精益ALM听起来像一句空话。对于企业组织而言，ALM的采用并不是很成功。很多实现都缺乏支持和延续性，
导致做出的努力付之东流。精益
是一些伟大想法的集合，实现它需要对组织进行支持和投资。但不要因此就抗拒它，我并不是提议你和一个昂贵的管理咨询团队一起工作，也不是要改变任何事情。相反，我的本意是鼓励你借鉴这些想法。融合这些想法需要组织通过系统的、按步骤的方式实现软件交付，关注简单的变化，而不是采纳复杂的、不切实际的想法。

第一步——识别流程拥有者识别流程拥有者。如果没有流程负责人基本上很难改善任何事情。像需求、开发及测试这些传统的单一学科领域已经在关注自身功能的改善，但它们没有考虑整体。管理或SDLC驱动的方式趋向于关注控制、管理和产品，而不是端到端的价值链。真实的流程是对团队、场景和手头问题唯一的那些流程，而不是停留在抽象层次上的那些。SDLC关注于告诉人们该做什么，而不是为什么这样做，也不会帮助个体提高工作技能。通过联合精益和ALM，你定义一个流程拥有者或者“老师”的角色，关注于改善整个过程，将变化应用到合适的人、流程或技术。为此，流程拥有者应该：

学习一些精益技术 ——你不必成为一个精益专家，或者读上千本与该话题相关的书，但是花费时间学习一些关键的概念和技术是一个不错的起点。James P. Womack所写的《精益思想：在你的公司中消除浪费，创造财富》就是一本非常不错的学习精益的书。该书通过汽车制造的例子介绍了所有的精益概念。

构建团队——变化过程中你并不是独自一人，软件交付是一个包含很多利益相关体的价值链。一个团队应当包含所有学术领域的领导者，如测试、开发和运营。通过包含这些人作为变更代理人，能够衡量变化同时也更有可能实现变化。

理解变化的任务——精益、ALM和流程改进是非常有趣的事情，没有清晰的变化任

务和目标很难获得动力，也难以做出正确的改进决定。例如，变化是为了缩短周期，是为了降低成本，还是为了提高质量，亦或是为了传递更多的业务价值？通过选择这些目标的一个子集，就能够创建并驱动一个计划。

第二步——流程流理解端到端的流程流。要遵循价值而不是数据或者活动，这样流程拥有者才能够对交付流程目标所必须的步骤而不是对今天所完成的活动进行建模。通过审查工作过程的各个阶段，流程拥有者能够识别流程的参与者以及他们所做的工作。这样批量的工作也会变得清晰。例如，所有的需求都从分析阶段转入开发了么，还是只开发了少部分？做流程建模时的一个警示就是，进度比完美的结果更重要。你可以花费数年时间构件一个完美的应用程序生命周期的流程模型，描述所有可能的路径和场景，但是很显然价值来自于对主要步骤的建模。因此，集中精力理解流程的整体运行情况是很重要的，而不是花很长时间构建一个完美的流程图。为了关注流程模型你应该：

定义场景——确定所有的场景，关注主要场景。通过列出所有的场景，你可以先根据那些与分析相关的场景做出决定。例如，bug解决、平台升级或者产品缺陷这样的场景可能没有新项目或者主要增强这些场景重要。

审查工作转化——详细描述工作过程中的各个阶段。任何熟悉流程建模的人都能够舒适地创建工作流程模型、工作过程的各个阶段以及参与人。另一个可能的观点是人工制品；如何处理缺陷和需求以及这些人工制品在其生命周期里都经历了哪些状态？有限状态机这样的技术能够帮助你揭露一些有趣的状态。

活动建模——清楚操作是由谁完成的能够帮你完全理解流程。活动在本质上有正式的和非正式的两种。尽量发掘工作中的社交内容；因为组或者个人之间的交互可能和正式的流程一样重要。社交图形已经成为一种非常流行的社交互动建模的方式，同时它也提供了一个非常有趣的观察点，让你清楚是谁真正地在使用应用程序。通过绘制应用程序变化历史过程中的社交图形，能够识别真正的流程参与者。

查看工具和技术——对于许多组织而言工具定义了流程边界。通过查看工具现在的使用情况以及对它们的连接方式，能够确定流程自动化、数据定义和流程边界。也有可能在同一个流程中使用多种工具，这通常源于对它们的连接点没有一个清晰的理解。通过概括工具的使用情况，能够找出它们的交集。

技能审查——技能审查是流程审查过程中产生的一个有趣的副产物。虽然工具和流程技能并不是必要的技术技能，但却是团队交付可工作软件所需要的。对于很多组织而言，他们将流程看作是自己业务DNA的一部分，但是通常该DNA并没有被使用它的团队完全理解。

第三步——路线图精益提供了一个改善流程流的框架，因此创建一个计划或者路线图描述改善的流程、显著的变化和改善的区域是非常重要的。有时，路线图会标识出各组之间连接中断的区域，这些区域所隐含的和显式的流程模型并不相同。测试组所假设的流程（开发如何管理缺陷以及工程师的错误理解）就是一个很好的例子。解决这些流程问题不一定需要改变流程，但是需要培训、交流和讨论。路线图的本质是为软件交付组织提供一个清晰的、能够改善理解的交流上下文。为了构建一个路线图，应用程序交付专家应该：

