## 第9章 变得成熟：Scrumban如美酒，越陈越好

本章中涉及：

* 优先级排序和管理变更
* 常见的提升机会
* 超越团队，放大能力（组织级的敏捷力）

*观察并非只依赖于人们看到的东西，毕竟看到什么源于如何去看；所有的观察不仅仅是接受或发现，也应该是一种产出，而且从某种程度讲，决定因素在于观察者本身具备的能力。*—— Soren Kierkegaard

===

教练小贴士

充满热情很好，但不要急于改变。首先掌控你正在做的事情。

===

之前章节我们提到过一些可见的实践，工具和技术——即相对容易观察和衡量的实践——实际源于一些不可见的力量。对组织而言，已经发布的产品或服务能否与其在市场上体现出的价值有效匹配，最终是由这些不可见力量决定的。为了能运转在最高绩效的水平上，我们面临的挑战是识别并管理这些不可见的力量——针对这一点，Scrumban可以提供帮助。

Scrumban受精益原则的深远影响，也立足于一种长期策略。它倾向于根据长远的发展制定决策，即便有可能牺牲短期利益。从流程观点看，Scrumban选择通往高绩效的道路是——首先最大化流的效率，然后从实用主义的角度优化资源使用。由于管理资源的需求总会存在，控制流就成为资源优化必需的前置条件。

Heavily influenced by Lean principles, Scrumban is a long-term strategy. It is biased

toward basing decisions on a long-term view, even if it means sacrificing short-term

gains. From a process standpoint, its chosen path to high performance is to first

maximize flow efficiency, then to pragmatically optimize the use of resources. While the

need to manage resources is always present, taming flow is a necessary precondition for

resource optimization.

**A014-1 优先级排序和管理变更**

如果你希望达到高绩效，就算无法达到完美，那也要为学习和追求“完美”持续灌输很强的驱动力。Scrumban通过务实的方法来支持这些努力，强调使用实践性实验来探测未知。

如果你想达到高性能, 那么你需要灌输一个强大的学习动力和追求 "完美", 虽然完美永远无法实现。

Scrumban 通过强调探索未知的实际实验的务实方法来支持这些努力。

因为Scrumban可以很容易与其他框架和模型（比如A3思维和Cynefin）整合，我们可以用它创建一些自律的方法。这些方法不但能在组织内推广，也可以验证当前的理解（或令其失效），并产生新的见解。通过每次实验，这些学习成为下一个持续循环的基础。

If you want to achieve high performance, then you need to instill a strong drive for

learning and the pursuit of “perfection,” although perfection can never be achieved.

Scrumban supports these efforts through a pragmatic approach of emphasizing practical experiments that probe the unknown.

Because Scrumban easily integrates with other frameworks and models (such as A3

Thinking and Cynefin), we can use it to create disciplined approaches that scale across

an entire organization and that validate (or invalidate) current understandings and

generate new insights. With each experiment, the learnings form the basis for the next in a continuous cycle.

Scrumban也是一种人员策略，核心假设是，如果人们不能发挥出自己最强的能力，那么是系统的问题，而不是人的问题。最好的改进建议应该源于身处其中的人，由于坚信这一点，Scrumban按照最大化学习的方式构造。

如果业务领导和管理者期望在自己的组织中可以最大化绩效，那么就应该为团队提供不同的方法，切实提高他们对当前正在工作的系统的理解，并允许他们进行改进。Scrumban提供了框架，让这一切成为现实。不过和其他框架类似，没有什么是可以简单实现并马上投入使用的——实践者仍需要三思而后行。

**A014-1-1 管理成熟度**

在改进之前，团队应该具备一些最起码的能力。最基本的包括需要理解正在发生的事情以及应有的目标。随着团队的成长和逐渐成熟，持续评估他们在以下各方面的能力同等重要（如果不是更重要的话）。

* 把团队能力的改变和趋势可视化
* 引导团队将精力集中在产生最大影响力的事务和方向上
* 度量团队付出所产生的影响
* 激励团队进行更多的改变和改进

刚开始，会有一些相当明显的简单经验法则。比如那些需要在自己的环境下获取控制权的团队通常会关注可视化；工作中没有正式流程的团队通常关注政策定义；而很难管理在制品（WIP）的团队通常会关注限制在制品的数量。

随着团队能力的逐渐成熟，只关注一两个原则而忽视其他原则不会带来任何帮助。不管付出多少努力，寻找“表面”原则的成本，必定少于在技能方面逐渐加强所付出的成本。如果这些领域总会互相影响，那么团队需要在各个方面平衡他们的能力。在这点上，Scrum和Scrumban没有那么不同，尽管Scrumban较少依赖于保持能力级别，驱动和支撑商业价值的增强发布。

为了更好的理解如何最好的培养这些能力的成熟度，第一步就是拥有适当的机制来评估过程和能力。让我们来回顾一下团队在这方面可以采用的一些范例。

**A014-1-2 飞行高度**

把Scrumban作为一个概念进行介绍时，很容易认为它的原则和实践受限于开发团队。但本书中，我一直想尽力说明，Scrumban的原则和实践能够也应该扩展到组织的不同层级。给组织中更多的人介绍这些原则的时候，需要不同的方法和技能。

我发现Klaus Leopold开发的范例——Kanban方法的飞行高度很有用。他创造出的这个范例，可以作为一种沟通方式，说明在超越独立团队边界时，为什么使用Kanban方法会更加有效。其定义的四种“飞行高度”应用在Scrumban框架中也同样贴切。

* 第一级：无输入协调：组织中的一个单元比如一个团队或者部门中应用Scrumban，常见特征是高度专业化的人才在一起工作，例如为一个很小的产品或者子系统工作的跨职能的团队。在这个高度，工作是“从栅栏外扔进来”的，而且这些工作没有进行协商和协调。组织单元通常为大量干系人服务，通过一个可视化看板来协调那么多干系人的优先级活动根本就是痴人说梦。在该级别实践Scrumban的组织单元的另一个普遍的特性是，他们经常需要处理大量紧急的工作，尤其是干系人认为很重要的变更。

Scrumban可以在组织单元内提高效率，但是缺乏输入协调意味着，很多时候只是在更有效率德做“错误的事情”而已。这样的改进会对组织的整体业务发布产生负面影响。

从理论上讲，现有的Scrum团队不应该处于这个级别。产品负责人的角色，产品代办列表，Sprint计划会议和相关的Scrum元素，都应该提供大量输入协调。但是跟不能通过书的封面来判断一本书一样，重要的是深入表面，探求本质。

* 第二级：协调好的输入：Scrumban仍然应用并集中在组织单元内，但是在这个级别，团队会有花一些精力进行协调，限制进入系统输入队列的工作数量。Scrum的现有框架实际上也发布了和这个等级相关的很多能力，不过这些能力在不同的单元之间也是不同的。
* 第三级：价值流：对某些现有Scrum团队而言，这个级别可能是第一个具有新的重要能力的级别。本级别的目标不是单独的团队或者部门，而是系统级别的价值流部分，这需要多个组织单元（比如团队，部门）之间的协调。理想的状态是，在这个级别，组织运转会关注在整体价值流方面，而不只是某个独立的部分。
* 第四级：扩展的Scrumban：这个“项目集”——或者叫“程序”——级别的实现包括了跨团队，单元和部门的所有输入的协调以及业务发布。一般来讲，不同项目和产品的价值流不尽相同，在这个级别的工作项通常根据功能来定义（比如概要描述通常作为“传奇”来参考）。这些工作项会在其他各个级别被分解成更小的部分（史诗，用户故事或者需求）。

