Scrum 参考卡

作者：*Michael James*

关于 Scrum

管理框架

Scrum是一个适用于增量式产品开发的管理框架，由一个或多个平均7人左右的团队组成。

它提供了一个包含角色、会议、规则和工件的结构。团队负责在此框架范围内创建和调整他们的流程。

Scrum使用固定时间长度的迭代，称为Sprint，通常是2周到30天之间。Scrum团队试图每一个迭代都构建初一个潜在可交付（充分测试过）的产品增量。

替代瀑布的又一选择

Scrum采取增量迭代方式，抛弃了"瀑布"开发的传统阶段，以获得可以快速地吸纳反馈、优先开发一部分高价值特性的能力。

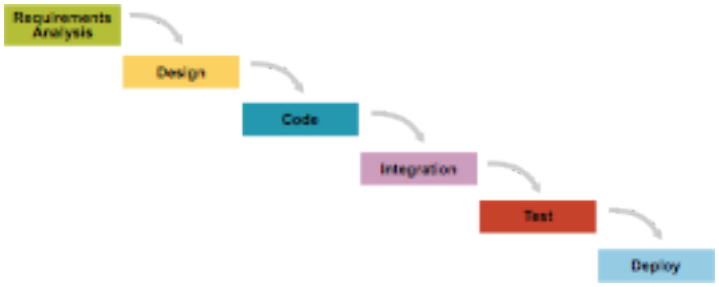

图*1*：传统的"瀑布"开发依赖于在最开始就能够完美地理解需求，并且在执行每个阶段时都只产生最少的错误。

图*2*：*Scrum*在每个迭代中混合了所有的开发活动，并进行调整，以利于定期查明实际情况。

能够发挥Scrum最大功力的，是牵涉到知识创造和协作的复杂型工作，例如新产品开发。Scrum通常都跟面向对象软件开发关联在一起。它也被用在了半导体、抵押贷款和轮椅等产品的开发中。

真的做Scrum，还是在装样子?

Scrum持续地进行实况检查，从而暴露出个体、团队和组织方面受到的不合理约束情况。很多人说是在做Scrum，实际上却对那些需要去破除组织障碍的地方都进行了修改，失去了获得大量好处的机会。

Scrum 角色

产品负责人

- 最大化研发工作投入回报（ROI）的单一责任人
- 负责产品愿景
- 持续地按优先级顺序重排产品列表，对长期性的所有预期进行调整，比如说发布计划
- 需求问题的最终裁决者
- 接受或拒绝单个产品增量
- 决定是否要发布
- 决定是否要继续开发
- 照顾干系人的利益
- 也可能以团队成员身份做出贡献
- 是一个需要起到领导作用的角色

Scrum 开发团队

- 跨职能（例如，包括了具备测试技能的成员，以及业务分析师、领域专家等一些过去通常不被称作开发人员的其他人）
- 自组织/自管理，无需团队之外的专人管理
- 跟产品负责人协商Sprint的承诺问题，每个Sprint一次
- 可以自主决定如何兑现承诺
- 高度协作
- 都待在一间团队室里的效果最好，尤其适用于最开始的几个Sprint
- 队员长期、全职参与的方式效果最好。Scrum把工作带给灵活且快速学习的团队，以避免调动人员或是把人拆散到团队。
- 5~9名成员
- 是一个需要起到领导作用的角色

ScrumMaster

- 引导Scrum的流程
- 协助移除障碍
- 创建一个可促进团队自组织的环境
- 收集经验数据以调整预期
- 隔离团队的外部干扰和分心事，以保持团队心流状态（也即心流区）
- 保障时间盒
- 保持Scrum工件始终可见
- 推动改善工程实践
- 没有任何团队管理职权（任何具有团队职权的人默认都不是团队的ScrumMaster）
- 是一个需要起到领导作用的角色

