Software Project Management (10)

Du Yugen

ygdu@sei.ecnu.edu.cn

Chapter Managing people and organizing team

软件项目中的人员管理

OBJECTIVES

When you have completed this chapter, you will be able to:

1. **Identify some of factors that influence people's behaviour in a project environment;**
2. **Select the most appropriate people for a project;**
3. **Understand the role of continuing training and learning;**
4. **Increase staff motivation;**
5. **Improve group working;**
6. **Use the most appropriate leadership styles;**
7. **Understand the characteristics of the various team structures that can be employed.**

11.1引言

组织行为研究考虑三个问题：

- 员工挑选
- 员工发展
- 员工激励

11.2理解人类的行为

- 具有实际经验的人是项目中最重要的要素
- 人类行为的解释“如果A情况出现，B可能是结果”
- 主要的问题是真实世界中影响因素过于复杂，难以真正确定A与B的关系
- 对这些问题的思考，人们可以对人类行为的认识达到深化

11.3组织行为学

- **Organizational Behaviour**
- 组织行为学起源于**19**世纪末，**20**世纪初，由泰勒**(Frederick Taylor)**发起
- 泰勒研究人类行为的目的有：
  - 为工作选择最合适的人员
  - 让这些人员采取最佳方法
  - 给最好的工人以更高的工资来刺激
- 泰勒主义（**Taylorism)**经常被用来指代“粗鲁的”和“机械的”，但是有趣的是泰勒主义方法被用在体育训练中，如标枪运动员的训练中。

组织行为学

- 工资不是唯一的因素
- **Donald McGregor提出X理论和Y理论**
  - **X理论：**
    - 一般的人对工作有一种内在的不喜欢
    - 因此需要强迫，指导和控制
    - 人们有逃避责任的倾向
  - **Y理论**
    - 工作就像休息和玩耍一样自然
    - 外部的控制和强迫并非使大家一起为公司目标努力工作的唯一途径
    - 对目标的承诺是与将成绩与报酬相联系的函数
    - 一般的人都能够学会拥有并扩大责任心
    - 想像和其它创造性能力广泛分布

组织行为学

- 判断目前处于理论X和理论Y的方法是观察管理者离开后员工的表现，如果表现没有变化，则处于理论Y环境，如果每个人明显松弛下来，则为理论X环境。
- 期待能够影响行为。如果经理认为你能够干好，那么你可能就会尽力去达到他的期望。

开发人员性格

- **MBTI**测试方法
  - 外向或内向
  - 判断或直觉
  - 理性或感性
  - 推理或主观
  - **4**个方面可有**16**种组合，即**16**种性格类型
- 两个广泛的调查表明，计算机专业人士比一般人更加“内向”，**MBTI**测试中的“内向”只是表示对内心的想法而不是对外部世界的人和事更感兴趣。大约有**50%**－**65%**的计算机人士表现为性格内向，而普通人只有**25%**－**33%**（**Lyons 1985, Thomsett 1990)**
- 这两个调查还发现，**80%**的计算机专业人士更具理性倾向，而普通人只有**50%**。计算机人员更倾向于推理，**66%**的计算机人员属于推理类，而普通人只有**50%**。推理类人喜欢有计划，有条理的生活方式，主观型人更灵活，更容易适应环境。

11.4人员选择

- 软件工具和采用的方法学都会对编程效率产生影响，但是最大的还是人员的影响。
- 1968年进行的调查发现在对同一任务的编写程序时，时间上可能有1：25的差异，而编译则会有1：28的差异。
- 最好的软件人员有何特征？是有经验的编程者还是具有很好数学基础的刚毕业的学生？
- 研究发现最为关键的是经验，而数学基础相对而言影响很小。

人员选择

- 研究发现，从事软件行业的人员与其它行业人员相比，不太愿意“社会交往”。
- **“If asked, most programmers probably say they prefer to work alone where they wouldn’t be disturbed by other people”-Gerald Weinberg**
- 因而选择从事软件开发作为职业并成为专家的人，将来并不一定成为一个好的经理。

