你们根本就不知道什么是第3级用例。所以，为了满足这个要求，你们就让市场部检查你们的用例文档。也许，他们知道什么是第3级用例。

糟糕的是，市场人员正忙于销售支持而无法和你们讨论。事实上，自从项目开始，你们就没能召开过任何有关市场的会议。他们所能做的最好的就是提供一份不停变化并且矛盾的需求文档。

当一个团队纠缠于无穷无尽的用例文档时，另一个团队在开发领域模型。不停变化的UML文档淹没了这个团队。每周，模型都要重做。团队的成员无法决定在模型中是使用《interfaces》还是使用《types》。关于OCL的适当语法以及应用方面，出现了很大的不同意见。团队中的有些人完全违背了5天课程中关于“分解”的内容，他们创建的图不可思议的详细和晦涩，所有其他人都无法理解。

3月27日，距离分析完成还有一周时间，你们已经产生了大量的文档和图示，但是你们对问题的分析却和1月3日时一样的浅薄。

\* \* \*

接着，奇迹发生了。

“如果我们到3月底能够完成所有这些素材卡，我们就可以把系统移交给我们的beta版测试客户。那么我们会从他们那里得到很好的反馈。”

你回答说，“好，我们已经定义了首次迭代，并且具有了此后下3次迭代的素材。这4次迭代可以完成我们的首次发布。”

“那么，”Jay说，“你们真的能够在接下来的3周内完成这5个素材吗？”

“我确实不知道，Jay，”你回答道，“我们来把它们分解成任务，看看能得到些什么。”

于是，在接下来的几个小时中，Jay、你和你的团队把Jay为首次迭代挑选的5个素材中的每一个都分解成小任务。开发人员很快就认识到某些任务可以在素材间共享，并且其他一些任务具有一些可以加以利用的公共点。很明显，开发人员的头脑中已经出现了一些可能的设计。他们不时地结成讨论小组并在一些卡片上勾勒出UML图。

很快，白板就被任务充满了，一旦实现了这些任务，就完成了本次迭代中的5个素材。你开始了签订过程，说道，“好，我们来签订这些任务吧。”

“我做初始的数据库生成，”Pete说，“在最近的一个项目我做的就是这个，这看起来并不困难。我估计它需要两天时间。”

“好，那么我做登录屏幕。”Joe说。

“噢，该死，”Elmo，团队的一个新成员，说道，“我从来没有做过GUI，我有点想做这一个。”

“哦，年轻人真急躁。”Joe贤明地说，并朝你使了个眼色，“你可以来协助我，年轻人”，他对Joe说，“我认为我需要大约3天完成它。”

开发人员一个接一个的签订了任务并估算了它们。你和Joe都知道让开发人员自愿选择任务要比把任务分配给他们好。你也充分地知道你不敢质疑任何一个开发人员的估算。你了解这些人，并且信任他们。你知道他们会尽最大努力的。

开发人员知道，他们签订的任务不能超过在他们参与的最近一次迭代中所完成的任务。一旦开发人员关于本次迭代的时间表安排满了，他就不再签

\* \* \*

4月1日,星期天,你在家中检查你的电子邮件,看到了一封你的上司发给 BB 的便函。上面明确地写道你已经完成了分析!

你给上司打电话并抱怨道,“你怎么能告诉 BB 我们已经完成了分析呢?”

“喂,你看日历了吗?”他说,“今天是4月1日!”

你没有忘记这个日期的讽刺意味,“可是,我们还有很多问题要考虑,很多东西要分析!我们甚至还没有决定是用《extends》还是用《precedes》!”

