敏捷中国大会

ThoughtWorks® InfoQ Enterprise Software Development Community

看敏捷开发的实施

从组织行为学视角

杨 崑 博士

MDS Beijing

第一部分：软件开发方法的管理学解读
第二部分：　研究方法
第三部分：　主要发现
第四部分：　讨论
第五部分：　总结

第一部分：软件开发方法的管理学解读

软件开发是一个复杂的过程。这一过程可以定义为：
为了构想、开发、部署、维护一个软件产品所需的一整套前后一致的政策、组织结构、技术、流程以及中间产品。

---- Fuggetta （ 2000).

第一部分：软件开发方法的管理学解读

现代软件系统存在的四大固有难题（Brooks，1975）：

复杂性
一致性
可变性
不可见性

“Has to be Difficult”

第一部分: 软件开发方法的管理学解读

计算机系统演进的历史可分为三个阶段
(Friedman, 1989)

1. 第一阶段 (1940s-1960s)

2. 第二阶段 (1960s-1980s)

3. 第三阶段 (1980s-1990s)

第一部分: 软件开发方法的管理学解读

第一个阶段 **(1940s-1960s)**： 硬件制约时期

***** 软件开发的“黄金时代” **(Bergland, 1981)**

***** 主要管理策略：

在团队成员之间形成负责的自治权。

第一部分: 软件开发方法的管理学解读

第二阶段 **(1960s-1980s)**： 软件制约时期

软件工程思想 **(Naur and Randell, 1969).**

瀑布模型 **(Royce, 1970).**

无私编程
Egoless Programming **Weinberg,1971**

结构化编程 **Dijkstra, 1970**

结构化设计 **Yourdon and Constaintine, 1979**

结构化分析 **Demarco, 1978**

第一部分: 软件开发方法的管理学解读

第二阶段 (1960s-1980s)： 软件制约时期

主要管理策略：

（1）精心设计开发流程，兼顾负责任的自治性和直接控制。其中更强调直接控制的技术。

（2） 系统开发人员作为用户组织之外的一个独立小组的业务形式开始出现。

第一部分: 软件开发方法的管理学解读

第三阶段： 用户关系制约时期 **(1980s-1990s)**

原型法	**Budde, et., al., 1992**
跌代开发	**Basili and Turner, 1975**
增量开发	**Graham, 1989**
喷泉模型	**Boehm, 1988**

第一部分: 软件开发方法的管理学解读

第三阶段： 用户关系制约时期 (1980s-1990s)