招收人员（Recruitment)

- 尽管这是一项非常重要的工作，但是经常项目经理无法作出自己的选择。
- 合格的人员**(eligible)**：简历**（CV,curriculum vitae or resume)**显示该人员在某些岗位上有相当时间的工作经验。
- 合适的人员**(suitable)**：真正能够干好事情的人。
- 需要避免选择了合格但不合适的人员。

招收人员

- 刊登招人广告
- 列出人员要求
- 获取申请者：注意刊登广告的杂志或报纸要与要求人员的素质匹配。
- 审查简历
- 面试
- 其它过程：如要求推荐信或体检等。

招收人员

- 某一公司承担了一个系统改造项目，需要招聘一位有一定工作经验的分析员/编程员，请列出在招工广告中需要写明的该岗位的工作。
  - 对已有系统的新需求的详细分析
  - 对调查的结果的分析和解决方案的评估，包括相关成本的估计
  - 根据组织标准准备系统描述
  - 系统测试
  - 准备功能模块定义
  - 准备和修改模块结构图
  - 软件模块的编码和修改
  - 执行单元测试
  - 用户文档的准备
  - 用户培训

11.5最佳工作方法

- 当一个新成员被聘用后，引入到一个项目组时，要仔细考虑
- 团队领导需要不断地考虑团队成员的培训需求
  - 提供参加培训班的机会
  - 内部培训

11.6激励机制

- 泰勒主义模型
- 在制造行业采用计件工资制，在销售部门采用销售奖金
  - 当新技术引入后，计件工资需要调整，但这是一个敏感的问题。
  - 即使在技术很稳定的场合，也并不见得所有人会最大化地去生产，因为该过程中还受到社会心理的影响
  - 另外，在某些需要协同工作的场合，如软件开发中，很难将他们的工作分开。

激励机制

- 问题：某一软件开发部门想通过重用软件组件来提高效率。他们准备采用奖金来鼓励采用这种方法。你认为如何去实施，实施过程中会有什么问题？
- 问题1：使用组件的编程人员可能减少了代码
- 问题2：他们需要开发软件组件，但是对当前的项目而言并没有提高生产率
- 可以采用功能点法去衡量交付系统的规模，可以采用重用代码比例来度量每个人的工作，重用的代码也可按比例算成编程者的工作量
- 通过记录组件重用的次数并进行奖励来鼓励编程者提供可重用的组件。

激励机制

- **Maslow**的需求层次：不同的人需要不同的激励
- **Abraham Maslow**认为需求是有层次的，低层次的需求满足后，高层次的需求就出现了。最基本的需求是食品和住所，最高层次的需求是自我实现
- 在实际生活中，该原理告诉我们，不同的人在生活的不同阶段需要采用不同的激励方法。例如加薪对新来的，工资较低的员工作用很大，而对工资本来就很高的老员工作用就未必那么明显。老员工可能更希望工作有自主性。
- 问题：报纸报道在许多公司中对核心员工支付了大量薪水，是不是意味着这些人位于需求层次的底层，他们是不是真需要这么多钱才能被激励，这些薪水真正的意义是什么？
- 钱不是用来仅仅满足物质需求的。

激励机制

- **Herzberg**的两因素理论
  - 卫生学或保持因素（**hygiene or maintenance factors)**：如果这些因素出了问题，例如工资或者工作条件将使你不满意
  - 促进因素（**motivators**）：让你觉得工作是值得的。
- 问题：请举例说明你学习或工作中使你感到特别高兴的事，并说明使你特别不满意的事，比较一下，是否能够从中发现一些规律？

