敏捷建模的特点及误区

姜鲲鹏 郭惠芳 滕志刚 (郑州解放军信息工程大学电子技术学院 450004)

摘要：本文简单介绍了敏捷建模及在利用敏捷建模时应注意的问题，较详细地介绍了敏捷建模的特点及使用时的误区。通过介绍使大家能够更好地使用敏捷建模。

关键词：敏捷建模 迭代 协作

1 前言

2001年2月的美国犹他州，在17个业界的知名人士努力下，敏捷软件开发联盟(Agile Software Development Alliance)成立了，在这之前，联盟的成员们做了很多的工作，知名的XP(极限编程)方法就是众多Agile(敏捷)方法论中的一种，最早，Alistair Cockburn是用"light"一词来区别新型方法和传统方法的不同，而在这一次的会议上，Agile一词最终被确定为新方法学的名称，从此敏捷建模一词也越来越多出现在我们的面前，也越来越为我们大家所熟悉，这里想说明的是敏捷建模边界条件，方便大家对敏捷建模有一个认识。

敏捷开发方法和其他开发方法一样都面临同样的问题，即开发人员声称，并也认为自己遵循了这种方法，而实际上他们并没有完全按照该方法所要求的那样做，只是按照自己所听到的方法中的一部分来做，这就必然会在实际运用中产生了意想不到的问题，这时候他们就会认为这种方法不好，而不是看自己是否真正遵循了这些方法的要求。在极限编程(eXtreme Programming XP)就有这样的例子，一些编程狂宣称他们完全遵循XP所有的原则，但实际上仅仅是遵循了他们听到的关于XP的一小部分内容，如：你不需要这么多文档，他们的产品要么质量低劣，要么根本就不能满足实际用户需求，但他们将这一切的失败都归罪于XP，真是不公平！

同时也应注意到敏捷建模不是万能的，它并不适合于每一个人，每一种情况，即使是你的条件非常适合于敏捷建模，也不能保证它每次都能良好的运行--你在组织内实施敏捷建模时还是有可能犯下错误，大家的经验说明，在以下条件满足时，敏捷建模极有可能发挥其效力，我们也应尽可能发挥它的效能。

因而我们应该注意敏捷建模的条件，了解敏捷建模的特点及使用时容易产生的误区。

2 敏捷建模的特点

(1) 根据Project Stakeholder定下的优先级顺序，首先解决优先级最高的需求，在随后的工作进展中，集中解决风险最大的问题。

敏捷软件开发工作要想成功，需要有project stakeholder的积极支持和参与，Project Stakeholder是那些受软件项目的开发和部署潜在影响的人，包括直接用户，非直接用户，经理，高级经理，操作人员，支持人员，测试者，和这个系统有关(整合或交互)的其他系统的开发人员，以及维护人员。敏捷建模要想成功，你需要了解你的Project Stakeholder是谁，你还要能够和Project Stakeholder保持日常的接触，project Stakeholder能够及时的为你提供信息，做出决策。此外，还要有管理层的大力支持。

(2) 接受需求的变化，并遵照变化了的需求行事--不存在"需求冻结"。

Martin Fowler在他的《新方法学》中指出，如果你的项目就像是自然探险(大多数项目都如此)，那最佳的选择就是采用敏捷方法来开发软件。当需求不明或易变时，你就应该采用一种能够适应这种情况的开发方式。敏捷建模拥抱变化，它采用递增的开发方式，寻求快速的反馈，并且一贯坚持Project Stakeholder的积极参与，这就是敏捷建模对付需求变化的方法。使用敏捷建模，你就能够迅速而有效地发觉客户的需求，并适应客户多变的需求。

(3) 客户，用户都能够积极的参与需求建模，分析建模工作。

敏捷建模需要与客户建立一种新型的关系，特别是当开发是由一家签约公司来进行的时候，因为当雇佣一家签约公司来进行开发时，多数客户愿意订一个固定价格的合同。他们告诉开发方他们所需要的功能，招标，签约，然后剩下的便是开发方去建造系统了。适配性过程和不稳定的需求意味着你不能做这种固定价格的合同，把一个固定价格模式弄到适配性过程将导致一个痛苦的结局，最糟糕的是客户将与软件开发者受到同样的伤害，毕竟客户不会想要一个不适合自己情况的软件，即使他们未付开发方一分钱，他们仍然失去许多。因此，在可预设性过程不能用的情况下，签订固定价格合同对双方来说都有危险，这意味着客户须换一种工作方式，在适配性过程中，客户实际上能够对软件开发过程进行很深入细微的控制，在每一个迭代阶段中，