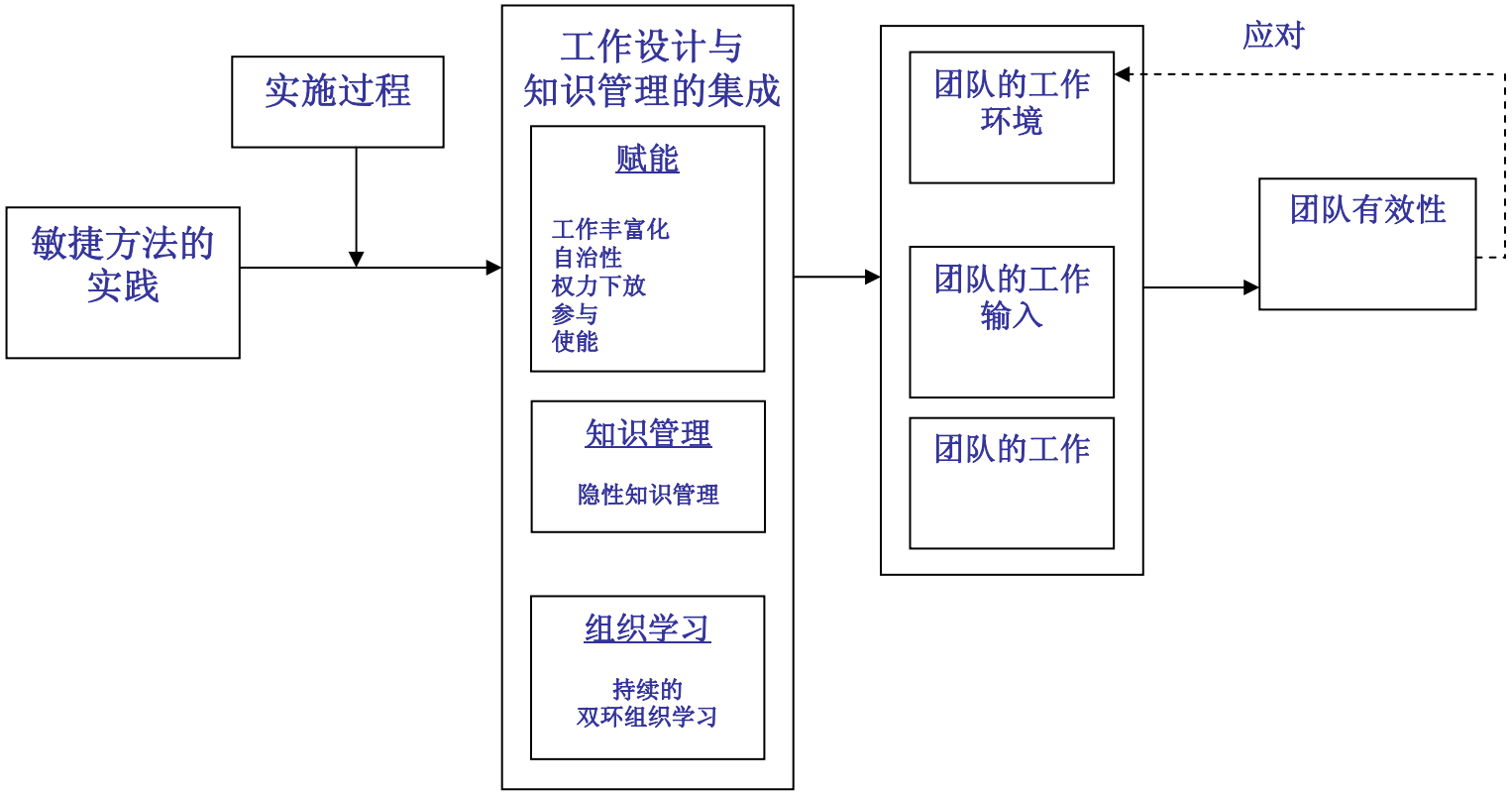
主要管理策略：

重新引入负责任的自治性，

但是，是通过使用总体系统开发员以及更广泛地把用户管理和独立的技术支持综合起来，以正式地的方式引入的。

第一部分: 软件开发方法的管理学解读

XP的管理学分析（Yang, 2008）



敏捷方法对团队有效性的影响

第一部分: 软件开发方法的管理学解读

敏捷开发的实施问题：

Yegge（2006)

“敏捷的倡导者倾向于宣称他们描述的所有好的东东都是真正的敏捷”

“而他们描述的所有不好的东东都是团队实施过程中犯的错误，（与敏捷无关）。”

好敏捷 **vs.** 坏敏捷

第二部分：研究方法(1)

本研究基于两个深度案例分析

案例A:

组织T的一个项目G。软件团队与客户组织在一起开发产品。

整个过程中XP是主要的软件开发方法。

研究人员进入项目组进行为期一周的观察；
举行了3个焦点小组讨论，平均45分钟；
2个访谈（业务分析师，客户CEO），平均60分钟。

第二部分：研究方法(2)

案例B:

组织H中的项目S采用敏捷方法XP进行开发。 三个阶段：
（1）XP为主要方法；
（2）敏捷方法在传统管理模式下使用；
（3）退回到传统开发方法，即“牛仔”式编程

研究人员进入项目组进行为期三个月的观察；
对项目成员进行访谈，平均每人88分钟；
观察并记录14个每日立会，平均每次15分钟；
观察并记录3个CMM的EPG会议，平均每个会议180分钟。

第二部分：研究方法(1)

两个案例实施的XP实践

Practices of XP	Case A	Case B
The Planning Game	√	√
Small Releases	√	√
Metaphor	√	
Simple Design	√	√
Tests	√	√
Re-factoring	√	√
Pair Programming	√	
Collective ownership	√	√
Continuous Integration	√	√
40-hour Weeks	√	
On-site customer	√	
Code Standard	√	√

第三部分:主要发现

- 利益相关者参与
- 开发人员与客户的关系
- 真实需求的挖掘与真实价值的创造
- 协调问题
- **HR**政策
- 组织缝隙
- 组织政治

利益相关者参与 (1)