Scrum 会议

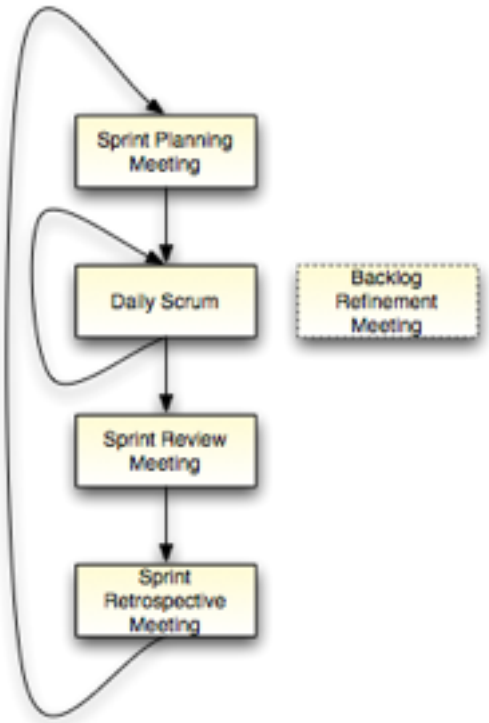

图*3:* *Scrum*流

所有Scrum会议均由在这些会议上不具备决策权的ScrumMaster来负责引导。

Sprint 计划会议

每个Sprint刚开始的时候，产品负责人跟团队一起召开Sprint计划会议，商讨在这个Sprint中他们想把哪些产品列表条目变成为可工作的产品。产品负责人负责讲清楚对于业务来说哪些需求是最重要的。团队则负责选定在不积累技术债务情况下可完成的工作。团队将工作从“产品列表”拉入Sprint列表。

在面对具有内在不确定性的复杂型工作时，团队除了参考前述Sprint的情况，还必须协作以便更直观地了解大家承诺工作项的能力。计算可用小时数、比较估算值跟实际值的方式，会导致团队自以为很精确，还会削弱团队的主人翁意识。除非工作真的可预测，否则他们就该在前几个Sprint就放弃这类实践甚至是彻底不用。

在团队学会如何每个Sprint都能完成一个潜在可交付的产品增量之前，团队应该削减他们所承诺完成功能的数量。若无法改变积习，将会产生技术债务甚至是设计坏死（如图15所示）。

如果产品列表顶部的内容尚未确定，那就需要占用计划会议大部分时间都用来做这件事，就如列表修整会议章节所述。

Sprint计划会议快结束的时候，团队把所选择的条目拆分成Sprint任务，得到一份初始清单，并对工作做出最终承诺。

30天的Sprint，分配给计划会议的最大时长（也即时间盒）是8个小时，对于较短的Sprint，也要相应地缩减会议时间。

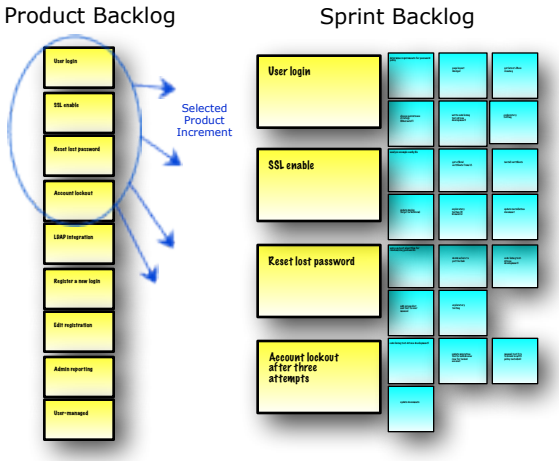

图*4:* *Sprint*计划会议的产出物是已承诺的产品列表条目以及相应的*Sprint*任务。

Scrum日会和Sprint执行

每天在相同时间相同地点，Scrum开发队员们花费总计15分钟相互报告情况。每名队员都要总结他昨天做了什么、今天将要做什么，以及是否遇到了障碍。

站着开Scrum日会，有助于保持会议简短。如果有需要额外关注的话题可以等所有人都报告完之后感兴趣的人再继续讨论。

团队或许会发现，维护使用一份当前Sprint任务列表、一张Sprint燃尽图以及一个障碍列表是很有用的。在Sprint执行过程中，发现要达成Sprint目标必需新增任务，是很常见的情况。那些由团队控制范围之外的问题所造成的障碍，应视作组织级障碍。

产品负责人如果能参加Scrum日会，通常都会有所收获。但如果有任何参会者同时也是团队主管，隐形枪效应就会阻碍自组织和演进式领导力的形成。缺乏团队自组织的实际经验的人往往看不到这个问题的存在，就跟鱼儿看不见水一样。反过来说，产品负责人提高参与度能帮到那些缺乏产品需求相关经验的团队，包括参加Scrum日会在内。