本级别使用Scrumban的主要好处是提供一个框架，用来管理那些竞争时间和资源的相对重要的复杂工作。Scrumban的实践迫使组织在优先级排序方面做出更清醒的决定，并重视需求和能力之间的平衡。在这个级别，组织学会基于风险来决定下一步完成什么，而不是立即开始所有的事情（这么做通常什么都完成不了）。

Leopold还用键盘做了一个绝佳的比喻，来说明为什么不能把Kanban/Scrumban当做一种框架，因为框架概念会局限团队，让他们只关注帮自己提高绩效。

*把你的公司想象成一个键盘，公司中的每个团队负责某一个键。你的客户希望你帮助他们写情书。只要愿意，你的“一个团队”可以尽力优化到最好最快的打出字母“A”。但是他们优化的结果并不能使客户的情书更快的完成，因为很多其他团队在生成最终的系统输出时也很重要。可以把这个比喻用到你自己的组织。以什么样的质量和速度发布哪些价值给最终用户，对这一点产生影响的不仅仅是独立管理一个或两个团队的绩效，而是如何管理不同团队间的交互。*

不管是你和你的团队自己实施Scrumban，还是花精力协调组织内部采用Scrumban的实践，这些飞行高度的概念可以帮助沟通，用来解释为什么获取最高等级的商业绩效最终需要整个组织都采用类似的想法和能力。

**A014-1-3 度量能力深度**

在任何组织中，以下场景都很常见，一些团队可以掌控自己的改进，而另一些在基本线上苦苦挣扎。理解团队在运用核心原则和实践上的深度，有助于决定以下因素：

* 团队已经实现的每个核心原则的程度
* Scrumban实践在管理工作和创造更好的商业价值方面产生的影响
* 团队应该将其精力放在哪些原则上。

Chrstophe Achouiantz和Johan Nordin开发了一个简单有效的方式，能够评估并可视化一个单元的“Kanban深度”。他们的工作基于David Anderson, Hakan Forss和Mike Burrows的素材。

Achouiantz和Nordin的方法包含了一系列关于每个Kanban核心实践（可视化，限制WIP，管理流，策略明晰化，实现反馈环以及寻求持续改进）的问题，这些问题只需要回答“是/否”。同时，包含了一个评估模块来测量这些实践的影响程度（图9.1）。评估的结果展示在以每个核心实践作为一个坐标轴的雷达图上（图9.2）。

（下面表格是图9.1里面的文字翻译）

|  |  |  |
| --- | --- | --- |
| **可视化:**  **1. 工作（所有的，根据当前政策）**  **2. 工作种类**  **3. 工作流（“流程”，工作方式，价值流）**  **4. “下一步”和“完成”**  **5. 目前团队的关注点（头像）**  **6. 障碍物**  **7. 当前政策（Dod，DOF，能力分配等等）**  **8. 等待拉出的事项（在工作流/列中“完成”的）**  **9. 衡量矩阵（前置时间，本地周期时间，SLA目标等）**  **10. WIP的限制**  **11. 工作中的依赖项（层级，父子关系等）**  **12. 工作流内的依赖**  **13. 风险规模（推迟成本，技术风险，市场风险）** | **限制在制品**  **没有WIP的限制，但是承诺完成工作而不是开始新的（最终可以达到一个令“团队”满意的WIP的水平）**  **有一些明显更低等级的WIP限制，而不是在工作流等级（也就是Proto-Kanban）；个人看板，限制每个人的WIP，限制某些列或者泳道的WIP，工作流中具有无限限制的“完成”队列等等。**  **工作流级别的明确的WIP限制——单一工作流的完全拉动**  **拉动系统中多个互相依赖的工作流** | **管理流**  **每日计划会议（如果政策允许，尽量“每日”）**  **提出超出团队控制的障碍寻求解决方案**  **下一步是持续重新排列优先级（下一步没有承诺）——推迟的拉动决定（动态排列优先级）**  **CFD（最起码一周更新一次）**  **控制图表（最起码一周更新一次）**  **限制正在进行中的工作项的大小（大的工作需要进行拆分）**  **灵活的人员安排（分群）**  **具备节奏感（计划，发布，回顾）**  **流的测量矩阵（阻碍的天数，前置时间的效率）**  **SLA的期望和展望（前置时间的目标）**  **容量分配** |
| 政策清晰化——所有的政策都必须是当前的（为人所知并且真正使用）  1. 工作类型和工作项定义（模板）  2. 如何拉动工作（从“下一步”/根据WIP的优先级进行选择）  3. 谁在什么时间管理“下一步”和“完成”队列  4. 人员安排/工作分配（关注个体）  5. “下一步”的准入定义  6. 谁在什么时候如何评估工作量  7. 限制工作项的尺寸（工作拆分）  8. 如何为“下一步”队列选择&准备工作  9. 所有步骤的完成定义（看做一个目标条件）  10. 知识扩展/分享策略  11. 业务的分类  12. 能力分配 | 实现反馈环  团队每日站会  关键干系人（管理者，客户，其他组）会定期更新当前状态  管理者定期审视（够做’gemba’）  与上下游的合作者定期讨论  定期展示和讨论财务方面的绩效  定期展示和讨论质量KPI（缺陷率，客户满意度等）  “定期”意味着一月一次或多次 | 改进  团队明白其存在的原因以及成功的条件  定期回顾/kanzen  团队清楚当前的状况（可能需要测量矩阵）  True North是真实存在的，有团队进行沟通和共享  团队了解当前目标的状态（挑战）  对当前目标状态，有有效的准入原则能够知道目标是否达成  团队知道有哪些障碍物阻碍他们达成目标  退队知道哪些障碍物当前已经确认  团队知道解决当前障碍的下一个步骤  团队会审视采取这一步骤他们能够学到什么 |
| 影响（看见的相关证据）  团队成员可以看见并理解整个项目（团队级别 vs.当前状况）  更好的“团队精神”（互相帮助来完成工作，尊重）  关注于移除障碍  关注于完成工作而不是开始新的工作  工作不超出团队的容量（限制压力，优化前置时间）  团队有动力去驱动改进  当前过程的演进（可视化，工作流，政策，WIP限制等）  增加Kanban实现的深度  过程改进是模型驱动的  过程或者管理策略改进是随之而来的结果  工作流之间或者管理策略的改进源于回顾操作流程 |  |  |
|  |  |  |

图9.1 简单 是/否的问题来评估团队在重要实践的六个方面能力 来源：Christophe Achouiantz

评估一个Kanban实现的深度

图9.2 领域经验帮助这张雷达图的作者来定义能力范围，识别团队何时在绩效关键领域运行

Achouiant首先把这个工具应用在一个正在使用Kanban的组织，他们使用Kanban的主要目的是为了其IT团队用可持续的方式进行持续改进。为了达到这个目标，这些团队必须具备最小等级的能力才能启程——尤其是清晰的执行力和清晰的目的。Achouiant最终在这个框架里量化了这些能力，并发现在沟通概念和捕获趋势方面，他们可以提供非常有价值的参考。

这个雷达图中心最暗的区域反应了Achouiant对一个团队所需“核心能力集”的评估，团队需要满足“核心能力集”，才能开始提升自己。这种方法确保Achouiantz和Nordin使用的快速启动过程（在第五章中有建模过程）能够影响团队在“持续提升”内不断运转。

从管理成熟度的角度出发，与其理解核心实践的“深度”，不如把这个过程用作一个框架来激发讨论。举例来说，简单的过一遍评估的过程就可以激发提高团队工作方式的讨论。类似的，可视化这些能力可以帮助团队，让他们认识到只在一方面提高是不够的，还需要把他们的能力平衡在其他方面。