激励机制

- 激励的期待理论**(expectancy theory of motivation)（Vroom)**
- 影响士气的三种因素：
  - 期待：相信努力工作会有好的成果
  - 手段：相信好的成果会有好的报酬
  - 感觉值：报酬结果
- 三者都很高的话，将得到高的激励，如果有一者为**0**，则得不到任何激励
- 如果你在为一个企业安装调试一个第三方开发的软件包，如果你意识到因为其中有一个**Bug**所以你无法使其工作，你将放弃
- 如果你在安装调试该软件包，但是你听到客户正在采用其它办法，不再需要你的工作，你也将放弃
- 如果用户真的需要该软件，你得到的只是一些感激，但是如果出来问题，他们将向你提出抱怨，你可能想尽量避开这种事情。

激励机制

- **Oldham-Hackman**工作特性模型
  - 技能多样性：工作中需要多种不同的技能提供了练习的机会
  - 任务的独特性：你工作的内容和结果多大程度上与别人相区别
  - 任务的意义：你的工作对别人的影响，“为飞机拧螺丝感觉上要比为装饰镜拧螺丝更重要也更有意义”
  - 自治性：控制工作方式的能力，“自己做老板的感觉”
  - 反馈：工作结果的反映
- **Couger**和**Zawacki**研究发现编程人员对这些因素的分值较低，而系统分析员较高。
- **Cheney**发现反馈和意识到他们的工作对决策的影响能够对编程者产生很好影响。
- 需要使每个人意识到他的工作对整个产品的进度的贡献。

激励机制

- 激励的方法
  - 目标设定
  - 提供反馈
  - 工作设计
    - 工作范围扩大，例如，编程人员参与维护小组将提高他的责任感，使他在编程时更加仔细。
    - 工作的丰富

不同人员动机比较

	开发人员	项目管理人员	普通人
1	成就感	责任感	成就感
2	发展机遇	成就感	受认可程度
3	工作乐趣	工作乐趣	工作乐趣
4	个人生活	受认可程度	责任感
5	成为技术主管的机	发展机遇	领先
6	领先	与下属关系	工资
7	同事间人际关系	同事间人际关系	发展机遇
8	受认可程度	领先	与下属关系
9	工资	工资	地位
10	责任感	操控能力	操控能力
11	操控能力	公司政策和经营	同事间人际关系
12	工作保障	工作保障	成为技术主管的
13	与下属关系	成为技术主管的机会	公司政策和经营
14	公司政策和经营	地位	工作条件
15	工作条件	个人生活	个人生活
16	地位	工作条件	工作保障

数据来源：《软件工程经济学》（Boehm, 1981）

不同人员动机比较

- 与一般人相比，开发人员更容易受发展机遇、个人生活、成为技术主管的机会以及同事间人际关系等因素的影响；而不容易受地位、受尊敬、责任感、与下属关系及受认可程度的影响
- 与管理员相比，开发人员易受发展机遇，个人生活及成为技术主管的机会等因素影响，而不容易受责任感，受认可程度及与下属关系等因素的影响。
- 如果一个管理者以对自己有效的方式来激励开发人员，则很可能会遭到挫折。

最重要的激励因素（1）

- **“踹一脚并不能产生动力，只能产生被动行为” （Herzberg *1987*）**
- 成就感
  - 自主权
    - 当人们为实现自己设定的目标工作时，会比为别人更加努力地工作。
    - “开发人员提出的进度表总是雄心勃勃的。”（**Cusumano and Selby, 1995**）
  - 设定目标
    - 设定明确的发展速度目标是加速软件开发的简单有效的方法。

最重要的激励因素（2）

- 一个有趣的实验：对于5个小组，安排他们完成同样的任务，该项任务中每个小组都需要完成5个相同的目标，但是对每一组分别要求使不同的目标最优化。结果5个小组中4个最先完成了要求他们实现最优化的目标，另外一个小组第二个完成。每个小组都设定了第2目标，有3个小组第2个实现了第2个目标，1个小组最先完成，1个小组最后完成。没有一个小组对所有目标完成得同样好。试验表明：开发人员会去做安排他们做的工作，他们会为指定的目标工作。
- 如果一个小组一下子有了几个目标，对他们来说每一个目标都做好几乎是不可能的。ITT的一项研究发现，当提出多个目标时，生产率会严重下降。（Vosburgh et al, 1984）
- 为提高项目效率，项目管理人员应该选定一个最为重要的目标。