他们都能检查开发进度，也能变更软件开发方向，这导致了与软件开发者更密切的关系，或曰真正的伙伴关系。但并不是每一个客户，也并不是每一个开发商都准备接受这种程度的介入，不过如要让适配性过程能很好工作，这种合作程度是基本的要求。适配性过程对客户最关键的益处是软件开发中的"回应性"很好。一个可用的，尽管是很小的系统能够尽早投入使用。根据实际使用情况，以及变更了的需求，客户可及时改变一些系统功能。

(4) 采用迭代、递增的方法建模。

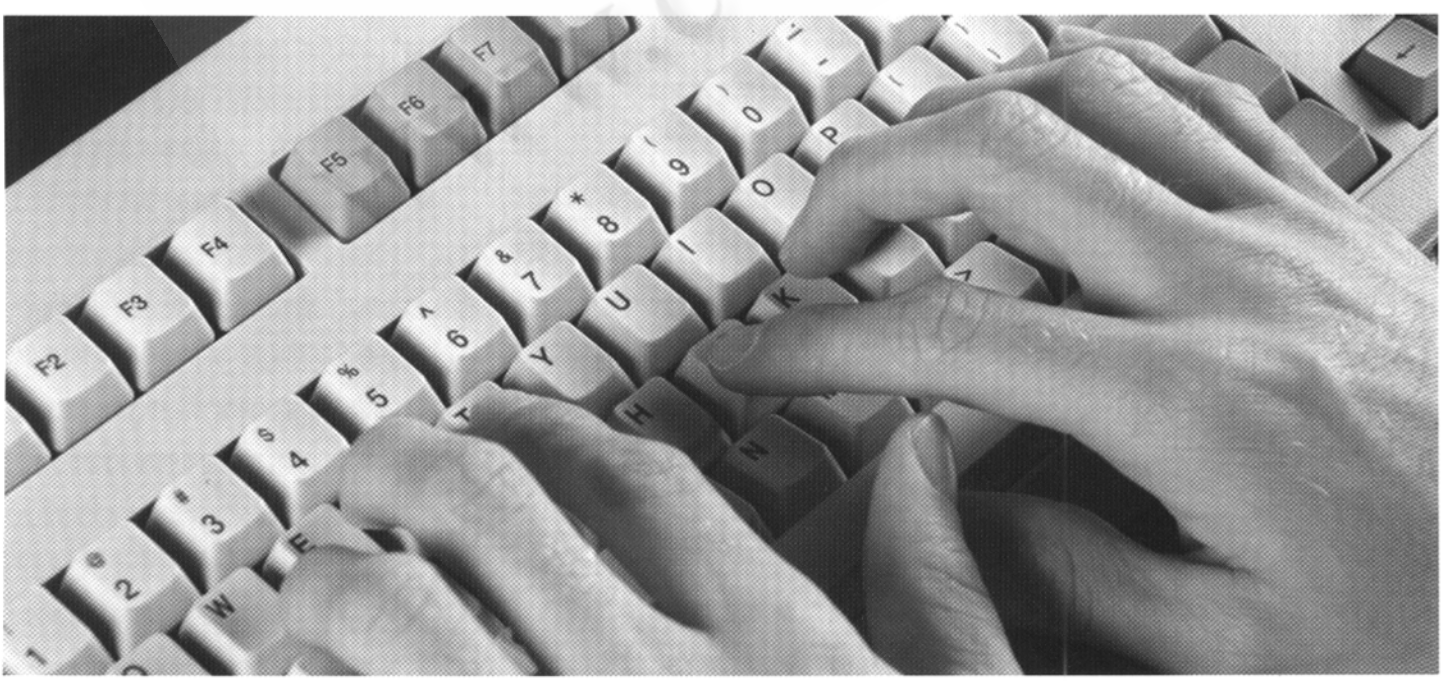
我们所处的世界是不可预测的，人类对世界的正确认识既具有绝对性，同时也具有相对性，那么，我们如何对付一个不可预测的世界呢？最重要，也是最困难的是要随时知道我们在开发中的处境，这就需要一个诚实的、快速的反馈机制来不断准确地告诉我们。这种机制的关键之点是"迭代式"(iterative)开发方法。这并不是一个新思路，迭代式开发方法已存在很久了，只是名称不同，如"递增式"(Incremental)，"渐进式"(Evolutionary)，"阶段式"(Staged)，"螺旋式"(Spiral)等等。迭代式开发的要点是经常性不断地推出阶段性的系统的工作版本，这些版本逐步地实现用户所要求的功能，它们虽然功能不全，但已实现的功能必须忠实于最终系统的要求，它们必须是经过全面整合和测试的产品。这样做的理由是：只有经过整合，测试过的系统才能成为一个项目扎扎实实的成果。文档可以隐藏所有的缺陷，未经测试的程序可能隐藏许多缺陷，但当用户实实在在地坐在系统前来使用它时，所有的问题都会暴露出来。这些问题可能是源码缺陷，也有可能是对需求理解有误。虽然迭代式开发也可用于可预设性环境，但它主要还是用作"适配性"过程，因为适配性过程能及时地对付需求变更，需求变更使得长期计划是不稳定的，一个稳定的计划只能是短期的，这通常是一个"迭代周期"。迭代式开发能让每个迭代周期为下面的开发计划提供一个坚实的基础。迭代式开发的一个重要问题是一个迭代阶段需要多长，不同的人有不同的答案，XP(极端编程)建议一到两周，SCRUM建议一个月，Crystal(水晶系列)更长一些，不过，一般的趋势是让每一个周期尽可能地短。这样你就能得到频繁的反馈，能不断地知道你所处的状况。