Scrum日会旨在破除独自工作的积习。队员们要警惕那些旧习惯复苏的蛛丝马迹。例如，只面对着ScrumMaster讲话，就是其中的一种症状，表明团队还没有学会如何像自组织实体那样运转。

Sprint 评审会议

Sprint执行之后，团队召开Sprint评审会议，向产品负责人和其他有兴趣了解的人演示可工作的产品增量。

这个会议应该是现场演示，而不是作报告。

演示完之后，产品负责人逐个检查在Sprint计划会议上所做出的承诺，并申明他认为哪些条目是完成的。例如，只有“编码完成”的软件条目是被算作没有完成的，因为没有测过的软件是无法交付的。未完成的条目会被打回产品列表，然后再根据产品负责人的最新优先级顺序判断是否放入后续Sprint。

ScrumMaster协助产品负责人和干系人将他们的意见转化为新增的产品列表条目，以便产品负责人进行优先级排序。新发现的需求范围往往都会超出团队开发的速率。如果产品负责人觉得新范围比最初的预期更加重要，新范围就会取代旧范围在产品列表中的位置。

Sprint评审会议是适合于外部干系人（甚至是最终用户）参加的会议。它是在产品演进过程中进行检验及适应，和人们不断改进对需求理解的机会。对于新产品，尤其是软件产品来说，凭空想象是很难想出来的。大多数客户都需要把软件运行起来、用一用，才能发现他们自己真正想要些什么。迭代开发是一种价值驱动的方式，可用于创建那些无法按照计划驱动方式要求在前期就能讲清楚的产品。

Sprint 回顾会议

一个Sprint在回顾会议之后即告结束。团队将在会上反思自己的流程。他们检验自己的行为、采取措施以适应后续的Sprint。

全职的ScrumMaster会想方设法以避免使用了无新意、让人畏惧的会议形式。深度回顾需要在人们觉得心理安全的环境中才会出现，在大多数组织中都缺乏这样的环境。安全感的缺失会导致回顾讨论变味，要么是一团和气、避而不谈敏感话题，要么就变成了批斗会，人们互相指责、发泄敌意。

绩效评估执行者的出现是影响团队能否敞开心扉的一个常见阻碍。

人们急于得出结论、提出行动建议的偏好，是有效回顾的另一个障碍。《敏捷回顾》[1]是相关书籍中最为流行的一本，它通过一系列步骤来放慢进行此过程：预设会议基调、收集数据、激发灵感、决定做什么、总结收尾。还推荐一本书给ScrumMaster们，《学问》将此过程拆解成几个不同层次的相似步骤：客观性层次、反映性层次、诠释性层次、决定性层次（ORID）[2]。

地理位置分布是做到心理安全的第三个障碍。散布各地的团队在协作上通常都不如共处一室的团队做得好。

回顾通常都会暴露出组织级障碍。在团队解决了自身影响力范围之内的问题后，ScrumMaster应该接着扩大影响，入手解决组织级的问题。

ScrumMaster应该综合运用包括默写、时间线以及满意度直方图在内的多种技术来引导回顾会议。任何情况下目标都一样，要汇聚多种观点以形成共识，并寻求能够带领团队更上一层楼的行动方案。

产品列表修整会议

大多数产品列表条目（PBI）在初期都需要修整，因为它们个头太大而且很难理解。团队们发现，可以在每个Sprint的执行过程中都拿出点时间用于为下一个Sprint计划会议作准备，这种做法很有效。

在产品列表修整会议上，围绕着产品列表上的条目，团队针对他们完成这些条目所需要的投入，并提供相关技术信息以辅助产品负责人对它们进行优先级排序[3]。大且模糊的条目将被拆分并澄清，此过程中需要综合考虑业务和技术两个方面进行处理。有时候也会在大家集体估算之前，叫上团队的少部分人，再加上产品负责人和其他干系人，把产品列表给先理一理、拆一拆。