图9.3 在雷达图上绘制出一个团队的能力，可以帮助团队（和团队的教练）对可以提升的地方排出优先级。

粗略看一下图9.3就能发现这个团队在Kanban的大部分核心实践上有相当大的提升，但是在限制WIP这个方面提升有限。

最终，每个团队应该决定是否关注于某些原则放弃另外一些。团队成员必须谨记变得更“深入”所需的精力一定大大超过刚开始达到的那些低级别目标花费的精力。另外团队还需要明白这个事实：就是这些原则会互相影响，在不同方面付出不同比例的努力，不是只沿着一个方向提升，反而能够获取最好的回馈。

#### 一些警告

如果用作基础比较或者跨团队使用，类似Achouiantz和Nordin模型之类的评估框架的结果可能没什么用或者令人困惑——尤其是试图将这个模型“不加修饰”的用在某个环境又不为其量身定制。反之，该框架应该只用在为了更清楚的了解团队随时间不断提升的方面。它能够帮助回答诸如“团队是否变得更好了？”和“为什么是或为什么不”等此类问题。

用这种方法，Achouiantz和Nordin的框架和用户故事点以及团队速度就有了同样的特性；速度与每个团队相关，更相关的是随着时间推移，定义出模式。“实践深度”评估的最佳使用就是激发团队级别讨论Scrumban的不同方面，以及团队用什么方式帮助自己提升能力。

**A014-1-4 另一种方法**

Mike Burrows 还将Kanban深度评估工具继续升级，认为与其使用核心原则和实践，不如把Kanban方法的价值当做晴雨表。他的工具使用4个等级的分数来衡量与核心价值相关的一些评估问题（见图9.4）。

图9.4 这个团队的评估工具关注于团队在实践核心价值时的程度，与具体实践相比较

比起根据简单的实践相关的检查列表进行评估，这种方法的确看起来有进步。不出意外的话，Kanban社区的领导者们会继续进行这项工作，所以Scrumban实践者可以用这些模型来建立和改善那些对他们最有用的方法[1]。

[1]想获知此工具的更多详情，请访问<http://tiny.cc/Rebooted> 以及Mike的著作 Kanban from the Inside(Michael Burrows，Blue Hole Press，2014)的第23章。

**A014-2 不断进化你的核心能力**

纵观全书，我已经给出很多方法让团队和组织用来持续改进他们的实践。Scrum Master可以作为辅助角色来提供建议，并在团队应该如何进行实验方面给出指导——不过同时要记住的是Scrum Master们是引导者，需要变成管理工作的人，而不是管理人员。

牢记这一点，让我们回头来看，随着对Scrumban理解的日趋成熟，团队可以从事的一些核心机制。我们从管理流动开始，因为这是首要且长期的优先事项。

**A014-2-1 管理流动**

当团队在层流（或平滑）状态下运转时，团队的效率是最高的，而且通常可以最大程度的控制他们的工作。层流是一种期望的状态，因为它使得系统更可预测。

Scrum试图通过强调创造真实，跨职能的团队来优化流动。但是并非所有的组织都已经准备好接受这一结果。不过，在不具备跨职能能力的时候，Scrumban可以让我们开发更灵活的能力，以便于更好的吸收多样性（虽然它可以进行资源优化，但是并不能让紧张的状态神奇消失——更多的是在某一时刻发挥作用）。这里有一些直接的方法，包括下列技术：

* 识别不同的工作种类（以便更好地管理多样性）
* 使工作搭配清晰可见——包括各种相关的风险（更好的排出优先级）
* 设计系统，允许其用某种方式拉动工作，这种方式需要在每种工作之间保持可预测性（使我们能够识别和解决瓶颈）
* 在策略中（优先级规则，限制，时间缓冲和其他约束——隐藏的策略会影响流动管理！）捕捉设计。

不管团队在基本的Scrum实践中达到什么样的精通程度， Scrumban框架允许团队成员探讨多种选择，寻求更多的层流。这些专业人士可能会从Kanban系统的基础开始，并最终进入管理更多的选项和特征，如使用具体的工程实践，或者扩大团队的技能。

为了这个目的，管理各种工作项类型的Kanban系统，通过服务的类型进行工作的管理，结合频繁的系统需求输入（sprint代办列表/就绪队列），并实现频密交付，自然会表现出较高的流动效率。因此，随着时间的推移，团队和组织应该设法获得并提升这些特征。

关于何时优化流动效率，一个主要的复杂因素是，在天然的商业压力下，团队需要得到一切——也就是说寻求与流动效率并行的资源优化。这种压力归结为经营战略，其基本原则可以帮助平息这场争论，而且Scrumban也基于这个基本原则。在这种情况下，强调以下事实很重要，也就是虽然我们最终要同时优化， 但是Scrumban会把优化流动效率排在第一位，高于资源优化。因此，应该鼓励企业支持这一优先顺序，并将其作为实现最终期望状态的一种手段。

很难同时实现高流动效率和高效资源的原因之一是，需要与知识工作的自然变化（例如，变化的到达时间，大小和工作的复杂性）相抗衡 。根据所处的环境，我们可以管理其中的一些变化，无需担心失去创新和适应市场的能力。可以通过实施以下策略达到这个目的：

* 减少批量规模
* 减少交接
* 使用混合的功能组合，包括不同优先级和大小的工作项
* 减少迭代的长度或者完全消除迭代
* 避免在具体项目内完成工作的诱惑（在我们日常工作流程中，项目和早晚高峰的交通拥堵类似）
* 避免使用那些支持负载均衡，蜂拥或者建立临时的进程的机制或者流程（所以更容易发现天然的瓶颈或其他问题，并解决它们）

敏捷团队提高流动效率的另一种方法是把他们的团队合并成较大的部门，这样所有的团队成员都可以工作在所有业务的单一工作流内，有多个需求来源，跨越不同的工作项类型，并使用不同类型的服务来处理各种风险，不需要批传输（时间盒迭代）。日本人把这种理念称为Heijunka（均衡化系统）。

=====

实践小贴士

**警告**：对于很多Scrum环境而言，Heijunka可能不是一个可行的选择。在实践Heijunka时，团队和组织应该确保他们正在增加的流动效率是在客户承诺和实际发布的点数之间， 而不只是在sprint内部通过衡量流动来进行人为的增长。

====

均衡化系统的目的是使得在代办列表中的是产品的合理组合。这是一个期望结果，因为单一工作项类型的流是不能立即把最终产品发布给客户的（而且在一定程度上，推动也是必需的）。时刻做好准备去处理困难的事情，比如在单一系统中混合工作项类型，可以让我们学习如何顺利的工作。从均衡化系统可以很容易的看到，流动效率和优化资源之间的竞争会使得实现层流的过程极具挑战。

**A014-2-2 限制WIP**

即使做了很多可视化方面的努力，限制WIP也并不容易，因为仍然有很多工作是不可见的——在它们出现之前，我们会想当然的对其视而不见。通常在限制WIP中很容易忽视的一个方法是：让隐藏的WIP可见。但是怎么样才能找到隐藏的WIP呢？