(5) 你的精力主要集中于软件的开发，而不是文档或模型本身。

文档的创建和维护都会增大项目的投资。敏捷文档尽可能的简单，尽可能的小，目的只集中在和目前开发的系统有直接关系的事情上，充分了解受众的需要。在敏捷建模中对于文档有一个共同点就是：文档的存在是为了软件的开发，文档的存在只是为了使软件开发有一个目标，在别人想读软件、使用软件时，能够足够快地了解该软件。文档一定要简单到能达到目的就可以。

(6) 注重团队协作精神，欢迎任何人提出意见或建议。

现代的软件开发，几乎不太可能一个人包打天下。软件开发活动是一种群体活动。要使这个群体有效地运转，最重要的莫过于成员之间的相互交流与合作了，敏捷型过程中"以人为本"的理念可以有不同的表现，这会导致不同的效果，而并非所有结果都是完全一致的。实施敏捷型过程的一个关键之处是



让大家接受一个过程而非强加一个过程。通常软件开发的的过程是由管理人员决定的并没有开发人员参与，因此这样的过程经常受到开发人员地抵制，特别是如果管理人员已脱离实际的开发活动很长时间了。而接受一个过程需要大家的自愿，只有大家自愿接受了某个过程，大家才能以积极的态度参与进来。这样导致了一个有趣的结果，即只有开发人员他们自己才能选择并遵循一个适配性过程。这一点在XP中特别明显，因为这需要很强的自律性来运行这个过程。作为一个互补，Crystal（水晶系列）过程则只要求最少的自律。另一点是开发人员必须有权作技术方面的所有决定，XP非常强调这一点。在前期计划中，它就说明只有开发人员才能估算干一项工作所需的时间。对许多管理人员来说，这样形式的领导是一个极大的转变。这种途径要求分担责任，即开发人员和管理人员在一个软件项目的领导方面有同等的地位。注意这里所说的同等，管理人员仍然扮演着他们的角色，但需认识并尊重开发人员的专业知识。之所以强调开发人员的作用，一个重要的原因是IT行业的技术变化速度非常之快。今天的新技术可能几年后就过时了。这种情况完全不同于其他行业。即使管理层里的以前是干技术的人，都要认识到进入管理层意味着他们的技术技能会很快消失。因此必须信任和依靠当前的开发人员。