最重要的激励因素（3）

- 发展机遇
  - 目前从事的工作中用到的知识有一半在3年内必将过时
  - 一个企业可以从如下方面鼓励职工的职业发展：
    - 提供进修机会
    - 给员工提供参加培训或自学的假期
    - 购买专业书籍
    - 派开发人员进扩展其技能的项目工作
    - 为每个新的开发人员指定导师，同时向他们表明企业致力于其职业发展
    - 避免进度压力过大
  - 企业为此的花费
    - 日产公司在田纳西州的Symrna设厂时，其进厂培训的预算为每人$30000(Peters 1987)
    - 各行业排名前10%的企业平均每年为软件开发人员提供2周的培训，为软件经理提供3周的培训（Jones 1994)

最重要的激励因素（4）

- 工作乐趣
  - 三组人都把工作乐趣排在第三位
- 个人生活
  - 个人生活因素对开发人员的影响排第四位，而对经理的影响仅排在第15位，责任感对经理影响占据第1位，而对开发人员仅列第10位
  - 差异的一个结果是，有时管理者会将最具有挑战性的工作分配给最好的人员以示奖励。对管理人员来说，额外的责任是一种乐事，由此带来的个人生活损失则无关紧要。对开发人员而言，这简直是受罚。

最重要的激励因素（5）

- 成为技术主管的机会
  - 开发人员比管理人员更重视技术管理的机会。对于开发人员而言，技术管理的工作代表成功，它意味着这名开发人员已经具备了指导他人的水平。
  - 技术管理并不限于一个项目组的技术负责人：
    - 指派每个人分别作为某个特定领域的技术负责人，如负责用户界面设计、数据库、打印、图形、报表、网络等等。
    - 指派每个人分别作为某个任务的技术负责人，如技术复审、代码重用、集成、工具评估、性能评测、系统测试等等。
    - 除新手外，指定所有的人作为指导者。

其它激励因素（1）

- 奖赏和鼓励
  - 现金方式的奖励必须谨慎处理，开发者一般都善于进行数学计算，他们会估算出和他们的付出相比奖励是否值得。
  - 在《提高软件生产率》一书中，**Barry Boch**提到：糟糕的奖励制度就是给了最佳表现者**6%**的奖励，同时，也给了表现平庸者**5%**的奖励。
  - 赞赏和欣赏的态度，有时比物质刺激更有效。
    - 诚恳而直接地赞扬一项特别的成就
    - 小组的**T**恤衫，运动衫，手表，徽章，标语，奖杯等
    - 幽默或严肃的牌匾、证书、纪念品等

其它激励因素（2）

  - 重大成果的特别庆祝活动。
  - 为该小组颁布特殊政策，如为该小组添置一张乒乓球桌
  - 专门的培训方案
  - 单独开的特别例会
  - 特殊津贴
- 在《他们如此优秀》一书中，**Peters**和**Waterman**指出，一个公司如果想在本行业保持**20**年以上的领先地位，就必须有卓有成效的非货币形式的激励措施。

- 业绩评价
  - **Intel**总裁**Andrew Grove**先生说，业绩评价是“我们作为管理者所能提供的最重要、最贴切的工作反馈”

其它激励因素（3）

- 向导项目
  - **Elton Mayo**和他的助手们曾经做过一个非常著名的关于动机和生产率的试验。他们在**1927**年到**1932**年间，对芝加哥西部电力公司哈森工厂的工人的生产率进行了一系列测试，其目的是了解调整照明度对生产率的影响。首先，他们把灯开亮，生产率随之上升，然后，把灯调暗，生产率随之下降，当把照明保持正常时，生产率又上来了。——“哈森效应”
  - 生产率的一切变化和光照毫无关系，而只是做试验本身这一简单的行为导致了生产率的提高。
  - 在打算采用新方法或新技术之前，必须确认开发小组了解这个项目是一个向导项目。