“你凭什么说你们还没有完成?”你的上司不耐烦的问到。

“我……”

但是他打断了你,“分析永远也做不完,必须要停在某个点上。因为今天就是计划要结束的日期,所以它就停在今天。星期一,我希望你把现有的所有分析资料收集起来放到一个公共的文件夹中。把该文件开放给 Percival,这样他就可以在星期一中午前把它记入 CM 系统。接下来就开始设计工作吧。”

当你挂断电话时,你开始思考在写字台底部的抽屉中保存一瓶波旁威士忌酒的好处。

\* \* \*

他们举行了一个宴会来庆祝分析阶段的按时完成。BB 发表了一通有关授权的激动人心的讲话。你的上司,另外又增长了 3mm,也祝贺他的团队所表现出的不可思议的团结和团队协作。最后,CIO 登台并告诉大家 SEI 的审计工作进行的非常顺利,并且感谢大家学习并撕碎了所发的评估指南。看来,在6月肯定可以被授予 CMM3 级。

(有传言说,一旦被 SEI 授予 CMM3 级,和 BB 同层以及更高层的管理者就可以得到丰厚的奖金)

几周过去了,你和你的团队一直在进行系统的设计。当然,你发现设计基于的假想分析是有缺陷的……不,毫无用处……不,比无用还糟。但是,当你告诉你的上司你需要返回去再多做一些分析工作以加固分析中的薄弱部分时,他只是说道,“分析

订任务。

最后,所有的开发人员都停止签订任务。但是,当然,白板上仍剩有任务。

“我就担心这会发生,”你说。“好,现在只有一件事情要做,Jay。我们在这次迭代中做的过多了。我们可以去掉哪个素材或者任务呢?”

Jay 叹了口气。他知道这是惟一的选择。在项目一开始就加班工作是非常愚蠢的,并且出现这种情况的项目也不会成功。

于是,Jay 开始去掉最不重要的功能。“嗯,此时,我们还不是真正需要登录界面。我们完全可以在登录后的状态中启动系统。”

“胡说!”Elmo 叫道,“我实在是想做这个。”

“耐心点,急性子(Grassopper),”Joe 说道,“只有等蜜蜂离开蜂箱后,享受蜂蜜时才不会螫肿嘴唇(欲速则不达)。”

Elmo 显得很困惑。

每个人都显得很困惑。

“那么……”Jay 继续说道,“我觉得我们也可以去掉……”

于是,任务列表渐渐地变少。失去任务的开发人员在剩余的任务中又签订了一个。

商谈的过程不是没有痛苦的。其中有几次,Jay 显示出了沮丧和急躁。有一次,当局势特别紧张时,Elmo 自愿要求“超时工作来弥补时间的不足。”当你正打算纠正他时,还好,这时 Joe 看着他说,“一旦你走上了错误的道路,它就会永远控制你的命运。”

最后,终于确定下来了一个 Joe 可接受的迭代。这不是 Joe 想要的。事实上,它比 Joe 想要的要少的多。但是这是团队觉得他们可以在接下来的3周中能完成的东西。并且,毕竟仍然在迭代中完成了 Joe 想要的最重要的事情。

“那么,Jay,”当会谈接近尾声时你说,“你何时能够提供验收测试呢?”

Jay 叹了口气。这是事情的另一个方面。对于

阶段已经结束了。惟一允许做的事情是设计。现在回去设计吧。”

于是，你和你的团队尽最大努力去拼凑设计，不知道是否正确地分析了需求。当然，实际上，这也没有什么大问题，因为需求文档仍然每周都在剧烈地变动着，并且市场部仍然拒绝和你们见面。

设计是一场噩梦。你的上司最近错读了一本书：完成期限（The Finish Line），其中，作者 Mark DeThomaso 轻率地建议设计文档的详细程度应该达到代码级。

“如果我们要达到这个详细程度，”你问道，“那么为什么我们不直接去编写代码呢？”

“因为那样的话，你当然就不是在设计了。而设计阶段惟一允许做的事情就是设计。”

“此外，”他继续说，“我们刚刚购买了一个 Dandelion 的公司范围内使用的许可证！这个工具支持‘round-the-horn 工程’！”你只要把所有的设计图传递给它，它就会为我们自动生成代码！同时，它还会保持设计图和代码的同步。”

你的上司把一个色彩明亮、用塑料薄膜包装的盒子交给你，里面装着销售版 Dandelion。你麻木地接过它，步履蹒跚地回到你的小卧室。12 小时后，你终于将该工具安装到服务器上，安装的过程经历了 8 次崩溃、一次磁盘重新格式化并玩了 8 轮 shots of 151 游戏。你想了一下你的团队参加 Dandelion 培训要浪费的那一周。接着，你露出笑容并想到，“不过在这里度过的任何一周都会是愉快的一周。”