答案其实非常明显：就是总是假设有隐藏的WIP存在，然后去寻找藏在“阴影”下的它们。三个常见的地方可以找到隐藏的WIP：

* 尺寸非常大的工作项：如果用户故事都非常大，那就很有可能隐藏了一些WIP。这些隐藏的工作自身可能会是黑暗面的（随着我们理解的增加，工作会自然显现）或者需要重做的（无法很好的理解工作而产生的缺陷）。由于一个或者多个隐藏的任务很难完成，用户故事可能会陷入泥潭，停滞不前。
* 过于关注团队而不是个人：在采用Kanban/Scrumban实践时，限制团队的WIP是对团队产出的首要改变。但是，这个经常会导致在整个流中没有考虑个人WIP的影响。比如，有一个或两个团队成员参与了所有的用户故事，一个团队仍然有可能“成功地”工作在五个或六个工作项的WIP限制下。尽管团队成员可能可以完成工作，但这是超负荷运转达到的限制WIP的目标。所以请关注白板上的现实。
* 未完成的库存：在确定为什么特定的工作项不应该放在白板上时，我们都有各种说辞。一个常见的例子就是开发团队的看板上会忽略多方支持的任务，因为他们认为这些任务不是“真正的工作”。类似的，有很多行政管理工作的团队也不会将这些任务可视化，因为它们不是“真正的工作”。实际上，所有工作都是真正的，必须要在看板上体现出来。

如果不能意识到这些，我们就不能管理我们所有的工作。当团队对Scrumban的原则有了更深的理解之后，隐藏的WIP就是触手可及“挂在枝头的果实”，团队可以不费吹灰之力找到它们。

**A014-2-3 管理瓶颈和改进流程**

当团队开始使用Scrumban进行流程的可视化管理时，需要重点强调的是，必须更好的理解系统。最开始的目标应该总是确保系统是稳定的，因为试图提升一个不断变化的系统是不可行的。

可视化工作流可以用备用（累积起来需要完成的工作）和饥饿（缺乏工作的下游流程）的形式展现实际开发瓶颈。创建WIP限制可以帮助解决备用的问题（至少可以使备用更少）。不管WIP限制的执行是否到位，饥饿总是很容易看到。

还有一点很重要，就是要在实际环境中明确区分“瓶颈”和“约束”。“瓶颈”是那些需求大于其能满足的资源，一个系统中可以有很多瓶颈。

相对而言，约束是整个系统中具有最小能力的瓶颈。只有在我们决定去识别和管理瓶颈时，瓶颈才会变成“约束”。为了更好的进行区分，我们可以使用Goldratt关于约束理论的五个聚焦步骤：

1. 标识约束：如果不能加强链条中最薄弱的环节，加强其他任何东西都是浪费时间，所以识别出这些薄弱环节非常重要。
2. 利用约束：约束的输出决定了整个系统的输出 。最大化约束的输出通常是提高整体输出的关键，但是当人们在描述约束（这在像软件开发一样的知识工作中很常见）时，必须要考虑到其他方面的因素。
3. 将其他所有的都作为约束的附属：约束的本质是，除了该约束，系统的其他部分可以产出更多。允许其他部分按照自己的生产能力产出的话，会导致大量的备用库存，更长的前置时间，以及频繁催交的任务。另外同样重要的是确保约束不要产生饥饿。
4. 提升约束：只有在已经最大化现有系统的能力之后，才可以思考通过增加资源来进行系统扩展，比如更多的开发人员。在大多数环境下，使用合适的管理方法就有可能获得显著收益。在需要之前过早投入更多的资源只会带来不必要的风险。
5. 防止惯性：一旦改善或提升了约束，这个约束可能不再是系统中最薄弱的环节。持续监控和管理系统是实现持续性能的关键。

按前文所说，在知识工作环境应用约束理论时，在消除所有浪费这个目标背后隐藏了一个缺点。这里有两个原因。

首先，该理论寻求消除许多“非增值”的步骤，其实这些步骤是人而不是机器执行的。根据人类动力学，人们会对自己的工作产生感情。有时候，这种关系的产生是由于某些特定的做法是他们职业身份的核心；在其他时间，社会地位是更棘手的问题。因此，努力消除“浪费”往往遭到情绪上的反对，而这又难以察觉。

这就是在减少和消除延迟的来源时，具备可视化，测量矩阵，反馈机制和套路的Scrumban系统最终能为必要的改变所提供的动力。虽然参与人的职业身份仍然会受到影响，但落实这些变化将成为持久的和可持续的，因为最终参与的人都是自我激励，而不是被外在因素强迫的。

其次，Codratt的约束理论在优化工作流方面很有用，但对于优化工作流程本身就不好说了。假设有一个系统，每4 〜8小时分析师可以把一个用户故事拆分成开发人员可以完成的离散任务。然后进一步假设，这个系统有四名开发人员，每个开发人员都可以在12至16小时内完成所有必要的任务。那么这个系统的约束就是分析师，因为与分析师提供的工作相比，开发人员可以完成更多任务。

现在考虑这样一个场景，即原始约束升高。系统中加入了足够的分析师，能够产生足量的工作来保持开发人员的能力。从工作流的角度看来，这样的系统可能不再有可识别的约束。但是从单独用户故事/工作产品的角度来看，提高分析师的约束对其并没有影响，因为仍然需要在一个用户故事/工作项的两个阶段花费同样的时间： 4至8小时的分析， 12至16小时的开发。

与其说这和减少过程中一个工作项的时间有关，倒不如说， 相关的重点应该放在如何改善工作过程本身。在本节的例子中，过程约束基于整个过程中工作项在每个阶段花费时间的总和。

**A014-2-4 避免不安全的变更模式**

在写关于看板方法可持续发展的议程时，Mike Burrows明确提出了三种关于变更的令人不安的态度，这些态度已经被许多管理者和他们所在的组织证实 ，而大多数Scrum环境中并没有特别提及这些模式：

1. 激进：这会导致过犹不及，体现在即使不理解或没有关注如何变更，就开始倡导变更，并将组织的完整性和人员的福利置于危险之中。这种改变往往“过快”，几乎可以肯定没有考虑到管理变更的能力以及是否能实现持续改进。
2. 自满：在这种情况下，管理层的无作为是组织失败的重要原因。当检查环境来确定需要什么的时候，这种模式通常被认为“过慢”。
3. 篡改：这个是之前提到的一个概念，借鉴了W. Edwards Deming的工作。它涉及到管理者倾向校正正常的变更并干扰现有流程。这种变更的模式“过于随机”。

作为Kanban方法的原则之一，尊重目前的职责和流程，不仅使得Kanban方法成为一种革新的变更方法，而且也是对严肃对待变更的一种认可。出于这个原因，建议在引入Kanban时，用两个简单的角度，来避免不安全的变更模式。

* 认识到组织当前做的事情在服务用户的同时，也使得用户有挫败感
* 认识到组织当前做事情的方式对于组织内部而言，是成功的，也是失败的。

当暴露在Kanban系统中，这些现实和其潜在根源问题就成为反馈循环的主角，接着变更随之而来。每一个变更——不管成功与否——都是有教育意义的，并能提升共同理解。所收获的经验和教训会产生更多机会，为最终成功实现安全、可持续的变更带来更多可能性。

**A014-2-5 创建自律的改进方法**

不管引入哪些框架，在整个组织中产生可持续的精益和敏捷实践通常需要“文化”上的质变。如果在这儿失败，那最终会陷入绩效糟糕的黑洞中，永无止境。

在这个话题的语境里，文化藏在诸如“这就是在这儿，我们如何做事情的”此类的说辞中。一个新的流程或变更计划并不会影响现存文化。文化的改变始于流程改变，而这恰恰是Scrumban能够发挥重要作用的地方。