技艺娴熟的ScrumMaster可以帮助团队学会，如何在严守包含适量测试和重构的“完成”定义的同时，去发现具有业务价值的工作竖切片。

编写产品列表条目时普遍使用用户故事的形式[4]。此方式中，过大的PBI被称作史诗级故事。传统开发方式将特性拆解为横向任务（按瀑布阶段进行组装），它们既无法被独立地进行优先级排序，也缺乏从客户角度可见的业务价值。这种习惯很难打破。

敏捷力要求我们学会把大型的史诗故事拆分为可代表小个头产品特性的用户故事。例如，某医疗记录应用的一个史诗故事“向医生显示患者过敏记录的全部内容”，从中可以提炼出“不管有没有过敏记录都进行显示”。此时，工程师预见的是解析过敏记录内容所存在的技术大难题，然而，到底有还是没有过任何的过敏记录，这才是医生最想知道的重要信息。拆分史诗故事时，通过业务人员和技术人员的通力协作，找出了只需要原始史诗故事20%的投入就能得到80%商业价值的故事。

大多数产品的大多数特性大多数客户都不会用到，因而，明智的做法是拆分史诗故事以便能先交付最有价值的那些故事。即便后续交付低价值特性时可能要返工，那返工总比没工好。

产品列表修整会议没有官方名称，通常被叫做“列表修整”、“列表维护”或是“故事时间”。

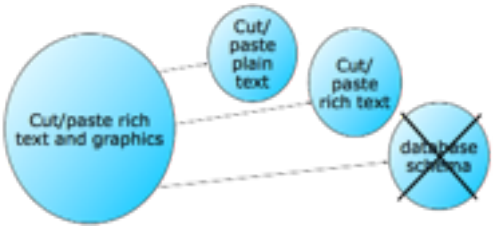

图*5*：在产品列表修整会议上，需要对产品列表顶部附近的大个头***PBI***（通常称作“史诗”）进行拆分，要竖着把它们切成特性薄片（“故事”），而不是横着切成实现阶段。

Scrum 工件

产品列表

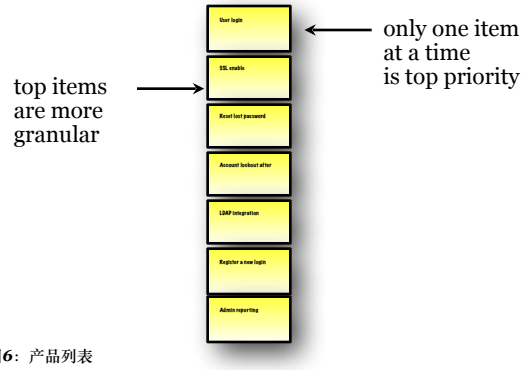

图*6*：产品列表

- 所期望功能的排行榜
- 所有干系人可见
- 任何干系人（包括团队）均可添加条目
- 产品负责人持续调整优先级
- 顶部条目的粒度比底部条目更细
- 通过产品列表修整会议进行维护

[1] 《敏捷回顾——团队从优秀到卓越之道》，原著：Derby/Larson (2006)

[2] 《学问——100种提问力创造200倍企业力》，原著：Brian Stanfield (2000)

[3] 团队应通过协作得出针对某个条目的集体估算值。请参看http://blogs.danube.com/estimation-game

[4] 《用户故事与敏捷方法》，原著：Mike Cohn (2004)

产品列表条目（PBI）

- 主要澄清一个以客户为中心的特性是什么，而非如何做
- 通常都写成用户故事的格式
- 制定产品级公用完成定义以防止技术债务发生
- 可以制定条目专用的接收标准
- 工作规模由团队估算，理想状况是使用相对单位值（例如，故事点）
- 工作规模应该在2~3个人工作2~3天左右，高级团队会更小些

图7：一个产品列表条目代表一个以客户为中心的特性，通常都需要完成一些任务以满足完成定义。

Sprint 列表

- 由Sprint计划会议上团队和产品负责人协商承诺的PBI所组成
- 在Sprint执行期间，所承诺范围是固定不变的
- 初始任务是团队在Sprint计划会议上识别确定的
- 在Sprint执行期间，团队将发现兑现既定范围承诺还需要的附加任务
- 团队可见
- 可供Scrum日会时参考使用