一个接一个的设计图被你的团队创建出来。Dandelion 使这些图的绘制变得非常困难。其中会遇到大量的深层嵌套的对话框，并且必须正确填写这些对话框上的一些滑稽可笑的文本域以及检查框。接着，就会碰到在包之间移动类的问题……

起初，这些图都是来自用例的。但是由于需求的频繁变动，用例很快都变得毫无意义。

关于是否应该使用 VISITOR 设计模式还是 DECORATOR 设计模式的争论爆发起来。一个开发人员拒绝使用任何形式的 VISITOR 模式，声称它不是真正的面向对象概念。另外一个拒绝使用多重继

开发团队实现每个素材，Jay 必须提供一组验收测试来证明它们可以使用。并且团队远在迭代结束前就需要这些验收测试，因为它们会明确地指出 Jay 和开发人员对系统行为认识上的差异。

“今天我会提供给你一些测试脚本的例子，”Jay 许诺道，“此后的每一天，我都会增加一些。到迭代的中期，你就会拥有完整的测试集。”

~ ~ ~

迭代在周一早晨开始了，我们开了一个急速的 CRC 会议。到上午 10 点左右时，所有的开发人员都已经组合成对，并在快速地编码。

“现在，年轻的学徒，”Joe 对 Elmo 说，“你应该学习一下测试优先设计的技术！”

“哇，这听起来相当不错，”Elmo 回答道，“你是如何做的？”

Joe 微笑了一下。显然，此时他已经很想学习了。“年轻人，现在代码做了些什么呢？”

“嘿？”Elmo 回答道，“它根本什么都没有做，还没有代码呢。”

“好，考虑一下我们的任务。你能想起一些代码应该做的事情吗？”

“当然可以。”Elmo 带着年轻人的自信说道，“首先，它应该连接到数据库。”

“那么，要连接数据库，必需的东西是什么呢？”

“你说话真是古怪，”Elmo 笑着说，“我认为我们必须要从某个注册表（registry）得到数据库对象，并调用其 Connect()方法。”

“哈。敏锐的年轻奇才。你正确地觉察到了我们需要一个对象，在该对象中我们可以缓存（cacheth）数据库对象。”

“‘cacheth’是一个真实的单词吗？”

“在我说出它时，它是！那么，我们可以编写哪些我们认为数据库注册表应该通过的测试呢？”

Elmo 叹了口气。他知道他必须得合作下去。“我们应该能够创建一个数据库对象并用 Store()方法把

承，因为它会带来麻烦。

评审会议很快就变成有关面向对象的意义、分析和设计的定义以及何时使用聚合和关联的争论。

在设计周期的中间，市场人员宣称他们重新考虑了系统的中心内容。他们彻底重新组织了一份新的需求文档。他们去掉了一些主要的特性范围，取而代之的是一些他们从客户调查中预见的更合适的特性范围。

你告诉你的上司，这些变更意味着你需要对系统的大部分内容进行重新分析和重新设计。但是他却说，“分析阶段已经结束。惟一允许做的事情是设计。现在回去设计吧。”

你建议创建一个简单的原型展示给市场人员，或者甚至是一些潜在的客户看一下，可能会好一些。但是你的上司却说，“分析阶段已经结束。惟一允许做的事情是设计。现在回去设计吧。”

拼凑、拼凑、拼凑、拼凑。你设法创建了某种也许会真实反映新需求文档的设计文档。但是，需求的彻底更改并没有导致它们停止变动。事实上，如果说有的话，就是需求文档的疯狂变动只是在频率和幅度方面有所增加。你在它们的包围中艰难的前进着。

6月15日，Dandelion 的数据库遭到了破坏。显然，破坏是逐步形成的。数据库中的小错误在几个月内累积成越来越大的错误。最后，CASE 工具完全停止工作了。当然，逐步形成的破坏在所有的备份中都有出现。

给 Dandelion 的技术支持人员打了几天的电话，都没有得到任何答复。最后，你收到了一封来自 Dandelion 的简短的电子邮件，通知你这是一个已知的问题，解决办法就是购买新的版本（他们承诺新版本在下季度的某个时候可以使用），然后手工重新输入所有的图。