帮助改变流程的一种方式是成立“改变小组”，其中包括跨组织的不同成员。这种做法把变革的责任加诸于一群人，迫使大家求同存异，为同一个目标走到一起，并对流程提出不同的意见和观点。

保证这个团队在职能和层级方面的多样性很关键，因为职能和层级之间的反差产生的联合决策，通常远远优于一个部门——甚至是项目管理办公室（PMO）——所能产生的。这个团队会一起合作多长时间也很重要。当这些努力变得制度化的时候，就跟真正的敏捷背道而驰（敏捷团队应该通过自主合作，激励自己获取持续提升）。

如果选择这条路，那么对于任何“改变小组”而言，第一目标应该是提高组织改进的方法。团队成员应该负责改进一些方法，用来改变工作方式。在寻求这些改变时，可以用的一些理想化的框架包括现有的回顾会议（这是Scrum团队已经很熟悉的）和一些传播甚广的自律方式，如A3思维等。

回顾会议需要回答几个非常重要的问题：

* 之前我们有哪些方面做的很好？
* 之前我们有哪些方面做的不好？
* 这次有哪些是我们需要提升的？（排好优先级的计划）

自律的方式像A3思维会创建一个合作的环境，让每个人都感兴趣，并参与到输入和流程中。

**A014-2-6 套路和习惯的重要性**

Aristotle(亚里士多德)说：“每天重复做的事情造就了我们，然后你会发现，优秀是一种习惯，而不是一种行为。”习惯的力量是持续成功的关键。

套路（日语词汇，描述运动的具体模式）的概念在第四章中有介绍，在那一章里我们考虑的是如何像Toyota一样用A3思考作为一种自律方式来解决问题。一个套路其实就是Toyota的员工重复练习的东西。套路会在组织内部产生相当于“肌肉记忆”的能力——也就是利用直觉快速的采取行动，无需求助于更慢更符合逻辑的流程。最终，套路代表思想和行为在娴熟的动作中合为一体。

Scrumban要求我们通过协作和科学实验来改进工作方式。改进无非就是一系列的实验。虽然源于制造业，Toyota套路的这个框架作为一个实验性的框架，很容易应用在软件开发中。

* 首先是计划：
  + 了解我们要去的方向（我们的愿景或终极目标）
  + 了解当前状况（我们自己的位置）
  + 设置一个挑战（一个目标——在我们的愿景和当前状况之前，需要解决的某个问题）
* 然后执行：
  + 采取行动
  + 测量结果
  + 评估和学习

在你自己的套路环境中，愿景应该关注于流程，而不是某些可以实现的特定目标。比如，以下任何一个都是对软件开发组织而言有效的愿景：

* 生产环境的0缺陷
* 完成99%的客户需求
* 按需交付最高价值的需求是第一位的

对团队和组织而言，Scrumban的可视化实践可以更容易地为他们的套路确定适当的愿景。同样，Scrumban倾向于对过程（周期时间，WIP，队列长度，节拍时间/迭代长度）和成果（前置时间，吞吐量）进行科学测量，这些测量可以为我们提供正确的数据资料——即那些可以让我们了解当前状态的数据。只用Scrum也可能最终暴露这些问题，但就帮助科学地确定这些问题而言，它能够提供的数据和明确的技术非常有限。

好的套路会要求你为实验设定高于现有知识的目标条件——理想状况下，这个目标应该是可实现的，但是如果不以任何方式提升（希望从行动中学习），这个目标就不容易实现。下面是软件开发环境中，一些有意义的目标条件的例子。

* 把前置时间从45天缩短到30天
* 把平均的WIP从20个用户故事减少到15个
* 计划每三周安排一个部署生产环境的固定日期

从这一点，应该可以理解Scrumban是怎么样给Scrum团队提供一个健壮的框架来进行实验，改变其生产过程，可视化和测量这些改进的影响。这种方式是Scrum不能单独提供的。

最后一个想法，在你的改进套路中，可能还需要内置一个“教练套路”——这是一个能够帮助团队掌握学习过程的框架。“教练套路”的很多方面在类似A3思维等自律过程中得到了证明；其他的方面可能还需要更多的反思。

**A014-3 进而进化对风险的理解和管理**

大多数流程专家和企业管理者有一种倾向，会过分强调扩展系统能力（图9.5 ）这个因素。这可能是因为需求-能力公式中，我们往往会把能力当做我们唯一能控制的。但现实是，风险管理会让我们裁剪那些加诸在系统上的需求。具体来说， Scrumban通过提供数据支持，鼓励与相关干系人的合作和对话，针对公式的需求侧，能够创造选项为我们提供更多控制。

（下面斜体字部分是图9.5里面的文字翻译）

*大多数流程奇客和IT管理者会更重视这边 政策位置，分配，市场细分 ==》理解风险==》裁剪需求*

*能力 看板系统 需求*

*提高能力 减少导致需求失败的根本原因*

*图9.5 Scrumban框架不但可以帮助团队和组织提升自身发布工作的能力， 也可以很好的帮助他们对新增工作需求施加影响*

更强大的风险管理手段始于以下理解，即绝大部分的不确定性源于知识的不完整。这些不确定性可能有很多来源，这些来源跟管理项目（还包括试图理解和管理过程中的偏见）的三个方面有关：

* 项目或者项目组合的活动
* 我们对项目或者项目组合的计划和实际行为的认知
* 基于我们收到的行为数据和信息，我们对项目或项目组合的看法

**A014-3-1 风险应对策略**

既然我们要寻求提高风险的理解和管理，那么认识一些风险管理策略会很有用，我们用这些策略来决定如何管理风险。这些策略通常包括以下内容：

* 假设：假设风险的后果（了解潜在影响，并确保有可用的资源来处理风险）
* 规避措施：更改发生风险的状况，就能够完全避免它
* 缓解风险：主动进行规划来降低风险
* 风险转移：把风险的后果转移到另一方（或者从另一方进行恢复，并将结果转移给我们）

**A014-3-2 成熟实践的例子**

下面给出一些例子来说明通过使用这些方法，成熟的系统可以关注等式的“另一边”来创建平衡系统。

### 裁剪需求

想要裁剪需求，必须开启识别不同工种的风险的能力。因为裁剪需求使得围绕工作交付物创建策略和期望成为可能，其中包括组合一些工作项，评估不能按时交付的影响，以及最应该承担风险的干系人。比如开发业务的不同分类，就是一个巨大的工具。不太成熟的Scrumban的实现可以展示整体性能，但是这可能不足以克服与风险相关的心理障碍。

考虑以下场景：你的团队为所有工种都开发了前置时间度量矩阵。由于这套工具，你目前所处的情况是，需要保证在特定时间内，交付85%的工作项（图9.6）。

图9.6 测量所有工种的前置时可以粗略了解可能的发布时间

现在，如果根据不同的风险类型对工作项进行分类，就可以在这些分类上开发出更细粒度的指标。如图9.7所示，针对不同类型的工作，可能的交付日期实际上是相当不一样的。关于每一类服务的前置时间会反映出不同的分布，基于工作的性质，这就为有争议的工作协商出合理的交付预期提供了“弹药”。

图9.7 随着工作类型的爆发，很容易衡量其不同的前置时间分布，并进行可视化

### 提高主观的风险评估能力

我们必须随时警惕风险管理的合理有效性——这是风险管理中的一个主要陷阱。大多数处理风险管理的工作都是主观的，而且我们往往将其有效性合理化，而不是依靠更为客观的测量，比如在第八章中提到的。那么我们到底应该寻找什么？

