硬件也敏捷！

Yes, Hardware Can Be Agile!

Nancy Van Schooenderwoert

Mar 18, 2015

原文：https://www.infoq.com/articles/hardware-can-be-agile/

作者简介

Nancy Van Schooenderwoert

President and Principal Coach at Lean-Agile Partners, Inc.

Nancy Van Schooenderwoert 是最早将敏捷方法应用于嵌入式系统开发的领军人物之一，她曾担任电子、固件和软件工程师、经理和顾问。她在苛求安全、高度监管的行业中引领了软件开发的敏捷变革活动，并教授现代敏捷方法，如 Mob 编程、敏捷硬件和精益开发方法。她曾在美国、英国和德国指导过敏捷团队。她是全球敏捷相关会议的定期演讲者。她是大波士顿地区首屈一指的敏捷用户组 Agile New England 的创始人和前任总裁。

中文版本翻译：齐　欢（CSP[SM]）

Version 1.0 - me@nap7.com

“没有人能够在硬件领域推动以两周为单位的循环迭代！”当人们谈起敏捷方法在包含了硬件及软件产品开发时，第一反应都是类似的论调。然而，已经有一些团队，尝试将已有的可靠硬件开发理念与少量从敏捷软件中借鉴的新鲜思想结合，进而去应对——甚至已经解决了——进行硬件快速迭代的挑战。

敏捷运动的硬件根源

当今我们称之为**测试驱动开发（TDD）**的方法，其起源正是来自 Ward Cunningham 在嵌入式系统方面的研究工作，以及在此过程中与他的伙伴 Kent Beck 的协作过程。上世纪 70 年代，Ward 在一家研究机构中任职，当时他正致力于找出一条测试新型硬件的切实可行的途径。对此他曾回忆到：“在那个时代，微处理器是件新鲜事物。而我得出了这样一个结论：必须让机器告诉我们，它正在做什么。如果我们打算信任机器传递给我们的信息，那么这部分内容必须是非常简单的…… 自此之后，这个理念在我所做的每一项工作中都留下了它的影子。”同样地，Kent 的经验也折射出类似的本质。

我在某次早年的敏捷会议上与 Ward 结识。当时我十分好奇，是什么吸引了他，让他步入由 Kent Beck 倡导，后来称作“**极限编程（XP）**”的敏捷过程。Ward 告诉我，在上世纪 80 年代，他与 Kent Beck 在 Tektronix 从事 Smalltalk 方面的工作时，发现如果将一些简单的实践经验结合在一起，就会将研发效能提高到令人惊艳的程度。

Ward 介绍到，当新的产品理念推进到了特定节点时，就需要面对公司内部的官僚流程。为了避免陷入到相应流程带来的文书工作之中，他们两位首先确定一个新颖的产品构思，然后推动另一个团队采用它。接下来，他们重复这个过程，做好新产品构思的快速迭代，并展示足够的可行性，从而使得其他人也开始认识到这一产品构思的价值，并促使其顺利通过内部官僚体系的流程。

结对、简洁、测试优先（Test First）等后来演化为 TDD 的实践经验和概念就此诞生并发展。就这样，在硬件领域，基础的敏捷实践方法 TDD 便应运而生。

在敏捷专用集成电路（ASIC）设计中引入敏捷软件中的 TDD 理念

受到在软件开发中运用 TDD 的启发，硬件验证工程师 Neil Johnson 首先将其应用到他对 SoC（片上系统）的验证工作中。

为了便于理解验证工程师（Verification Engineer）的工作职责，我们可以观察下面的工作模型。在 SoC 开发过程中，验证工程师与电路设计师（Circuit Designer）并肩工作，采用与类似裁缝和设计师之间的工作模式。这幅图片展示了在设计并制作一套全新的服装时，所需的工作步骤。在最后一步里，只有裁缝完成了自己的工作，而且没有引入任何缺陷（Defects）的情况下，设计师才能够对设计方案进行问题修正。而如果裁缝使用了错误的织物，或是未能正确测量模特，那么设计师将不得不去应对这些由她的设计方案以外的缺陷所引出的问题。