\* \* \*

接着，7月1日，另一个奇迹发生了！你完成了设计！

这次，你没有去见你的上司并抱怨什么，相反

它传递给注册表。然后，我们应该能够使用 Get()方法把它从注册表中取出来并证实它就是上一个对象。”

“哦，说得真好，我的年轻捣蛋鬼！”

“嗨！”

“那么，现在，我们来编写一个测试函数来检验你所说的情形。”

“但是，我们不应该先编写数据库对象和注册表对象吗？”

“啊，你还有许多东西需要学习，没有耐心的年轻人。先编写测试。”

“但是这甚至无法编译！”

“你肯定呢？如果可以编译怎么办呢？”

“嗯……”

“先编写测试，Elmo。相信我。”

于是，Joe、Elmo 以及所有其他的开发人员都开始编写他们的任务，每次一个测试用例。他们工作的房间中充满了结对人员之间交谈的嗡嗡声。嗡嗡声不时被高呼声打断，这些高呼声是某一对人员完成了一个任务或者通过了一个困难的测试用例时所发出的。

在开发的进行过程中，开发人员每 1 到 2 天就更换结对伙伴。每个开发人员都会了解所有其他人做的东西，因此关于代码的知识就广泛地在整个团队中传播。

每当一对人员完成某个重要的东西，不管是一个完整的任务或者仅仅是任务的一个重要部分，他们都会把完成的东西和系统的其余部分集成起来。这样，代码基每天都在增长，并且集成的难度被减至最小。

开发人员每天都和 Jay 进行交流。每当他们对系统的功能或者验收测试用例的解释有疑问时，都会去找 Jay。

Jay 很好的履行了他的诺言，平稳持续地为团队提供验收测试脚本。团队用心地理解这些脚本，

你在写字台中间的抽屉中放入了一些伏特加酒。

\* \* \*

他们举行了一个宴会来庆祝设计阶段的按时完成，以及通过了 CMM3 级认证。这次，你发现 BB 的讲话非常的煽情，因此你只好躲到休息室去。

在你工作的地方遍布着一些新的标语和牌匾。上面显示着鹰和登山者的图案，并且写着关于团队协作以及授权方面的内容。上面增加了一些方格线后，辨认起来好多了。这让你想起你需要在你的文件柜中腾出点地方来放白兰地。

你和你的团队开始编码。但是你很快就发现设计在一些重要的方面存在不足。实际上，它在所有重要的方面都有缺乏。你在一个会议室中召集了设计会议，试图解决一些严重级别高一些的问题。但是你的上司在会议室中抓住你并解散了会议，说，“设计阶段已经结束。惟一允许做的事情是编码。现在回去编码吧。”

Dandelion 生成的代码实在是丑陋。你和你的团队终究还是误用了关联和聚合。为了改正这些错误，必须得编辑所有生成的代码。编辑这种代码异常困难，因为它上面被添加了一些具有特殊语法的丑陋注释块，Dandelion 要使用这些注释来保持图和代码之间的同步。如果你不小心更改了某个注释，那么重新生成的图就会不正确。结果表明，“round-the-horn 工程”还需要非常多的工作要做。

你越是想保持代码和 Dandelion 兼容，Dandelion 产生的错误就越多。最后，你放弃了这种做法，并决定手工地使图保持最新。一秒钟后，你发现使图保持最新根本没有意义。此外，以谁的时间为准呢？

你的上司雇佣了一个顾问来构建一个计算所编写的代码行数的工具。他把一张很大的坐标纸贴在墙上，在顶部标出了数字 1 000 000。每天他都会延长红线来显示增加了多少行代码。

贴出坐标纸 3 天后，你的上司在大厅里拦住你。“那张图增长的不够快。我们要在 10 月 1 日时完成 100 万行代码。”

“我们还不确信该产品会需要 100 万行代码，”

从而对 Jay 期望系统做的东西有了更好的理解。

到第 2 周初时，所完成的功能已经足以演示给 Jay 看。Jay 热切地观看着，演示通过了一个接一个的测试用例。

“这真是太棒了，”当演示最后结束时，Jay 说道，“但是这看起来好像不到 1/3 的任务。你们的速度比预期的慢吗？”