首先测试安慰剂效应。如果风险管理缺乏独立、客观的测量，那么成功的看法可能仅仅反映了，因为已经“解决”了一些事情，所以干系人“感觉”对某种情况有更好的了解。假性成功的指标会包括以下这些：

* 声称有效性是由于“结构化”的方法。实际上有很多结构化的方法是不能用的（比如，占星术就是结构化的。）
* 声称有“改变文化”。“文化”的改变可能带来好处，但是不一定与风险管理的改进相关。
* 声称依赖成熟的科学或数学模型。人们是否能够证明这些模型被适当的使用？仅仅存在“证明过”的模型是没有意义的。
* 声称“达成共识”。达成共识并不是目标；目标是更好的风险分析和管理。简单地花精力建立共识，然后走向一条灾难性的道路，只会确保更快的通向错误。
* 声称只要努力比什么都不做强。其实并非如此；不知情或者被误导的努力同样会带来伤害。

其次，通过更好的校准最初代（提高精度的培训和技术）和最终评价，来提升主观风险评估能力。

考虑工作中，我们被要求做的一些主观风险评估（比如一个IT专家说本年度没有黑客行为的概率是90%）。研究发现，就校准技术而言，训练有素的人和未经训练的人在准确度方面有明显的差异。未经训练的人声称他们的评估有90%的置信度时，实际的正确值在52%和78%之间。经过风险评估训练的人，如果声称达到同样的置信度，那么准确率在85%到95%之间。

有一种方法可以用来更好的校准自己的主观风险评估，就是去更深的了解评分方法（如Scrum的点数估计）。 我们倾向于把评分方法看成呈线性比例，而没有考虑到量级差异。考虑一个5分的评分方式。在一星级饭店吃上四顿饭也不会有在四星级饭店吃一顿饭一样的满意度。如果我们在应用标度评分系统时不考虑这种差异，那么主观风险评估的质量仍然很差。

同样重要的是，帮助个人认识到，过去的表现会导致我们在预测风险时，人为地提高自己的信心。有时候，这种信心膨胀是因为我们低估了从过去经验中学到的东西（我们会试图说服自己只能在事后看清楚一些事情（马后炮））。有时候是因为我们把有惊无险和成功当做同等收获。最后，有些时候它反映了我们有一个成功的追踪系统。

例如，比他或她的同龄人多出五年管理经验的一个IT安全经理，尽管其成功地管理一个完全没有病毒和黑客的环境，就一定会有更好的能力吗？出人意料的是，答案是“不”。最近的统计分析表明，除了运气之外，性能的异常值可能不能反映其他任何东西。但是对牵涉其中的人而言，经验更容易让人就未来的评估产生信心上的错觉。

### **A014-3-3 拓宽你的风险管理见识**

在第六章介绍风险管理时，我们发现风险通常驻留在三个基本领域：基于市场的，环境的，和基于工作的领域。第六章还提供了一些跟工作项相关的典型风险特性，来说明划分不同服务等级的方法是怎么样成为特别有效的风险管理方式。

随着对风险概念理解的日趋成熟，团队和组织可以开始进行更广泛和更具复杂度的风险识别和管理。这种努力总是应该由环境驱动。但是，企业从事风险管理的程度与其“成熟度”并没有必然的关联。“成熟”的团队和组织会识别出适合他们的特殊需求，并采取相应的行动。

谨记这一点，让我们来看看，信息技术职能交叉带来的一些额外的风险领域。不过先把风险分成两大类可能有助于思考，这两大类就是内部风险和外部风险。外部风险可以是基于市场的（经济，监管/合规），环境的，甚至是基于工作的（供货商/供应商）。内部风险倾向于放在环境（招聘实践/策略）或者基于工作（工程实践/技术债务）的方面。

#### 你的人员

在一起工作的人，能够有效地帮助组织实现目标。组织管理风险的第一个机会就藏在实际招聘中。

搜索每个候选人都花费不菲；同样，新员工天然的“学习曲线”也是一笔可观的开销。控制差劲的招聘风险的最有效的方法之一就是，学会为你的特定文化招聘。另外寻求当前和预期责任之间的正确的平衡，不仅仅是科学，也有一半是艺术。

考虑Scrum Master这个角色，我之前讨论过传统项目经理和Scrum Master之间的责任差异。在使用Scrumban的成熟系统中，Scrum Master的责任可能与保证新员工可以培训或参与到一些更复杂的预测和建模技术中有关，尤其是当你的组织正朝着这些方面努力的时候。

例如，那些不了解敏捷方法论的团队会很快意识到，现实世界中的“敏捷”完全不像理论中那么简单。但是，招聘一些有类似工作经验的人员可以加速团队的成熟。当新的团队成员无法“得到”敏捷性，成熟的敏捷团队变得沮丧时，这样的招聘也能够解决这种情况。

不幸的是，许多大型企业的IT管理人员都过分依赖人力资源专家，让他们驱动候选人的寻找和筛选过程。我们已经谈到传统的方法在财务和法律职能方面的衰减影响，因为他们不能从系统的角度看问题。那么你又能希望人力资源职能方面有什么不同？

角色的传统定义源于技术，技能和责任。如果你的团队在Scrumban下日趋成熟，对于定义通往成功的技能和特性而言，对影响整个系统成功的关键因素的理解更为重要。

在 Hiring Geeks That Fit 2 一书中，Johanna Rothman描述了具有可观洞察力和前瞻性的招聘过程，说明了一个组织如何在团队成员被录用之前，积极开展风险管理，并在他们开始工作之前，把机会细节化来最小化潜在的问题。最后，要注意系统的性质和质量，要培养团队中的人员关系，这对于招聘和留住相应的人才很重要，可以使他们成为那些最适合为组织成功付出的人。

2. Johanna Rothman, Hiring Geeks That Fit (Leanpub, 2013).

关于招聘过程的最后一个想法是：大多数时候，筛选和面试应聘者使用的是“问题-回答”模式，这种模式没有什么启发性。行为面试技术可以提供机会，使你可以了解在系统的通常情况下，应聘者会如何反应。将“试镜”元素引入招聘过程，同样会很有意义。比如，要求候选人来评审或评论另一个工程师的代码（或者要求候选人带一些自己的代码来让其他工程师评审），这样可以提供巨大的洞察力，来了解这些候选人能不能适应敏捷环境（敏捷环境总会有代码评审）。

#### 策略和实践

一个管理模式，如果传达的是不信任，并且控制不必要的行为，那么不可能实现绩效目标。这就是说，强调以下概念非常重要，即管理框架，监督和控制并非生性邪恶，重要的是如何执行他们。