士气杀手（1）

- 优良的环境，这些环境包括：
  - 合适的光线、供暖和空调
  - 足够大的办公桌和相对封闭的工作间
  - 比较安静，可以集中精力工作
  - 方便地使用办公设备
  - 随手可得的办公用品
  - 好用的计算机
  - 通信交流设施
  - 软件工具
  - 参考手册和出版物
  - 参考书和在线帮助工具
  - 自由的工作时间安排，特殊情况下时间的安排
  - ……

士气杀手（2）

- 管理操控
  - 开发者对如何被管理者操控很敏感。开发者倾向于处理明白无误的事务并希望管理者以直截了当和实事求是的方式来处理
- 执行计划的压力：把开发人员积极性下降为0的最快的方式是给他一个根本不可能的最后期限。
- 缺乏对开发而付出努力的表扬
- 因技术措施不当而受到牵连
- 开发人员没有参与同自己相关的决策行为
- 低质量

11.7团队工作

- 两种团队方式：
  - 功能团队： **command groups**
  - 任务团队**: task groups**

11.8团队的形成

- 并非将一群人简单的组合在一起就构成一个团队，它需要经过下列过程：
  - **forming**：小组成员相互认识并试图建立一些行为规范
  - **Storming:**为了领导权而相互冲突，逐步形成小组的运行方法
  - **Norming**：冲突被解决，团队的意识被建立起来
  - **Performing:**重点被放在任务上
  - **Adjourning:** 小组被解散

团队的形成

- 最佳的团队人员构成（**Belbin**）：
  - **Chair:**不用很富才气，但是必须能够擅长于主持会议，冷静，强势但有忍耐力
  - **Plant:**擅长于提出各种想法和解决办法
  - **Monitor-evaluator:**擅长于对想法和解决办法进行评价
  - **Shaper:** 有点像悲观主义者，帮助小组关注重要的内容
  - **Team worker:** 擅长于创造一个好的工作环境
  - **Resource investigator:** 能够找到各种资源和信息
  - **Complete-finisher:** 擅长于完成任务
  - **Company worker:** 愿意承担不太吸引人的任务
- 某一人可以承担多种类型

团队的性能

- 是否团队一定比单个人的工作更有效？
- **“Some work yields better results if carried out as a team while some things are slowed down if the work is compartmentalized on an individual basis”-one manager of IBM**
- 任务分类：
  - **Additive tasks:**各个参与者一起共同完成任务，参与者可以互换
  - **Compensatory tasks:**相互补充
  - **Disjunctive tasks:**只有一个正确结果，因而群组的效益与最好的那个成员取得的效果一样
  - **Conjunctive tasks:**进度取决于最慢的人员

团队的性能

- 社会性惰化（**Social loafing**）现象:小组成员并没有作出他们应有的贡献。
- 问题：如何鼓励在团队中所有的成员能够作出他们的贡献？
- 答案:
  - 使每个人的工作都是独特的
  - 使每个人对团队工作的结果感兴趣
  - 根据每个人对小组的贡献进行奖励

构造高业绩团队（1）

- **1989**年，**Larson**和**LaFasto**发表的一项研究发现，高效团队所具有的特性有惊人的一致性。不管是麦当劳炸鸡**McNugget**团队、挑战号太空飞行器研究小组、心脏病外科小组还是登山队。
- 共同的、可提升的远景或目标
- 团队的成员认同感：**IBM**公司的“黑色团队”，采用自己团队的服饰代码。即使原来的团队成员已经许多离开了，但是团队还存在。

构造高业绩团队（2）

- 结果驱动的结构
  - 角色必须明确，每个人必须在任何时候都对自己的工作负责。
  - 团队必须有有效的沟通系统以支持信息在团队成员间的自由流动。
  - 团队必须以某种方式监控个人表现并提供反馈。
  - 任何时候的决策制定都要以事实为根据，而不是以个人主观的意见为依据。