你皱起眉头。你本来想等待一个合适的时机把这告诉 Jay，但是现在他却提前提出了这个问题。

“是的，很遗憾，我们比期望的要慢一些。我们使用的新应用服务器配置起来很费劲。而且，还得常常重新启动它，每次即使我们对它的配置做了最微小的更改，都必须得重新启动它。”

Jay 用怀疑的眼光看着你。上周一商谈中的紧张状态还没有完全消散。他说，“那么，这对我们的进度意味着什么呢？我们不能再落后进度了，绝对不能。Russ 会很生气的！他会惩罚我们所有人，并为我们增加一些新人手。”

你一直看着 Jay。这样的消息是没有办法以令人愉快的方法说出来的。于是你完全不加思索的说道，“看，如果事情还像这样进行下去，那么到周五时，我们将不能完成所有的东西！现在我们是有可能找出一条可以快一些的方法的。但是，坦白地说，我不会依赖于它的。你应该考虑一下从迭代中去掉 1 个或者 2 个任务，而又不破坏给 Russ 的演示。无论如何，我们都会在周五进行演示的，并且我认为你不会想让我们来挑选去掉哪些任务。”

“啊，看在上帝的面子上！”当 Jay 摇着头大步离开时，几乎无法抑制住喊出最后一句话。

不止一次，你对自己说，“从来没有人敢向我保证项目管理会是容易的。”你非常肯定这也不会是最后一次。

~ ~ ~

实际的情况要比你期望的稍好一点。事实上，团队确实从迭代中去掉一个任务，但是 Jay 做了明智地选择，所以给 Russ 的演示很顺利。

Russ 对进度没有太深的印象，但是他也没有感

你急着说。

“我们必须在 10 月 1 日时完成 100 万行代码，”你的上司重复着。他的发尖再一次增长了，并且他在它们上面使用的希腊式配方营造出一种权威和能力的氛围。“你确信你们的注释块足够大吗？”

接着，他立刻闪现出了管理方面的洞察力，说，“我知道了！，任何一行代码都不能超过 20 个字符。任何超过 20 个字符的代码行必须得分成两行或者更多的行——越多越好。现有的所有代码都必须按这个标准改写。这会使我们的代码行增加！”

你决定不告诉他这需要 2 个计划外的人月。你决定根本不告诉他任何事情。你觉得静脉注射酒精是惟一的办法。你做了适当的安排。

拼凑、拼凑、拼凑还是拼凑。你和你的团队疯狂地编码。到 8 月 1 日，你的上司皱着眉看着墙上的坐标纸，制定出了强制性的每周要工作 50 小时。

拼凑、拼凑、拼凑还是拼凑。到 9 月 1 日，坐标图显示代码有 120 万行，你的上司让你写一个报告描述一下你们为何超出代码预算 20%。他制定了强制性的周六加班，并要求项目代码减少到 100 万行代码。你们开始着手对代码进行重新合并。

拼凑、拼凑、拼凑还是拼凑。脾气变得暴躁；人员一个一个地辞职；QA 把大量的故障报告发给你。客户在要求安装产品以及用户手册；销售人员要求给特殊的客户进行一些预先的演示；需求文档仍然在变动；市场人员在抱怨产品根本不是他们所要的，卖酒的店铺也不再卖给你酒了。必须要交出一些东西了。9 月 15 日，BB 召开了一次会议。

当他走进会议室时，他的发尖散发着朦胧的雾气。当他说话时，他精心修饰过的低音致使你的胸口要翻转过来。“QA 的管理人员告诉我，这个项目只实现了不到 50% 的必需的特性。他还告诉我系统总是会崩溃，产生错误的结果，并且非常慢。他还抱怨他无法跟上连续的每日发布，每次发布都比上一次出现更多的错误！”

他停顿了几秒，明显想镇定一下。“QA 的管理人员估计，像这样开发下去，要到 12 月份，我们才能够发售产品！”

到沮丧。他只是说，“相当好。但是记住，我们必须能够在 7 月份的展览会上进行演示，如果以这样的速度的话，看起来你们不能完成所有的要展示的东西。”