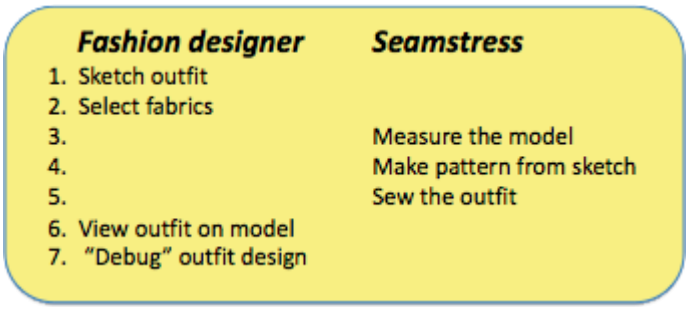

图 1. 验证工程师的工作内容 —— 与时尚设计的工作模型的类比。设计师画出服装的设计图并选择织物。接下来裁缝测量模特身材，在设计图的基础上打样并缝制服装。最后设计师查看模特服装穿着的效果，并对其设计进行"调试"。

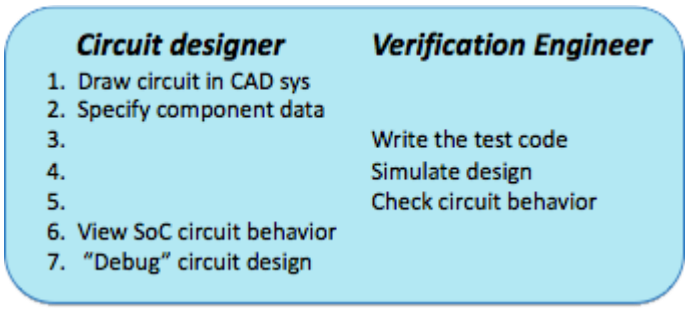

图 2. 验证工程师与电路设计师协同工作时，需要做哪些事情。电路设计师首先使用 CAD 系统绘制电路，接下来明确标注组件数据。验证工程师编写测试代码，进行仿真设计并检查电路表现。最后电路设计师查看 SoC 电路表现，并对电路设计进行"调试"。

同样，电子工程师负责设计电路；而验证工程师负责编写测试代码，并对 SoC 承载的功能进行确认，从而完成设计的验证工作。因此，**如果验证工程师编写的测试代码中出现了 Bug，那么显然会对测试效果的可靠性造成影响。**

如果验证阶段引入了新的 Bug，那么整个工作流程中就不得不增加额外的循环来查错和修复（返工），这将极大地增加硬件开发的时间。因此，Neil 希望弄清楚，如果在“编写测试代码”的阶段使用 TDD，那么与过去相比，他的验证工作能够变得有多“干净”。当 Neil 花费几周时间将 TDD 框架落实到位，而后又经过数周时间对 TDD 工作流进行了熟悉之后，他准备好了对采用 TDD 进行工作前后的情况进行评估和对比。

Neil 对使用 TDD 前后的结果进行了基准测试，他发现当使用 TDD 的 SVUnit 框架时，他的测试代码只出现了 1 个 Bug；而在使用传统 SoC 验证方法的时候，会出现 15 个 Bug。确实，这个结果只是基于一个项目数据样本，但依旧展现了令人惊讶的提升 —— 特别考虑到 Neil 本身就是一位经验丰富的验证工程师，这一研发效能的提升就更为明显。[1]

敏捷 IC 开发——每个 Sprint 阶段都产出可用硬件

德州仪器（Texas Instruments）德国分部的一位数字设计总监，对使用 Scrum 加速定制化 IC（用于闪存和铁电存储器 FRAM[2]）的交付非常感兴趣。在当时，德州仪器的一般交付周期为 12 至 36 个月，而面对这种特殊用途的 IC，其最短交付周期为 18 个月。但一个新客户对交付周期提出了更高的要求。那么，德州仪器将如何使用 Scrum 来加速 IC 开发呢？

在项目启动阶段，他们尝试寻找一位 Scrum 教练，但却根本找不到熟悉 IC 领域的人选。不过，他们依旧确信，至少可以将部分 Scrum 或敏捷实践引入到他们的硬件研发工作中。