构造高业绩团队（3）

- 胜任的团队成员：最佳的团队结构和团队成员
- 团队的承诺
- 相互信任：诚实、开放、一致和尊敬
- 团队成员间相互依赖
- 有效的沟通
- 自主意识：可以自由去做任何使项目成功所必须要做的工作
- 授权意识：被授权可以采取任何为获得成功所需要的行动
- 小的团队规模：8¯10人比较合适
- 高层次的乐趣：并不是所有愉快的团队都是高产的，但是高产的团队绝大多数是愉快的。

长期的团队建设

- 长期保留团队的原因：
- 更高的生产率
- 低启动费用
- 较低的个人问题风险
- 减少人事变动：
  - 现在的计算机工作人员每年的人事变动率大约为35%，公司20%的平均人力总花费是人员变动成本。
- 时间空闲问题：虽然团队可能空闲因而消耗了资金，但是组建新团队也需要花费。

团队结构

- 组建团队应考虑的第一因素：团队目标，从而也决定了三种类型的团队（**Larson, LaFasto, 1989**）
  - 解决问题：一组为疾病控制中心工作的流行病专家，在努力诊断霍乱爆发的原因
  - 创新：一组麦当劳食品专家尝试发明一种新的麦当劳事物
  - 战术执行：一组突击队员执行一次袭击任务，一个外科医疗团队和一个棒球队

团队类别（1）

- 业务团队
  - 技术领导带领的团队。
  - 技术领导是一个积极的技术贡献者，被认为是同类人中的佼佼者。技术领导经常在技术专家而不是职业管理中选择。
  - 一般，技术领导负责制定困难技术问题的最终决策，有时技术领导是一名普通的团队成员，仅仅对团队与管理部门的沟通负有特殊的职责。
  - 从外部看，业务团队的结构是典型的等级层次结构：它通过确定一个人主要负责项目中的技术工作来改善与管理部门的沟通；它允许每一个团队成员在自己的专业领域内工作，允许团队自己安排工作划分。

团队类别（2）

- 首席程序员团队
  - 思想产生于20世纪60年代末期和70年代初期的IBM。
  - 基本的出发点：某些开发者的效率是其他人的10倍
  - 由首席程序员处理大多数的设计和代码，其他团队成员可进行专门的研究，他们被部署为扮演对首席程序员的支持角色。
  - 首席程序员团队在20多年前最初被使用时，它达到了很高的成功，从那以后，许多组织尝试该结构却大多数没有能够成功地重复当年的辉煌，这说明真正的超级程序员有能力充当首席程序员的很少。

团队类别（3）

- 臭鼬项目团队
  - 一个臭鼬项目小组有一批有才华的、有创造性的产品开发者，将他们放在一个不受组织官僚限制的机构中，使他们能放手开发和创新。
  - 臭鼬项目团队是典型的黑箱管理方式。团队可以按照自己的方式进行自我的管理。
  - 该种团队形式的不利方面是没有为团队的进展提供足够的可视度，这样可能会对一些高度创新的工作涉及到的不可预见性有不可避免的影响。
  - 臭鼬项目团队对于创造性最为重要的探索性项目最为合适。

团队类别（4）

- 特征团队
  - 在特征团队方式中，开发、质量保证、文档管理、程序管理和市场人员采用传统的等级报告结构。市场人员向市场部经理汇报，开发人员向开发部经理汇报等。
  - 团队位于这个传统组织的最上方，它从每个部门抽取对产品的功能负有责任的一个或多个成员。
  - 特征团队适合问题解决项目，因为他们有必需的授权和责任来权宜地解决问题。他们也适合创新项目，因为多学科的团队结构可以刺激思维。