Jay 的态度在迭代完成后有了很大的改善，他回答 Russ 说，“Russ，这个团队工作的很努力，也很好。到 7 月时，我确信我们会有一些最重要的东西去演示。它不是所有的东西，并且其中的一些可能没用真正实现，但是我们会有一些东西的。”

虽然刚刚结束的迭代很费劲，但是它却校准了你们的开发速度。接下来的迭代好了许多。这并不是因为你的团队完成了比上一次更多的任务，而只是因为不必在迭代的中期去掉任何的任务或者素材。

到第 4 次迭代开始时，一个自然的开发节奏被建立起来了。Jay、你以及开发团队都可以准确地知道彼此期望的是什么。虽然团队工作的很艰苦，但是开发速度却是可持续的。你确信团队能够保持这个速度一年或者更长的时间。

在进度方面几乎没有出现什么问题；但是在需求方面却非如此。Jay 和 Russ 常常检查逐渐增长的系统并对现有的功能提出一些建议和更改。但是任何一方都知道这些更改是花费时间的并且必须要被列入计划。因此，更改没有导致对任何人期望的违背。

在 3 月份，给董事会做了一个该系统的较大型的演示。系统功能非常的有限，还不足以拿到展示会上去演示，但是进展却非常稳定，给董事会留下了相当深刻的印象。

第 2 次发布甚至比第 1 次还要顺利。现在，团队已经找到了一个可以自动执行 Jay 的验收测试脚本的方法。他们同样也对系统进行重构，直到确实可以容易地向其中增加新特性以及更改原有的特性。

到 6 月底完成了第 2 次发布，并被拿到展览会上。系统的功能要比 Jay 和 Rus 原本想要的少一些，但是它确实演示了系统最重要的特性。虽然展览会上的客户注意到了某些功能没有实现，但是在总体上却给他们留下了深刻的印象。你、Russ 以及 Jay

事实上，你认为更可能在明年 3 月，但是你什么也没有说。

“12 月！”BB 吼叫着，面带着嘲笑，每个人都低下头就好像他正用一只突击步枪对准自己一样。“12 月是绝对不行的。团队领导们，我希望明天上午在我的办公桌上见到新的估算。因此，我要求每周工作 65 小时直到这个项目完成。最好能在 11 月 1 日完成。”

当离开会议室时，就听他嘀咕道，“授权——呸！”

\* \* \*

你的上司秃顶了；他的发尖被安放在 BB 的墙上。荧光灯照在他的头顶所反射的光很快使你眼花。

“你这儿有喝的东西吗？”他问到。刚刚喝完你最后一瓶 Boone's Farm，你又从书架上取下一瓶 Thunderbird 并倒入他的咖啡杯中。“怎样做才能完成这个项目？”他问到。

“我们需要冻结需求，分析它们，设计它们，然后实现它们。”你麻木地回答。

“到 11 月 1 日？”你的上司怀疑地大叫到。“不！赶快回去编写这该死的东西。”他抓着他那光秃秃的脑袋气冲冲的走了。

几天后，你发现你的上司被调到公司研究部门。销售量大幅地增长。客户一知道他们的订单无法按时完成，就立即要取消他们的订单。根据市场情况，又对该产品是否符合公司的总体目标进行了评估，等等，等等。信函乱飞，人员被免职，政策改变，总的来说，事态变得相当严峻。

最后，到 3 月份。经过了大量的 65 小时工作周后，一个非常不可靠的版本完成了。实地使用时，错误的出现率非常高，技术支持人员对于发怒的客户的抱怨和要求束手无策。所有人都不高兴。

4 月，BB 决定通过购买的方式来解决，他购买了由 Rupert 工业公司开发的产品的使用授权并重新销售。客户的怒火被平息了，市场人员沾沾自喜，而你被解雇了。

在从展览会上返回时都面带笑容。你们都仿佛觉得这个项目是一个胜利者。

事实上，许多个月以后，Rufus 公司和你们进行了联系。他们曾经为了他们的内部业务开发过一个类似的系统。经历过一个死亡的项目后，他们取消了这个系统的开发，并和你们商谈有关在他们的环境中使用你们的技术的授权许可事宜。

情况确实在不断变好！