最终，这支团队选择采用以下敏捷实践：

- 跨职能团队
- 4 周迭代
- 采用相对估算
- 每周进行 2 至 3 次站立会
- ***每个迭代周期都交付可用的硬件***
- 实施回顾

此时，关键的问题是，当芯片制造流程超过 4 周，而且并不在团队掌控之中的情况下，究竟如何在“每个迭代中”将可用的硬件交付给客户？除此之外，上述的其它敏捷实践则可以非常容易地运用到硬件工作中。

要想以增量方式工作，可以采用的一个方法，是交付经过编程的 FPGA（现场可编程逻辑门阵列）——它们具有与完整 IC 内部的每个 IP 块非常相似的表现（一般来说，每个 IP 块的内部循环时间是 6 至 12 个月）。团队有能力逐个交付已经完成的 FPGA，而且他们主动向客户建议采用这样的方法。不过，虽然这么做能够在一定程度上降低风险，但该外部客户依旧不希望不断被这些中间步骤打扰——他们只想看到完整的 IC。

由于不能与外部客户进行共同迭代，数字设计团队找到了愿意配合的**内部客户**：团队可以向德州仪器内部的软件工程团队交付 FPGA——其软件是在进行硅片制造前的最后一步。

接下来，数字设计团队开始逐月向软件工程团队交付其 FPGA。通过使用可用硬件原型，而不是仅仅停留在设计阶段的方式，团队让 Bug 充分暴露。最终实现了“每个为期 4 周的 Sprint 完成一份可用硬件”的目标，尽管只是通过将 FPGA 硬件作为 IP 块原型的方式。由此，数字设计团队将循环开发时间从每个 IP 块“6 至 12 个月”，降低为“每四周”。

那么，面向外部客户的交付周期时间是否也有得到了优化？不幸的是，由于外部商业原因，这个项目在完成前就被终止了，因此我们无法通过数据来了解外部的交付周期被缩短了多少。

除了有效减少 IP 块的周期时间外，敏捷实践还带来了其它一些意义重大的好处。数字设计团队在工作流程中需要面对一套全新的平台，仅该平台极大增加了不确定性和复杂度。如果他们使用瀑布模型的话，意味着在开始工作前需要先投入为期 12 个月的工作规范化。

领导数字设计团队的 Tobias Leisgang 发现，“规划扑克和相对大小估算，有助于工程师们快速找出沟通问题以及对范围的误解，同时也让整体工作更具有现实感，而不必经历为期数周的详细规划阶段。跨职能团队确保反馈回路得以缩短，**我们不会遇到在制造出一块昂贵的硅片之后，测试工程师才发现模拟电路无法测试的情况。**”[3]

基于原型电路板的敏捷电气（Agile Electronics）

结合一些前期的规划，我们可以把迭代作为主要策略，特别是对于早期产品原型中的电路板和其它物理组件部分。施耐德电气（Schneider Electric）的一支跨职能团队开发了一套照明控制系统，他们在一个 Sprint 中，整合了 3 块电路板和告警灯。

该团队的领导者 Timo Punkka [4]，介绍了他们的射频控制照明调光系统（RF-controlled light dimmer system）是如何在为期 4 周 Sprint 中演进的：

“开发团队由固件、电气、PCB 布局和塑料方面的设计者组成。原型产品包括三块 PCB 板、一套电源、一套控制单元、一套 UI 以及射频转换器。在第一个迭代过程中，只有电源相关的 PCB 板就绪了，而其它两块 PCB 板则通过通用原型板代替。在第二个迭代周期中，所有 PCB 板则均可用，到了 Sprint 结束的时候，用于塑料部分的快速原型也已到位。在 Sprint 回顾中，原型帮助我们体会到，如果我们选择了这一概念设计，那么产品外观的真实感受会是怎样。另外，固件提供了简单的开关功能。要想做到这一切，我们需要**与原型制造商之间建立可靠的合作关系**。在这个案例中，原型制造商同意，当按照**先期约定**的时间收到最终图稿后，就能够在次日交付制造好的硬件原型。

图 3. 为期 4 周的 Sprint 产出的调光系统原型。

这支团队使用负载弹簧连接器和磁铁，将每个模块连接到一起（类似 Mac 的电源连接器）以验证他们的想法。他们还试着将射频部分和天线构建在 PCB 尺寸的 SD 卡上。这两个实验，都涉及了大量的不确定性 —— **最终这些硬件都能够运行，但没有人能够预先确保这一点。**