团队类别（6）

- 特种武器和战术（SWAT）团队
  - SWAT团队是以军队或警察SWAT团队为基础的团队模式。“SWAT”代表“特种武器和战术”。在这类团队中，每一位成员都被严格训练成某一方面的专家，例如神枪手、爆破专家或高速驾驶员。
  - 在软件行业，SWAT代表“掌握先进工具（Skilled with advanced tools)它最初是James Martin的RAD方法论的一部分。一个SWAT团队的重点是让掌握特定工具或实践的高技能的一组人去解决与这些特定工具或实践有关的问题。
  - SWAT团队的工作不是去创新而是去利用他们熟悉的特定的技术和实践来执行一个解决方案。

团队类别（7）

- 专业运动员团队
  - 专业运动员团队类似于棒球队。
  - 运动员团队的管理者处于幕后决策的地位，管理者的角色是清理障碍，并使开发者可以更有效地工作。
  - 运动员团队具有高度细化的角色。软件团队也一样
  - 这种特定的模式适用于战术执行项目，强调高度细化的个人角色。在这种模式中管理者扮演支持者的角色。

团队类别（8）

- 戏剧团队
  - 戏剧团队是以强烈的方向性和很多关于项目角色的协商为特点的。项目的中心角色是导演，他保持产品的远景目标和指定人们在各自范围内的责任。团队中的个人可以塑造他们的角色，锻造项目中他们负责的部分，但是他们不能走得太远，以至于和导演的目标产生冲突。
  - 戏剧团队的优势在创新项目中，在强烈的中心目标的范围内，提供一种方式来整合巨大的团队个人的贡献。戏剧团队模式尤其适合于被很强的个性控制的软件团队。特别适合于现代的多媒体项目。

团队类别（5）

- 搜索救援团队
  - 软件团队就像一组紧急医疗技师在寻找迷失的登山队员。搜索救援团队重点在解决特定的问题，它将专门的紧急医疗培训培训和登山运动或其它野外生存技能相结合。
  - 在软件方面的搜索和救援就是将特定的软件、硬件的专门知识和特殊业务环境的同样的专业知识相结合。
  - 搜索和救援团队非常适合重点在于解决问题的项目。它太基础，不能支持创造性；太短期，不能支持战术执行。

团队目标和团队结构

	主要目标		
	解决问题	创新	战术执行
主要特征	信任	自治	明确
典型软件工作举例	实况转播系统的校正和维护	新产品开发	产品升级
过程重点	着重于问题	探索可能性和选择性	高度关注有明确角色的任务成功与失败通常被清晰界定
适合的生命周期模型	编码修正模型，螺旋模型	渐进原型，渐进交付螺旋模型，面向进度的设计，面向工具的设计	瀑布模型，修改的瀑布模型阶段交付，螺旋模型，面向进度的设计，面向工具的设计
团队选择标准	理解力强，聪明，感觉敏锐，高度诚实	睿智的、独立的思考者，做事主动，顽强	忠诚，信守承诺，侧重于行动，有紧迫感，积极响应
适合的软件团队模式	业务团队，搜索救援团队，SWAT团队	业务团队，首席程序员团队，臭鼬项目团队，戏剧团队	业务团队，首席程序员团队特征团队，SWAT团队，专业运动员团队

来源：摘自《团队合作》（Larson and LaFasto, 1989）

11.9决策

- 决策可以分成：
  - 结构化：相对简单的，常规的决策，规则可以直接应用
  - 非结构化：更复杂，有一定的创性性
- 决策中遇到的问题
  - 错误的假设
  - 承诺扩大：一旦作出了决策，很难纠正
  - 信息过载

群组决策

- 在项目中经常需要团队一起作出某些决策，当对于某一问题，小组成员的知识是互补性的时候，问题比较容易解决。而在处理结构性很不好的问题时，特别是需要一种创造性解决方法时，效率就不高。头脑风暴技术是一种有效的方法。
- 群组决策中可能遇到的问题：
  - 耗时
  - 激发冲突
  - 决策可能主要由影响大的成员作出

减少群组决策缺点的方法

- **Delphi技术**
  - 选定需要合作的专家
  - 将问题提给每个专家
  - 专家提出建议
  - 建议被收集并进行整理
  - 建议发回给专家
  - 专家对其它人的建议提出意见并修改自己的建议
  - 如果已经有一致的结果，则结束，否则继续进行。