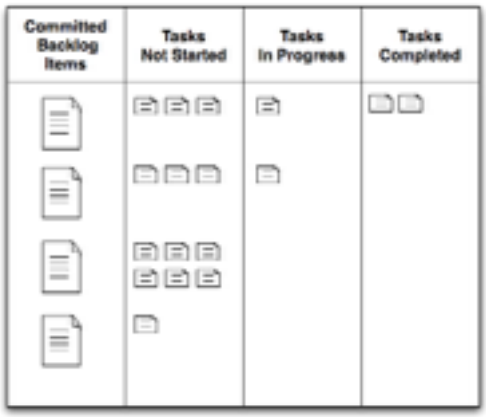

图*8*：*Sprint*列表通常都表现为"信息辐射器"的形式，例如，一块实体任务板。

Sprint 任务

- 澄清达成PBI目标的手段
- 完工需要花费一天或更少的时间
- 每天都要重估剩余工作量，通常以小时计数
- 在Sprint执行期间，每一个人可以主动认领一个任务
- 由整个团队拥有；需要协作

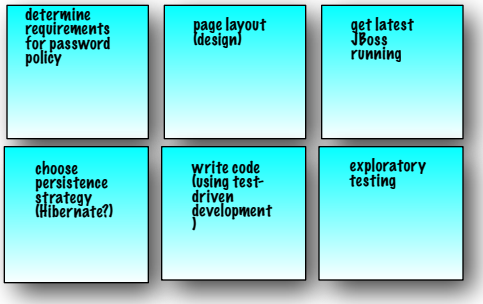

图*9*：完成一个列表条目需要混合开展多种不同活动的相关任务，而不是以往那样分不同阶段进行（例如，需求提炼、分析、设计、实现、部署、测试）。

Sprint 燃尽图

- 显示Sprint期间团队总的任务剩余时间
- 每天都重新估算，因而可能走高也可能走低
- 旨在引导团队自组织
- 按人分条目或增加趋势线等表面功夫会削弱鼓励协作的效果
- 在Scrum发展早期很被看好，但由于在实践中常被误用为一份管理层报告而备受争议。如果它成为了团队自组织的阻碍，ScrumMaster就应该停止使用这个图表。

图*10*：*Sprint*燃尽图

产品/发布燃尽图

- 跟踪记录剩余的产品列表工作规模跟随Sprint的变化
- Y轴可能会使用相对单位值，例如故事点
- 展现了过往的趋势变化以便调整预测

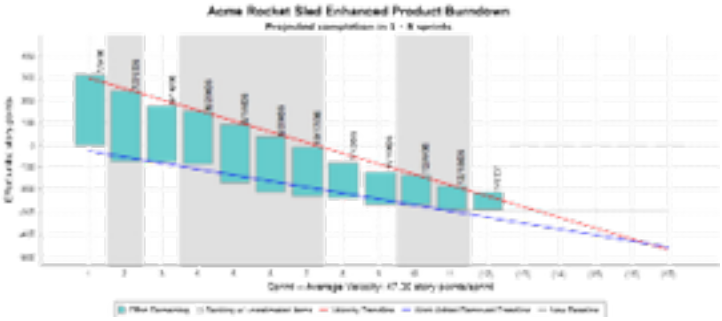

图*11*：一份流行的发布燃尽图变种，修改者是*Mike Cohn*[5]。红线是已完成*PBI*的踪迹（速率），蓝线是新增加*PBI*的踪迹（发现的新范围）。两条线的交叉点昭示了依据经验趋势推测出的发布完工日期。

扩张

坏消息：这事儿有难度。

Scrum将不同职能人员聚集于同一个团队（待在同一个团队房间里更为理想），从而最大化沟通带宽、可见度和信任，以发现不确定的需求及技术上的风险。

在需求不确定性和技术风险很高的情况下，增派人手无济于事，只会让事情变得更糟。根据专业进行分组同样会让事情变得更糟。按架构模块分组（也即，模块团队），最后也只能让事情变得更糟。