对团队来说，这些实践的结果推动了**实验**的前进，结合其它一些敏捷实践的支持，使之成为一次具有前瞻性的尝试：

- TDD 确保他们能在早期就尝试测试并持续进行实验，而不会打破早期设计决策。
- **模块化**让他们能够分离出需要进行更多实验的区域。
- **仿真**使得及早进行实验成为可能，从而规避了针对实际目标进行实验带来的高昂成本。

Timo 观察到，**“实验”或者说“逐步达成（Emergence）”，在敏捷思维中处于最核心的位置。它与“第一次就把事情做对”的思路截然相反。**通过在一次 Sprint 中构建一份原型，让硬件、软件和需求能够以并行的步骤逐步实现。

汽车行业的敏捷研发

Wikispeed 项目为我们带来了一部可以合法上路的汽车，它由志愿者打造，且团队每周都会对它进行修订。现在，消费者们就可以购买 Wikispeed 汽车，而且它的百英里油耗仅为一加仑！面对"需要数年之久才能够将一部新车模型投放到市场"的传统理念，Wikispeed 项目向其发起了挑战。Wikispeed 汽车的研发仅仅耗时 3 个月，而且志愿者们持续构建和销售它。该项目的核心策略是**模块化。**得益于稳定的**接口**，他们每周都会修订 8 个模块中的一个或几个并进行**快速集成。**这要归功于他们采用的交错迭代的方法，它支持在每个模块中并行地进行实验和学习。