(7) 尽可能地做到简单——使用你可以获得的最简单的工具；使用能够胜任的最简单的模型。

只要能够达到目的，你应当努力让你的模型尽可能保持简单。模型的详细程度会影响简单性，而所使用的符号范围也会影响简单性。例如，UML的类图就包括了无数的符号，包括对象约束语言，但大多数的图使用符号的一部分就能够完成。所以你常常不需要使用所有的符号，你可以限制自己使用符号的一个子集，当然，这个子集是足够让你完成工作的。因此呢，一个敏捷模型的定义就是一个实现它的目的，没有画蛇添足的模型；为你的预期听众所理解的模型；简单的模型；足够正确、足够一致、足够详细的模型；创建和维护它的投资能够给项目提供正面价值的模型。

(8) 随着开发的进展，你的大部分（不是所有的）模型都会被丢弃（而不作为正式的文档保存下来）。

由于项目的不确定性，世界的不可预测性，也同样由于敏捷建模所采用的迭代式开发的方法，使得开发中产生的大部分中间模型、文档因为有后续版本的模型、文档的产生而失去了存在的意义而被丢弃，只有那些最终的模型、文档才能作为正式的文档保存下来。其中一些中间模型文档因为可重复使用或意义重大的也可能会被保存下来。但也不能因为这些大量的中间模型、文档最终会被丢弃，就不产生它们注意敏捷建模的要求是产生足够正确、足够一致、足够详细的简单模型。

(9) 客户、业务组织拥有业务决定权，开发人员拥有技术决定权。

也正如上面所说的那样，对于敏捷建模一切都是在变化中发展，不仅是开发的代码在发展，模型、文档在发展，而且客户、业务组织的需求也在发展，任何一轮的迭代都是对客户需求的一个逼近。这也就决定了在项目发展中需要几乎所有人的参与，客户、业务组织对开发提出需求，它们决定了项目发展的方向，也最终给出项目是否成功的结论。同时开发人员根据他们的技术提出解决方案及实现该方案。

(10) 模型的内容比内容的格式、表现手法要重要的多。

同样也是由于敏捷建模的要求使得任何模型、文档都有可能被丢弃，这种情况下，花费大量人力、物力在建立和维护这些模型、文档上，都最终会被证明是不明智和多余的，因而对于所用到的模型和文档，首先应该考虑的是它们的内容既它们的存在表明什么，而不是它们的格式、表现手法既它们如何被表示、以何种形式存在。

(11) 在你建模的时候，需要不断考虑的一个关键问题是你该如何测试该模型所描述的内容。

无论哪种敏捷建模方法，都需要经常考虑的一个问题是如何测试某个模型是否正确、是否合乎客户需求，例如：XP中就有首先编写测试代码而后编写程序代码的要求，XP有一个最具冲击力的特点是它对测试的极端重视。诚然，所有的过程都提到测试，但一般都不怎么强调，可是XP将测试作为开发的基础，要求每个程序员写一段源码前都得写相应的测试码，这些测试片段不断地积累并被整合到系统中，这样的过程会产生一个高度可靠的建造平台，为进一步开发提供了良好的基础。

3 敏捷建模的误区

(1) 敏捷建模排斥计划。这是许多第一次接触敏捷建模方法的人容易产生的误解，认为既然敏捷建模是一种"新方法"，因而就应与传统的方法相互排斥、完全排斥传统方法所拥有的一切，并且在敏捷建模的许多文档中都一再地强调变化、强调敏捷建模拥抱变化、适应变化。因之，就认为敏捷建模中完全

不需要计划。这的确是一种误解，在敏捷建模中的每一次迭代过程开始时都应有一个关于本次迭代的计划，计划一下本次迭代所要达到的目标、完成的功能等。并在每一次迭代完成时，应对照开始时的计划，总结失误。只不过与传统方法不同的是这样的计划不需要写成正式的文档、在许多情况下可能只是大家的一个共识。

(2) 你的目的是产生文档，例如需求文档，并且，这些文档需要一个或几个Project Stakeholder签字认可。

这种情况是传统方法中的第一步，需求调查阶段，或整个项目的目的就是为了产生这么一个文档。要知道敏捷建模是为了在客户需求变化或不能完全确定的条件下的为了能够完成项目，尽可能地达到客户理想目标的一种方法。一种情况是客户也无法给你一个他们的需求，一种是客户没有能力完全地描述他们的需求。在这种情况下敏捷建模利用一次次地迭代，加上客户的参与，一次次地逼近客户的理想目标。