- 在案例**A**中，整个开发是在客户组织完成的。
- **“专家参与” VS. “客户参与” (Clegg, 2000)**。
- 最终用户的参与对开发过程十分关键，**BA**被认为是最终用户信息的重要来源。

*“用户的参与非常重要。**BA**是其中关键的角色。他从用户那里收集各种信息。我们有客户服务系统，像在线聊天室一样。客服人员将所有的抱怨提供给**BA**。**BA**进行分类形成**stories”.** **(案例 A__客户 CEO)***

利益相关者参与 (2)

- 在案例 B中，利益相关者没有充分参与，这对开发有阻碍作用。

“上一个版本是由这位业务专家开发的。他了解新的系统应当 解决你上一版本的哪些问题，并能提出解决方案。因此，业务专家提的UI已经是包含设计的需求了。但，他的设计只涵盖了这个产品的一部分目标。因此，包括项目经理，BA和我 在内，我们都觉得如果按照他【业务专家】的想法去开发，目前是不会有什么问题的。但将来会有麻烦。因为，这不是实在的客户需求。”

(案例 B__第一阶段__中层经理)

利益相关者的参与是重要的因素

开发人员与客户的关系 (1)

- 案例A, 合作伙伴式

*“我们的结构是平的,很容易沟通. 在传统团队中, 很常见的情况是客户服务人员求着开发人员修改**BUG**. 但敏捷实施以后, 强调以客户为导向. 你会邀请客户来看你的产品. 产品是靠持续地发布,这样开发出来的. 因此客户与开发人员的关系比较平等. ” **(案例 A__客户 CEO)***

开发人员与客户的关系 (2)

- 案例B, 买方与卖方关系, 存在承诺的鸿沟.

“我从没有说过需求是清楚的. 我从没有说过,我们能达成目标. 我只说过我们能在某个时间能接近目标. 我们只有四个程序员,开发主体部分. 我不可能达到最初设定的目标。 … 高层让我按时完成。可你看看我的程序员的水平都是什么样的？用这种水平的程序员要想按时完成，我做不到。”（案例B_第2阶段__项目经理)

建立新型的关系至关重要

真实的需求挖掘与价值创造 （1）

- 案例A，挖掘真实需求是项目团队的主要关注点。

“开发人员3：实施敏捷使我们能够更靠近客户的需求。基于跌代开发，我们努力靠近客户最近的需求。这有两个目的。一个是客户满意度。另一个是控制开发的成本。特别是，在最后阶段进行变化的成本。”(案例A_焦点小组讨论)

- 最终用户的交互行为;
- 背景信息与专家参与;
- 基于业务逻辑的用户故事;

真实的需求挖掘与价值创造 （2）

- 案例B， 挖掘真实客户需求创造真实价值的重要性没有被强调.

> *“当他 [业务专家] 说该这么实现的时候, 项目经理和我都觉得真的客户是否会有这样的想法. 下次,我们使用敏捷方法的时候,一定要找一个真正的客户,或者最终用户.但现在,我们找不到这样的人. (案例 B_第3阶段_中层经理)*

- 寄希望于”正确的人”

真实需求的挖掘和真实价值的创造过程

协调:角色责任

- 案例A, 协调并不是靠详细的角色分配来实现的.
- 案例B, 团队成员倾向于逃避责任.

问: 是集体负责吗?

业务专家: 恩,集体负责. 如果没有问题的话,一切OK. 如果项目团队在压力下,或者 如果每个人都坚持自己的想法,问题就出现了. (案例_第1阶段_业务专家)

- 受制于传统的HR绩效评估系统.

基于团队的绩效评估体系

协调: 权力系统 (1)

- 案例**A**, 团队有较高的自治性, 权力不是集中在高层的.
- 客户与开发团队的协调也不是靠权力来实现的.

*“我们 [客户与开发团队]很少有层级概念.我经常被项目经理批评. 有一次,他们讨论一个**UI** [用户界面] . 我说应当这么做…**blah,blah,blah. UI**开发的接受了并且改变了设计. 项目经理后来把我训了 一顿. 他说: ‘你不应该这么做. 这影响到我这部分了. 做了这些改变我都不知道.” 呵呵, 我在这个团队中没什么权力.” (案例 **A**_ 客户**CEO**)*

协调: 权力系统 (2)

- 案例B,各角色的权力是平等的,没有人拥有专家权力.
- 开发项目的决策形成了死锁的情况.