也许你已经注意到有个常见的模式：传统跨业务的管理方法往往会妨碍组织实现自己的目标。如何解决这些问题可以归并到Jez Humble说的“自适应风险管理”。Humble着重列出了如下几个方面，其中传统方法可能妨碍可持续的高绩效，尤其是在IT方面（表9.1）。

|  |  |
| --- | --- |
| **传统风险管理 “剧院”** | **自适应风险管理** |
| 基于规则和静态的：当碰到新的技术时，需要创建新的规则。 | 基于原则和动态的：无需在第一次制定规则时进行计划，就能够应用到新的场景。 |
| 用控制来防止意外和不好的行为：会无意的引入新的风险，比如标准流程过慢时，会过多依赖紧急变更流程。 | 使用透明性来避免意外和不好的行为：当很容易看到其他人在做什么的时候，人们就会更加小心。 |
| 假设问题都是人为的：所有的操作基于这样一个理论，如果人们遵循系统步骤，那么就不会发生任何错误。这个理论忽略了这样的事实，即流程规范总是需要解释和改编的。 | 假设系统容易出错：系统和环境是持续改变的。人们解决问题，而且必须依赖人们做出主观判断。 |
| 强制代码评审：通常强制使用签入门，也就是需要批准才能合并代码。这会降低效率，并使得重大回归测试的反馈推迟。 | 持续代码评审：在签入之前，同事可以对其他人的代码进行评审，技术领导评审自己团队签入的所有代码，并且一旦代码在主干上，让大家使用代码评审工具对代码添加注释。 |
| 手动测试：通常被作为整合的前置条件。与强制代码评审一样，这个会推迟关于代码修改的系统化努力的反馈。 | 自动化单元测试和接受测试：这些能够很快的通知工程师他们是否在主干上引入了已知的回归。而且应该在提交之前运行在工作站上。 |
| 综合文档：通常用在出现失败时，用文档来发现根本原因是人为错误。 | 发布管道：从签入到发布，提供关于改动的完整可追踪的历史，并且能够自动检测有风险的改动。 |
| 职责细分：实际上成为知识共享，反馈和协作的障碍。在有效响应突发事件时，降低这种情景意识是非常重要的。 | 情景意识：创建直达的工具使得很容易的进行监控，分析和纠正相关数据（就想很多Scrumban的框架！）。 |

表9.1 风险管理比较

自发自觉地让组织朝着自适应风险管理的原则发展，提高组织从类似测试驱动开发（TDD），验收测试驱动开发（ATDD）以及其他实践中持续获取好处的可能性。同样认识到你的组织文化尚未有能力支持和维护这样的做法，不但可以节省你的时间和心灵受伤的程度，也可以避免过早采取潜在积极做法的企图。

**A014-4 向IT之外扩张**

特别针对经理和行政人员

尽量尝试自己去了解可以把Scrumban和其他框架整合起来的各种方法，这么做就像用消防带喝水一样：信息可以非常快的进入，但是信息量过大，根本无法一次就完全理解。随着团队和组织不断提升他们对Scrumban的理解，他们就可以学到怎么样从消防带喝到越来越多的水。本章的“成熟”就是指能够认清这个现实。

为此这部分给出了一些附加的概念，这些概念在团队和组织达到他们旅程的后期时会更加相关。这绝对不是一个详尽的列表；它仅仅代表我在各种情况下已经使用过的一些额外框架。

**A014-4-1 超越预算**

在IT领域之外，很多组织在应用敏捷时都疲于奔命，因为绝大多数敏捷框架没有提供任何关于治理结构的指导意见。这就导致产生错误的假设，即敏捷方法在现有的方式下也是可持续的。但实际情况，许多做法与创建敏捷的价值观和原则有直接的冲突，并会最终扼杀组织的能力。

超越预算框架出现在企业融资领域，表示了一套特定的领导和管理原则，其基于跟Scrumban一样的精益，敏捷和系统化思维。由于Scrumban天然能够整合管理模型，这两种框架相当互补。为了为精益和敏捷原则维护可茁壮成长的环境，超越预算代表了组织指导的一个极好来源。

Scrumban植根于IT领域（知识工作）。虽然它可以应用在许多不同的环境中，它的起源让它在提升理解力和提升管理流程方面特别丰富，比如软件开发流程的机制等。换句话说，它的框架是在整体系统中，面向整合IT/知识工作流程。这个整体系统由商业组织所代表，而我们是其中的一部分。

超越预算框架与其根源（财务和执行管理）有同样的导向性。预算和其他金融机制代表了传统命令——控制结构的组成部分，这种结构妨碍了敏捷的方法论。超越预算框架承认了这个事实，并为管理提供了与Scrumban发布给开发团队同样的理解和实现机制。

有趣的是，只是因为Scrumban引入了精益思想的概念，以更好的管理知识工作的内在变化，所以超越预算框架在关于组织的智力方面也采用了类似的方法。

*环境越复杂，目标越“紧缩”，控制系统就必须有更多的灵活性。“只有多样性可以吸收多样性。”不能提供“必要的多样性”会带来不稳定（繁荣和萧条），并最终导致系统失败。*

Beyond Budgeting Institute发布的一个评估很有启发性，该评估是关于支持自适应的敏捷开发环境为什么需要对比治理模式（见表9.2）

|  |  |  |
| --- | --- | --- |
|  | **传统型** | **自适应性** |
| **组织模型** | \* 组织被视为不同部分的集合，每一个部分都可以被替代（而且这些部门会决定整体的绩效）  \* 组织中有清晰可预测的“因果关系” | \* 组织被视为一个整体，并且认为，贯穿整个系统的不同部分之间的交互决定了系统的绩效。  \* 组织是一个关系网，且不可预测。 |
| **管理模型** | \* 目标是最大化干系人的价值  \* 目标位于“终点”（标靶），而且通向目标的道路是固定的。  \* 领导团队相信，为管理和获取期望的输出，集中式的计划，协作和控制是必需的。  \* 规模驱动成本管理  \* 变化是被动的，或是项目驱动。 | \* 目标是自适应的，并会在很长的一段时间有效。  \* 关注“方法”（流程和人员），而且认为“终点”会自己涌现  \* 领导团队关注在设定方向；组织是自组织和自约束的。  \* 流程驱动成本管理  \* 变化是综合的，连续的和自适应的。 |

表9.2 传统组织和自适应系统运转组织的对比模型综述

当中型和大型系统寻求变得“敏捷”时，很多精力会花在搞清楚怎样将新的实践限制在当前管理策略和结构之内。超越预算框架采用了不同策略。它主张改变传统的管理做法，这样可以让组织意识到新做法的全部好处。特别是它源于金融领域，对高级管理层而言，比起源于IT的其他框架，超越预算框架更具说服力，也更容易引起共鸣。

传统管理中最顽固的神话之一就是：详细的成本预算和严格的监管相结合是实现以下目标的唯一方法：

* 有效的预测和管理支出
* 对付出排优先级
* 分配资源

由于担心消除预算带来的结果，支持更高敏捷性的方法将不可避免的碰到障碍。

与其攻击那些需要详细成本预算和严格控制的策略，不如对传统方法的不足之处进行更有效的讨论，并将辩论引导到更加相关的需求——如何区分成本到底是“好的”还是“坏的”（以及如何对他们进行更有效的管理）。让我们首先考虑不足的方面。传统的成本预算失败的原因主要是因为相关工作有如下特点：

* 多变性
* 不确定性
* 复杂性
* 模糊性

这些特性是不是听起来很熟悉？那是肯定的，因为这跟我们用Scrumban框架，更好的去管理所碰到的挑战是一样的。

现在考虑那些Scrumban依赖的，来帮我们应对这些挑战的机制。我们寻求把工作方式和正在做的东西进行可视化；对工作进行科学的衡量，并动态响应；寻求推迟决策直到最后一刻，并使用良好的流程和信息决策，对我们的决策进行本地化和最优化。

随着Scrumban的到位，组织实际上已经有了一个框架，能够解决传统的成本预算相关的财务和管理方面的缺陷。它只需要使用跟IT开发工作流程一样的方式，来发展传统流程，就可以实现这些好处。

这意味着把中心转移到更相关的方面。不过通过财务机制比如成本预算，来管理组织，其实会对自己的财务状况一叶障目。因为我们会止步于狭窄的重点，也就是流程的成本，而没有去考虑得到的回报。换句话说，我们忽略了去考虑支出是“好的”（也就是说比起花费，企业最终将获得更多的利益）还是“坏的”（也就是说比起花费，企业会获得更少的利益）。