将当前的状态映射到变化的指令是第一步——已有的流程模型为现在正在发生的事情提供了一个清晰的定义，而任何将来的状态都应当考虑改变的目标。如果改变的动机是为了缩短周期，那么就需要审查现有的流程找出延缓的原因，确定要改善的区域。

识别快速成功的窍门（wins）——许多人给流程改善贴上了缓慢、无效且难以达到预期目标的标签。但是改变大家对流程改善的看法，同时通过查找快速的、容易成功的窍门在组织内创建一个改变的文化是有可能的。

构建路线图——不要花费几个月的时间构建一个全面的、详细的计划，而是要站在高层次上用心描述需要做出的改变。例如，第一阶段可能是引入流程自动化代替开发和测试之间的电子表格，第二阶段可能是一个显示各方面内容的仪表盘。关键是要为改变提供一些简单的指导方针，从而允许团队对他们的工作做出计划。重要的是所有的计划改变起来都应该足够灵活，因为经验能够帮助变化的领导者理解现实场景。

第四步——增量地引入变化大动作很少会奏效，相反的我们应该增量地改进流程，慢慢地引入改善和变化。如果开发团队使用的是增量或者迭代的工作方式，那么就有可能向他们的工作计划或者待办事务中引入增量的变化。通过将流程改善作为日常工作的一个天然部分，平衡它与其他更传统的项目工作，能够让更多的人参与到变化中来。更重要地是，这样能够从真正做工作的人那里获取直接的反馈。引入改善的构建流程就是一个非常好的例子：如果团队现在正在进行他们自己的构建工作，让那个开发团队做一些额外的工作改善他们现有的构建流程。不要试图让一些重要的团队创建“完美的”构建流程从而跳过这一步，这样的尝试通常都会失败。增量的变化需要：

构建一个变化队列——需要将变化的任务放到一个队列中，然后增量地将这些任务分配到日常的项目迭代工作或冲刺中。有了这个聚合视图，就能够从迭代内部和整体两个方面报告这些工作条目的进展。

引入架构师——并不是所有的工作条目都需要采用新技术或工具，但是技术很可能

会是所有变化的一部分。增量的、分散式的改变能够增加风险，因为单个团队会采纳他们自己选择的但不是适合于组织的技术。为了管理这种情况，分配一个架构师到项目中负责确保工作符合整体的组织目标，同时确保支持所有人而不仅仅是一个团队。

必要的支点——任何事情都不能代替经验。增量采纳的优点是能够提供真实的使用体验和数据，这些内容能够被反馈到整个程序中，从而允许变化领导者改变程序的方向。在面对不可预料的困难、真实世界中的现实问题时很可能需要调整完美的计划。

第五步——量化没有明确的量化指标几乎不可能改善任何事情。通过回顾总体的变化要求，应该能够容易地确定关键指标。流程自始至终都应该应用这些指标，以便确定允许的原因和结果。无论如何，量化通常是最难执行的东西，因为许多组织关注于实际计划、生产故障和预算等内容，将其作为关键指标。虽然这些指标很重要，但是它们关注结果而不是原因。通过变化要求驱动的量化扩大这些指标——例如，如果周期时间是总体目标，那么从开发开始测量部署时间，测试自动化覆盖范围，同时使用时间任务完成部署构建中包含的工作。量化需要应用程序交付专家做到：

确定关键性能指标——通过原因和结果视图评估变化的要求。做一些实验，审查某些操作和相关的措施，洞察它们对目标的交付有什么样的不利影响。

放置记分卡——但是不要认为记分卡上的标准是固定的。这取决于你认为这些措施在改进的过程中是否重要。记住要包含来自于应用程序交付内部和外部的利益相关者，从而确保能清晰地理解他们的需要。有时这需要对变化的要求进行重新评估，因为他们关心的内容和关心的原因会反映到措施中。精益技术，例如Kanban，有助于可视化流，同时通常也是所有计分卡的第一个关键部分。

自动化数据采集——人工数据采集有很多错误，最重要的是它占用时间、耗费金钱。人工数据采集通常会耗费更多的时间和金钱，因此所有使用人工数据的部分都应该从流程中移除。因此，应该尽可能地避免间接措施，而要关注技术和工具的使用。这可能需要工程工作，处理那些改善项目应该包含的以及需要插入到工程流程中的工作。

结论

很明显，21世纪的开发需要更好的管理和更好的灵活性。这两个理念之间的摩擦为你创造了重新评估ALM的使用情况并引入精益思想的机会。ALM鼓励使用量化和连接工具，精益提供了基础的动机、技术和社区从而继续减少浪费并提升价值。这种

结合不仅为你提供了目标，还提供了实现它所需的技术。作为一个产业，是时候接受我们的流程并在落后的项目中