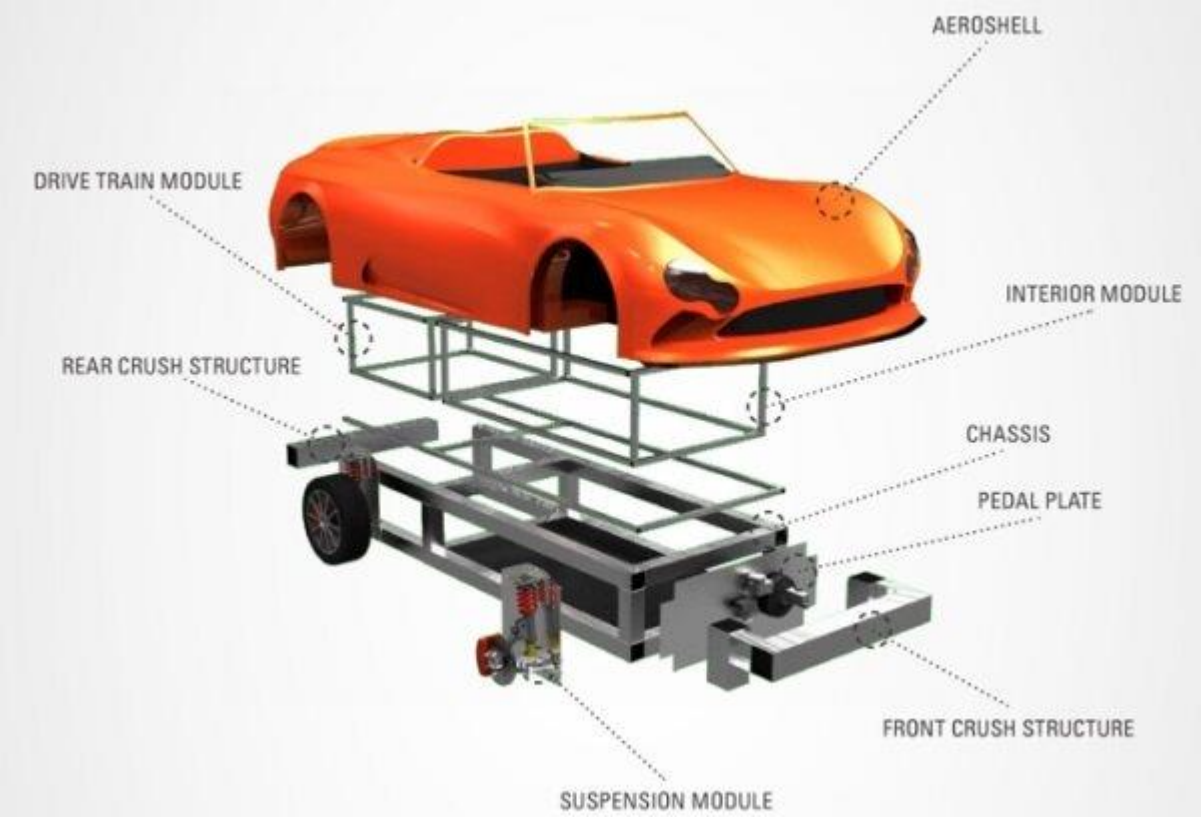

图 4. Wikispeed 汽车分解视图，展现了它的 8 个模块。

另一个关键策略是在材料类型和数量方面的简洁性："XM（**极限制造**，eXtreme Manufacturing）的核心实践是只使用那些能够在不超过一周的时间里进行低成本迭代（重新制作）的材料，并确保在实际中采用的材料和工艺的类型都尽可能少，**以快速积累丰富的经验。**"[5]

Wikispeed 团队的领导者是在西雅图工作的软件咨询顾问 Joe Justice，这是一支由国际化志愿者组成的团队，他们使用敏捷技术来构建汽车。在 2008 年，Justice 注意到 Progressive

Insurance X 大奖的信息——这是一项奖金额度达到一千万美元的评选，旨在探寻是否有可能构建出油耗低至 100MPG（英里 / 加仑）且符合道路安全规范要求的汽车。他参加了这一活动，并在博客上公布了自己的想法——于是一些读者加入了他的团队。

Joe 感到非常吃惊：有如此多的人，愿意努力且免费地工作，只是为了构建一些以开源形式存在的东西，并且能够让所有人受益。目前，已经有 23 个国家中出现了 Wikispeed 商店。其汽车售价大约为两万五千美元；消费者也可以选择花费一万美元，从套件开始自己搭建它。

Joe 从 Wikispeed 团队中获得的第二个关键认知是**人们的动机**。他表示，与其他志愿者相比， Daniel Pink 的思路在目标、掌握力和使命感方面都更胜一筹。他们**积极主动地相互学习，并保持对彼此的关注。**

Wikispeed 汽车由八个模块组成（引擎、框架、碳纤维车身等等）。得益于这些模块之间的**松耦合**关系，人们可以便捷地替换每一个部分。团队每周都会对八个模块中的一个或几个进行修订。并保证随时都有一部能够运转的 Wikispeed 汽车版本，同时还能够快速实现任何升级。那些已经拥有 Wikispeed 汽车的人们，则可以购入新的模块，并自己动手替换原来的对应部分，而无需任何汽车领域的技术背景。

在 Progressive Insurance X 大奖的竞赛中，Wikispeed 在核心组别获得了并列第十名的位置，将超过一百款来自全球各地、得到了企业和大学的充裕资金支持的汽车甩在了身后。[6]

Wikispeed 团队在运用“极限制造（XM）”过程中积累的经验，反映了这样一个现象：Joe Justice 和团队中的许多志愿者都具有敏捷软件开发的背景，他们把这一方向的背景知识引入到了 XM 中。当被问起在制造和工程领域运用敏捷方法的过程中，学到了哪些东西时，Joe 的回答是：

> *“当我开始与 Jeff Sutherland 一起进行联合培养的时候，我建议工程师们使用**面向对象的架构**、**契约优先的设计**以及**测试驱动开发（TDD）**。我所学到的第一个要点，是 Jeff 告诉我，我需要在工程领域打造这些理念——对软件人士而言，这些都是显而易见的知识，但对工程师而言则必须经过“翻译”才能够掌握。”*

对于进行极限制造（XM）：

第一步，也即是第一个 Sprint，将产品或服务划分为 10 个以内的松耦合模块。

第二步，明确面向对象架构、每个模块的输入输出、这些模块在物理上或通过数据如何互相联结、以及冷却等等方面的内容；但要注意，这个阶段应避免涉及任何模块内部的设计或架构。此时应只关注模块之间的**接口**。