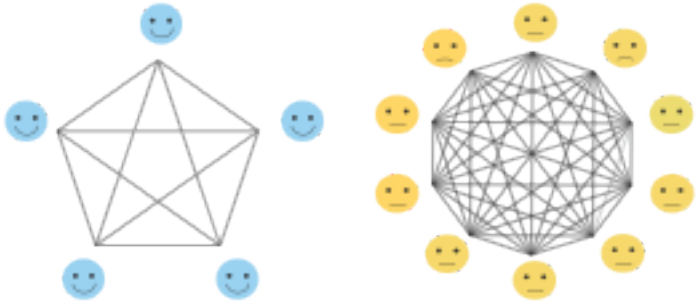

图*12*：沟通路径随团队规模变化而呈指数增长。

好消息：特性团队或许有帮助。

解决此问题最有效办法是创建完全跨职能“特性团队”，他们有能力操作所有架构层从而交付以客户为中心的特性。在大型系统中这意味着得要学习新技能。

当团队专注于学习而不是短期的些微效率时，就能创建学习型组织。

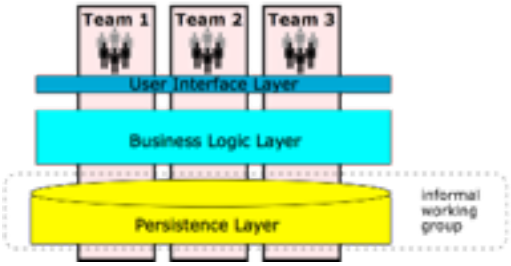

图*13*：特性团队学着跨越架构模块。

更多坏消息：还是有难度。

大型组织获得敏捷力的挑战尤其大。它们多数都陷在了装样子做Scrum的泥潭里[6]。在大型组织里，ScrumMaster们应该定期聚会，通过可视化组织级障碍清单推动转型，阅读《精益和敏捷开发大型应用指南[7]》这样的书。

相关实践

精益

Scrum是在软件开发领域的敏捷运动期间出现的一个通用管理框架，它部分地受到了精益生产方式的影响，比如说丰田生产方式[8]。

极限编程（XP）

ScrumMaster负责提升完成定义的严格程度，但Scrum却并未规定具体使用什么工程实践。名副其实方可谓之完成。自动化回归测试能够防止那些吸血型故事逃出它们的巢穴。设计、架构和基建必须要与时俱进，在持续地重新考虑和修整的过程中演进，而不是在刚开始我们一无所知的时候就试图“最终敲定”。

ScrumMaster可以鼓励团队去学习极限编程（XP）相关工程实践：持续集成（持续的自动化测试）、测试驱动开发（TDD）、坚持无情重构、结对编程、频繁提交，等等。据称运用这些实践可以预防技术债务。

图*14*：绿色直线代表敏捷方法的总体目标：及早且持续地交付有价值的功能特性。合理地运用*Scrum*就得学会如何才能满足严格的完成定义，从而预防技术债务[9]。

[5] 《敏捷估计与规划》，原著：Mike Cohn (2006)

[6] “Seven Obstacles to Enterprise Agility,” Gantthead, James (2010) http://www.gantthead.com/content/articles/255033.cfm

[7] 《精益和敏捷开发大型应用指南》原著：Larman/Vodde (2008)

[8] 敏捷运动的定义请参看敏捷宣言，链接：http://agilemanifesto.org/iso/zhchs/

[9] 图像灵感源自与Ronald E. Jeffries的交谈。

团队自组织

全心投入的团队表现远胜被操纵的团队

在Sprint执行过程中，团队成员们对共同目标有着内在的兴趣，并学着互相管理去达成目标。愿为同侪担责的人类天性与工作者多年来的习惯相互矛盾。接受一个团队靠自身驱动，而不是被外部奖惩所操纵，这跟经理人们多年来的习惯也是冲突的[10]。ScrumMaster所具备的观察能力和说服能力可以克服前期的不舒适感，提高成功概率。

挑战与机遇

自组织团队彻底超越了规模更大的以传统方式管理的团队。在满足一定条件的情况下，家庭规模的群体自然而然地就会变得自组织：

- 成员们以清晰的短期目标为己任
- 成员们能够衡量群体的进展如何
- 成员们可以观察得到彼此的贡献
- 成员们可以放心地给出坦率反馈

心理学家Bruce Tuckman将群体发展分为了“组建期、激荡期、规范期、执行期”四个阶段[11]。达到自组织的最佳状态需要时间。团队在最初几个迭代时的表现可能会不如按传统方式管理的工作组[12]。