(3) 你使用CASE工具进行软件的架构搭建和设计，但却没有进一步在此基础上自动地产生部分或全部的代码。

(4) 你的客户、用户和你一起工作的时间很有限。比如，他们加入项目需求的初始开发阶段，但时间非常有限，只是回答一些问题；然后就会撤出；然后在迟些时候再参加一个或几个对你工作的接受审查。

这种情况下，你不是最好选用传统的被称为"巨型"方法，既对一个软件开发项目在很长的时间跨度内作出详细的计划，对开发过程有着严格而详尽的规定，以期使软件开发更有可预设性并提高效率，然后依计划进行开发。这类方法在计划制定完成后拒绝变化，这种思路是借鉴了其他工程领域的实践。由于你不能和客户有很好的沟通，对于他们的需求在很长时间内是不变的，甚至在你的项目结束前也无法得到他们对你的项目的审查，这也就从根本上否定了敏捷建模的存在。

(5) 你一次只专注于一种模型。最常见的例子就是"用例建模会议"，"类建模会议"或是"数据建模会议"。这种问题产生的最根本的原因就是"单一的artifact开发者"。例如专门进行数据建模的人，专门进行用户界面建模的人。而在AM中，每个人都是多面手，都能够胜任这项工作。

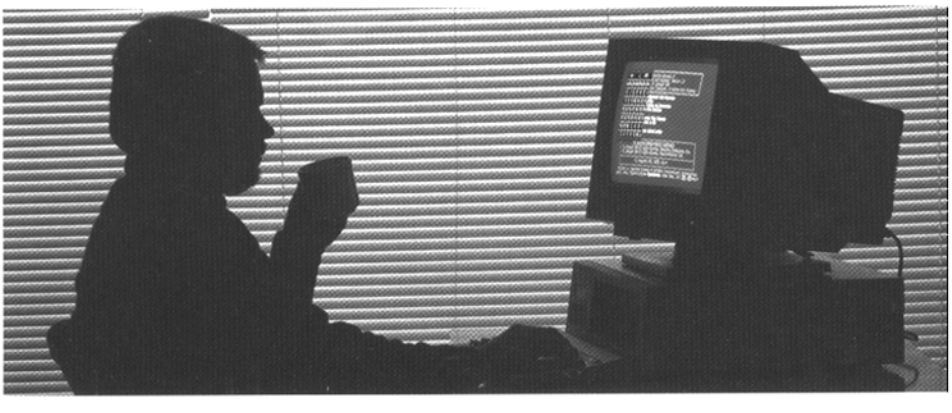
(6) 你工作的方式是做好一个模型以后再做下一个模型－也就是说，你是采取一种"串行"的方法工作。

(7) 你所在小组仅负责模型的设计或文档的编写，当完成这些模型或文档后再将其交付给另外的小组进行下一步的开发工作。也就是说，你是以一种"串行"的方式来"传出"你的工作。

在敏捷建模情况下，需要经常的反馈，不断地修改模型、文档并且敏捷建模的目的是为了更快、更好地开发程序，而不是建立模型和文档。因而，不要期望敏捷建模方法会在建立和维护模型、文档方面给你多大的帮助。

4 结束语

在实际应用中，掌握了以上这些敏捷建模的特点及容易产生的误区，应能很好地区分是不是在利用敏捷建模来处理问题，以及何时应该使用敏捷建模，何时不能使用它。敏捷建模虽然是一种新生的方法，但它的许多内容在以前的方法中已经存在，它的许多内容也同样被许多人自觉、不自觉地使用过。总而言之，敏捷建模是为了解决在立项时需求无法完全固定的情况下，在开发过程中需要全部与项目有关的人不断参与、不断交流，不断完善模型，直到项目最终达到客户需要的一种方法。在这个问题是还有许多需要完善和探讨的地方。这个方法本身也有许多地方等待着大家完善。



参考文献

1 《Continuous Integration》Martin Fowler http://martinfowler.com/articles/continuousIntegration.html，2001。

2 《The New Methodology》 Martin Fowler http://martinfowler.com/articles/newMethodology.html，2001。

3 《Is Design Dead?》Martin Fowler http://martinfowler.com/articles/designDead.html，2001。