第三步，通过以团队方式对每个模块进行版本搭建和重建，进行迭代；同时以增量的方式从实践中学习。一旦他们开始进行概念验证工作，就会在每个模块中增加“illities（对非功能性需求的功能响应）”，就像持久性、可承受性等等。

大部分工程师团队都会遇到这样一个经典问题：他们希望提前做好全部架构工作——所有的接口及“箱子”（接口背后的子系统）中的全部内在。然而，这**将产生巨大的前置时间并造成大量浪费。**

还有部分团队希望甩开架构直接投入到具体工作中。这种方式起步固然迅速，但很快就会放缓节奏。

这两类团队都很少采用最佳的方式：**预先进行架构设计，但只细化到接口层面，接下来停止架构工作，通过实践并构建“箱子”内部来学习。**这似乎是产出可交付产品的最快方式，而且它能够保证持续的速度。

在一款复杂的物理产品中，最好综合运用一些迭代和并行方法。当工作性质中具有“发明部分”的时候，这一点尤为有用。以 Wikispeed 汽车为例，**它的各个模块并行开发，错落添加各项变更，因此这些模块不会同时发生故障。**这确保了团队每周都能够交付可用的软硬件。

总结

我们刚才已经浏览了一些硬件开发过程中使用的经典技术：

- 仿真或原型设计（IC 开发领域与电气领域）
- 模块化（电气领域与 Wikispeed 汽车）
- 组成部分的简洁性（Wikispeed 汽车）

我们还看到了一些直接从敏捷软件实践中借鉴的内容：

- TDD（ASIC 验证）
- 短迭代（IC 开发、电气领域、Wikispeed 汽车）
- 面向对象设计（Wikispeed 汽车）

长期以来，硬件领域一直向着更具灵活性的方向不断前进。部分硬件现已支持非常迅捷的变更，使得不必提前进行完整规划，即可让良好设计逐步浮现成为可能。而对于尚未

达到这样灵活性水平的地方，我们会自然而然地混合运用模拟、模块化和预先规划。提前许久便锁定设计，是极具风险的策略，而综合运用这些技术则能够缓和这些风险。

这里还引出了另一个非常强大的缓和策略——众包（Crowdsourcing，或者称之为：调动团队的集体智慧）能够在一些关键的敏捷实践中发挥作用，例如**回顾、结对编程、每日立会、故事编写**等等。**在任何一支运行良好的敏捷团队中，集体智慧都能够发挥强大的作用。**

没错！要想在采用短迭代方式的同时，还保证每次迭代都能产出可用硬件，是件困难的事情。不过，我们刚才讨论了一些有助于团队落实这一方式的策略：

- 经典的硬件开发技术，例如仿真和模块化，能够极大提高灵活性和开发速度。
- 可以将诸如 TDD 和短迭代等敏捷软件实践，引入硬件开发工作中，从而加速实验和学习的过程。
- 有时候我们需要超越项目要求的范围（例如，在一个月内完成 ASIC）去看问题，并满足客户（或是内部客户）更大的需求。激发团队智慧将帮助我们获得达成目标所需的创造力。

借助我们手中的这些“思维工具”，硬件绝对可以“变得敏捷”。

参考文献

1 Johnson，Neil；测试驱动与硬件验证的新范式（TDD and A New Paradigm for Hardware Verification）；Agile 2014 大会，可前往以下网址阅读：http://www.AgileSOC.com

2 Leisgang，Tobias；如何与足球队一道打篮球（How to Play Basketball With a Soccer Team，论文）；Agile 2012 大会

3 Tobias Leisgang，德州仪器德国慕尼黑分公司系统工程经理；由作者进行采访。

4 Punkka，Timo：嵌入式敏捷（Embedded Agile，论文）；发表于 2010 年嵌入式系统大会

5 http://wikispeed.org/category/wikispeed_shop_manufacturing/

6 Denning，Steve，Wikispeed：如何在三个月内开发一部 100MPG 汽车（Wikispeed: How a 100 mpg Car Was Developed in 3 Months）；福布斯杂志 20125 月，可前往以下网址阅读 http://www.forbes.com/sites/stevedenning/2012/05/10/wikispeed-how-a-100-mpg-car-was-developed-in-3-months/