“我们这儿有一些管理上的问题. 没人能拍板. 项目经理把这些问题提交给高层. 高层说, 所有了解业务的人都在项目里了.如果项目组不能做决策,谁能做? 所以,把问题踢给高层没有意义. 还得团队自己来做决策. 但是,这个项目中团队的决策不是很有效.不应该这样的. ” (案例B__第一阶段1____开发人员6)

- 项目团队缺乏授权.

“由我来做决定? 我做出平衡的前提条件是, 你 [高层] 告诉我什么需求我可以拒绝,什么需求我必须考虑. 我现在没法控制需求. (案例B_第1阶段_项目经理)

授 权

协调: 冲突

- 案例A, 没有明显的冲突被观察到.
- 案例B, 团队结构的扁平化,引起更多冲突. 高层希望避免冲突.

“ 高层不喜欢团队中有冲突. 其实, 他们看见冲突了,但就是不想看见冲突.

(案例 B__第2阶段 ____项目经理)

冲突管理

HR 策略

- 案例A, 更多以人为本的HR实践被采用.

“[在敏捷团队里],人员流失的成本比你压榨一个人得到的收益要高多了. 因此,我们努力提供机会给开发人员,让他们学更多的知识, 经历更多的项目. 如果他对这些感兴趣,他会留在团队里. 维持这个团队,对我来说是很划算的.” (案例A__客户 CEO)

- 案例B,绩效考核系统是基于对个人的评价，奖励体制是集中式的.

配套的人力资源政策

组织缝隙

- 案例A, 组织和部门的缝隙弥合了：

（1）广泛的利益相关者参与；

（2）强调人际交往；

（3）团队自治性；

- 案例B,集体的自治性 促进了角色的综合。但由于拒绝管理层干预，项目团队和管理层之间存在明显缝隙：

（1）战略信息的传达

（2）敏捷方法的理解

关注项目团队与管理层之间的缝隙

组织政治

- 案例B，业务专家是由公司聘请来的。
- 中层经理不愿意公开反对他

“[我不反对业务专家]还有其它原因。这名业务专家是公司请来的。我不能很强烈地说“我们这么做，不那么做”。 我必须考虑他的感受。”

(案例B__第1阶段_____中层经理)

比使用传统方法更关注组织政治

第四部分: 讨论

- 任务特征与组织因素的协同
----互依性&自治性

- 任务特征与组织因素的协同
----互依性&协调模式

- 冲突管理

任务特征与组织因素的协同
----互依性&自治性

- 多元权变理论（**Sambamurthy and Zmud, 1999**）指出：

只有一套不相互制约的权变配置，才能引导项目走向成功。

- 研究表明（**Janz, et. al., 1997; Langfred, 2000, 2005)**任务互依性较高的团队，同时在高集体自治性，低个体自治性的情况下，会取得较好的表现。

案例	任务互依性	个体的自治性	集体的自治性	协同性
案例 A	高	低	高	√
案例 B	中	高	高	

任务特征与组织因素的协同
----互依性&协调模式

- 高互依性需要加强团队成员的交互。
- 有机的 **VS.** 机械的协调模式**(Anders and Zmud, 2002)**

案例	任务互依性	协调模式	协同性
案例 **A**	高	有机式	√
案例 **B**	中	机械式	

冲突管理

- 利益相关者参与以及团队结构扁平化使得冲突增加。
- 发挥冲突的保健作用.
- 软件开发中的五种冲突管理风格**(Gobeli et al. ,1998):**

冲突管理风格	研究结论	案例A	案例B
直接面对 **Confrontation**	☆	√	√
平等交换 **Give and Take**	☆	√	
退缩 **withdraw**			
熨平 **Smoothing**			√
强制 **Forcing**			√

总结

促进敏捷方法实施效果的组织因素:

- 团队成员应当了解敏捷方法背后的原理
- 团队自治性 + 利益相关者参与
- 协同配套的组织管理策略:

互依性 & 自治性 & 协调模式 & 冲突管理风格

- 人力资源政策应当与敏捷方法协同配套.
- 应当谨慎地考虑组织政治因素.
- 应充分提供培训,尤其是软技能方面.

谢 谢

ykwhere@gmail.com