11.10团队领导

- **Leadership**领导能力指的是在一个小组中影响他人并使他们为了小组目标而工作的能力。
- 领导力是与权力相联系的：
  - 地位权力
    - 强迫权力
    - 联系权力：与具有权力的人能够联系
    - 报酬权力
  - 个人权力
    - 专家权力：能够作一些特殊任务
    - 信息权力：获取其它人所得不到的信息的权力
    - 指示权力：基于领导者的个人吸引力

领导风格

- 一种分类
  - **Directive Autocrat(直接独裁):**单独决策并对实施进行密切监控
  - **Permissive Autocrat(许可独裁):**单独决策但是给下属以实施的自由
  - **Directive Democrat(直接民主):**参与性决策但是对实施进行密切监控
  - **Permissive Democrat(许可民主):**参与性决策但是给下属以实施的自由
- 另一种分类
  - 面向任务
  - 面向人员
  - 对于不确定性的环境，面向任务的更受欢迎
  - 对于没有经验的人员，面向任务更有效

管理风格

- 问题：什么样的管理风格更合适，是面向任务的还是面向人员的？
  - 学院工资系统项目中，一个以前当地政府中维护工资系统很多年的人员加入到该项目
  - 一个新的分析员/编程人员加入到一个开发小组
  - 一个年龄在45岁曾经在老的帐务系统上干过系统支持工作的人员加入到对帐务系统进行改造的项目中来
- 在开始时面向人员的风格比较合适
- 对于该新的人员，面向任务和面向人员的管理都需要
- 因为新的扩展将影响该人员的工作，在短时间内，需要进行面向任务的管理

11.11组织结构

- 正式的，非正式的结构
  - 尽管组织结构对项目有很大影响，但是项目经理有时没有什么发言权
  - 一般而言，正式的结构可以表示称人员的递阶结构
  - 非正式的结构成员之间的关系是交叉的。
- 递阶方法
- 人员和生产线
  - 直接生产产品的线上人员
  - 支持人员

组织结构

- 面向功能的分解
  - 人员更有效的使用，例如编程人员可以在需要的时候分配给某项工作，干完后分配给另外的工作
  - 人员的职业发展是面向技术的
  - 技术人员之间可以交流新技术，新思想
  - 可能在不同的部门之间存在交流的问题，特别在编程者缺乏应用知识时
- 面向任务的分解
  - 用户更愿意看到面向项目的结构，因为资源实现已经给定给这个项目
  - 职业的发展可能使编程人员最后成为系统分析员
- 面向生命周期的分解
  - 开发和维护由不同的团队完成
- 矩阵结构
  - 职员有两个经理

组织结构

- 集中式结构
  - 团队成员之间可以进行自由通信
- 分布式结构
  - 分成若干部分，每个部分有一个代表
  - 部分之间的通信由代表来进行
- 非利己主义编程**(Egoless programming)**
  - **Peer review**以保证程序是小组的公共财产
- 主编程团队（**Chief Programmer Team)**
  - 团队越大，效率越低，因而需要降低团队规模
  - 主编程完成分析，设计，编码，测试和文档编写
  - 为了辅助主编程工作，配备了副手帮助写文档，写程序，测试等
  - 合格的主编程人员难以寻找

11.12项目工作中的冲突（1）

- 项目中的冲突是不可避免的。
- 冲突并非一无是处，它让人们有机会获得新的信息，另辟蹊径，制定更好的问题解决方案，加强团队建设，这也是学习的好机会。

项目工作中的冲突（2）

- 冲突来源
  - 工作内容
  - 资源分配
  - 进度计划
  - 成本
  - 先后次序
  - 组织问题
  - 个体差异

项目工作中的冲突（3）

- 冲突处理
  - 回避或撤退
  - 竞争或逼迫
  - 调停或消除
  - 妥协
  - 合作、正视和解决问题