好的成本违背了会计的规则。是的，从会计的角度，合理的解释是它们代表了成本或者花费，但实际上它们是投资。由于传统思想有效的掩盖了这个事实，组织开始致力于将成本控制在尽可能低的范围。但是真正需要解决的是：如何识别和优化“好的”成本。

以这样的认知为背景，让我们考虑一下成本预算的附加弱点——实际上，对于企业的实际需求而言，批准预算的经理了解到的信息比起从事该工作的业务单位要少。那么为什么不给前线作战的人更多控制权呢？

超越预算原则采用了多种方法来满足这些需求。一些更相关的包括以下策略：

* 会计活动
* 趋势报告
* 从绝对转向相对业绩目标设定和评估
* 滚动预测

详细讨论这些活动超出了本文的范围，但是它们值得密切关注，因为这与Scrumban团队——目标设定和评估有着特殊关联。

要使得绩效评估有意义，那么评估必须针对高质量的目标。组织评估绩效的一个常见问题源于人们对数字的盲从，而且没有在产生这些数字的环境中使用任何常识进行判断。简单的说，就是比较两个数字更加容易而已。

由于目标设定和评估在本质上是具有前瞻性的，而且没有目标不希望做到完美。但是，可能发生不可预见的事情；假设也很有可能改变。因此，企业需要建立更有意义的绩效评估，而不是简单的评估某些东西是不是“在预算内”发布，或者关键绩效指标（KPI）是否“健康”。

考虑把成本预算当做一个目标，这个目标是在工作开始之前预先设定的一个绝对数值。那么是不是意味着始终在“预算内”运作的业务单位或者项目就总是“好的”？如果因为业务单位把在预算内运作作为重中之重，而没能追求价值——创造机会呢？如果产品的一个发布或者项目在预算内按期完成，但是质量很差呢？如果一个公司表现良好，只是因为它的主要竞争对手意外倒闭了而达到了目标市场份额，那将会怎么样呢？

个人绩效奖金也是如此。我们应该奖励一些偶然因素，让某些人很容易实现目标而另外一些人不得不非常挣扎吗？或者我们应该承认其他业务目标受到的影响，只是因为目标成为大家唯一的重点？

现在考虑一下通常意义上的绩效。我们已经说过高绩效是关乎持续改进和兑现承诺。能够真正评价我们是否处于“高”的绩效水平的唯一办法是测量我们是否在“平均”水平之上——也就是跟我们的同行进行比较，我们能够（也应该）做的更好。换句话说，高绩效是相对的。

体育联盟代表了相对绩效测量的最终环境。我们很容易意识到，如果一个棒球队的目标是每赛季得分650轮，那么这个目标非常无效。真正重要的是比对手赢得更多的比赛，至于最终获得的总轮数的数量是相对的。很多第一名的球队比他们的对手会有更少的得分总数。如果球队在自己的分组中失败了，很难想象，我们会因为它实现了得分目标，而去奖励这个失败的团队。但是商业领域下，并不能让这种模式等效运行。

正如我们使用功能需求来测试代码的质量，以确定是否可以交付给客户，我们也应该测试我们测量到的绩效，来了解输出到底代表了什么（尤其是相对目标）。执行这类的评估比单纯比较数字困难得多——它需要领导能力和思维的变化。

**A014-4-2 敏捷合同**

作为通往敏捷性的一条替代道路，Scrumban把Scrum现存违背基于合作，透明性和信任原则的偏见进行放大。正如传统的管理方法会影响这些（以及敏捷的成功应用及其可持续性），商业合同也可能成为类似的障碍。对法律事务而言，不管你是购买的IT运营服务，还是使用敏捷方法交付服务，敏捷合同都和“超预算框架”相当，并且与法律事务相关。

跟我们所知的系统思维一样，复杂系统中的行动者，往往会从他们的角色和职责出发，选择那些看起来最好的行为，但是不能清楚的知道自己的行为会对整个系统会产生更大的影响。律师特别容易有这种习惯，因为他们知道，当商业关系出现问题时，每个人都会首先想到去查合同，也就是之前这些律师负责谈判和起草的。带来的结果是，大部分律师倾向于对合同进行“本地优化”。在没有对系统很好了解的时候，软件开发团队会用同样的方式对性能进行局部优化。因此，如果你正为客户开发定制软件或者把开发服务外包给承包商，传统的合同方法会密谋反对你开发过程中的那些敏捷目标。

由于美国的法律是根据以往案例确立的先例，大多数合同结构都能够在之前的技术领域比如建筑领域找到根源。建筑和制造与知识工作有相同的关联，尽管有很多相似之处，但也有相当大的差异，尤其在引入敏捷方法时。考虑以下因素带来的差异：

* 与制造过程类似，在施工开始之前，通常会定义非常详细的工作。执行工作的过程具有很好的可预测性。软件开发更为多变，而且本身不太明确。
* 传统的建筑项目中，具备发布价值之前，通常会有一个长时间的延迟。相反，敏捷开发工作强调频繁增量的发布价值。
* 传统建筑项目中，对执行工作的反馈薄弱，而且通常会发生在过程的后期。敏捷软件开发强调频繁的反馈回路。
* 传统的建设项目中，如果合同提前终止，客户通常会面临巨大的问题。敏捷开发中，因为在合同期间增量交付工作，提前终止合同基本没什么问题。

幸运的是，敏捷合同的结构和条款与传统的合同区别不大。只需要采用全新的视角，专注于不同的东西。因此，尽管传统的合同关注点在问题（出问题时会发生什么），敏捷合同专注于操作流程和交付。

这种变化的视角要求通用合同条款关联到有风险的地方（合同是风险管理的一种形式），并包含一些术语，例如交付范围和可交付物，变更管理规定，处罚和奖励，支付计划，保修条款，限定义务，交付和验收以及合同终止。让我们把工作范围作为一个例子来考虑，为了起草一个传统的合同，双方需要花大量时间和精力确定非常明确的工作范围。与之相反，双方可以起草一个敏捷合同，将重点放在定义一个概要的工作代办列表，用更多的精力去建立过程，通过使用这些过程来分解用户故事/工作项，并将其拉入开发队列。

**A014-4-3 盈利能力的长尾**

Chris Anderson的长尾理论概括了市场利基如何表现潜在的高利润机会（请参考<http://tiny.cc/long-tail> ）。组织可以用它来识别和利用这些机会的一种方法，包括把产品的创新转义到用户社区。

成熟的企业可以将此概念应用到如何对功能列表排列优先级。功能列表通常通过策略视角，市场视角和风险视角产生。为什么不能使用长尾框架作为一个新的视角——一个能够找出赢利点功能的视角，使某些用户可以为这些功能买单。有了这个额外的角度，企业可以选择功能的组合，这个组合是在市场上必须完成的最小功能的集合（David Anderson称之为“筹码”），并且有高利润的附加值。这个“杀手级”组合在非常早期的阶段，最大限度的提高组织的生存能力和盈利能力。（请访问<http://www.codegenesys.com/scrumban/long-tail> 获取更多关于应用“长尾”思考的信息。）

**A014-5 综上所述**

随着使用Scrumban，日益提高理解和能力，团队和组织可以找到无数条可行的道路。虽然本章的大多数技术和实践代表了操作方面更稳健的模式，很多模式依赖于扩展到业务领域的Scrumban框架，远远超过了将其最初引入的软件开发和IT工作流领域。

如果需要强调一个主题，那么就是继续强调提高我们的理解和风险管理。虽然Scrumban共享了Kanban的这个重点，但是Scrumbna特别提到了需要在团队和更大的系统层面增强能力。