多样化团队应对复杂型工作的表现比单一化团队更好。他们经历的冲突也更多[13]。对于全心投入的团队来说，出现意见分歧很正常也很健康；团队的表现就取决于他们是否能妥善处理这些冲突。

坏苹果理论认为，一名消极成员（“拖团队后腿、散播负面情绪或是违反重要的人际交往规范[14]”）可以不成比例地拖累整个群体的表现。这种人并不多见，但如果团队未能及时清除他们，就会放任他们扩大其影响。在挑选新成员时给予团队更大影响力能够部分地缓解此问题。

有些人在老板与工人情况下无法施展拳脚（因为挑战不足或被微管理的原因），将在Scrum团队中焕发光彩。

自组织受很多条件的制约，包括地理位置分布、老板与工人动态关系、兼职团队成员以及与Sprint目标无关的各种干扰。让全职ScrumMaster去努力减轻这类障碍所造成的影响，对于大多数团队来说都很有好处[15]。

何时适用Scrum?

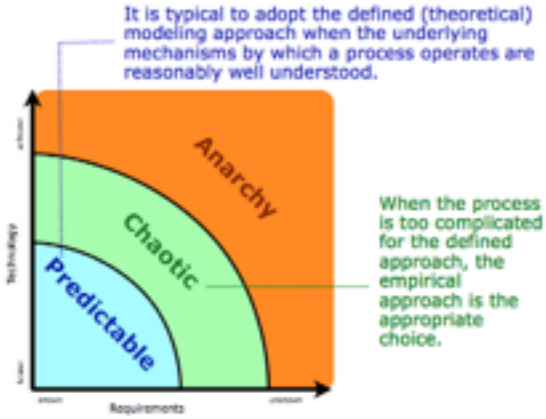

图*15*：Scrum是一个经验式框架，适用于存有需求不明确与或技术不明确情况下的工作。[16][17]

Scrum特别适合那种用预定义流程无法管理的工作，需求不确定、技术实现风险也无法预测。在决定采纳Scrum还是PMBOK®指南所描述那种计划驱动型方法的时候，务必要考虑：基本原理是否已经被充分理解、工作本身是否依赖于知识创造与协作。例如，Scrum就不是为可重复型产品和服务而设计的。

同样要考虑是否有足够的决心去培养一支自组织的团队。

作者简介

Michael James很多年前就已经开始编程。他曾经跟Ken Schwaber一起工作，并成为了一名Scrum培训师。他辅导技术人员、管理人员和高管们对业务进行优化以交付价值。

请将反馈意见发至：mj4scrum@gmail.com 或 http://twitter.com/michaeldotjames

简体中文版译者

有关简体中文版Scrum参考卡的意见和反馈，请发至徐毅：kaverjody@gmail.com 或 http://kaverjody.com

了解更多

- 在线的Scrum培训：http://ScrumTrainingSeries.com
- 最新版Scrum参考卡：http://ScrumReferenceCard.com

[10] 内在动机基于掌控力、自主性和使命感，“奖赏”则与之有害：http://www.ted.com/talks/lang/zh-cn/dan_pink_on_motivation.html（英）

[11] “Developmental Sequence in Small Groups”，Psychological Bulletin 63 (6): 384-99，作者：Tuckman，Schwaber经常引用此文。

[12] 《团队的智慧》，原著：Jon R. Katzenbach，Harper Business (1994)

[13] 《团队的天才：引爆协同创作的力量》，原著：Keith Sawyer，Basic Books (2007)。（此书位列Michael James推荐的ScrumMaster必读书单第2位）

[14] “How, when, and why bad apples spoil the barrel: Negative group members and dysfunctional groups”，Research in Organizational Behavior，Volume 27, 181–230，作者：Felps/Mitchell/Byington (2006)

[15] 全职ScrumMaster职责的检查清单：http://ScrumMasterChecklist.org

[16] 此图引用自Ken Schwaber与Mike Beedle所著《敏捷软件开发——使用Scrum过程》，是经过修改的版本，原图来自Ralph D. Stacey所著《战略管理与组织动力学》。

[17] 《*Process Dynamics, Modeling, and Control*》，原著：Babatunde A. Ogunnaike，Oxford University Press（1992）。