Software Project Management (9)

Du Yugen

ygdu@sei.ecnu.edu.cn

Chapter 9 Monitoring and control

软件项目监督和控制

Roger Du@SEI of ECNU

OBJECTIVES

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

1. **Monitor the progress of project;**
2. **Assess the risk of slippage;**
3. **Visualize and assess the state of project;**
4. **Revise targets to correct or counteract drift;**
5. **Control changes to a project's requirements**

9.1项目监控的内容

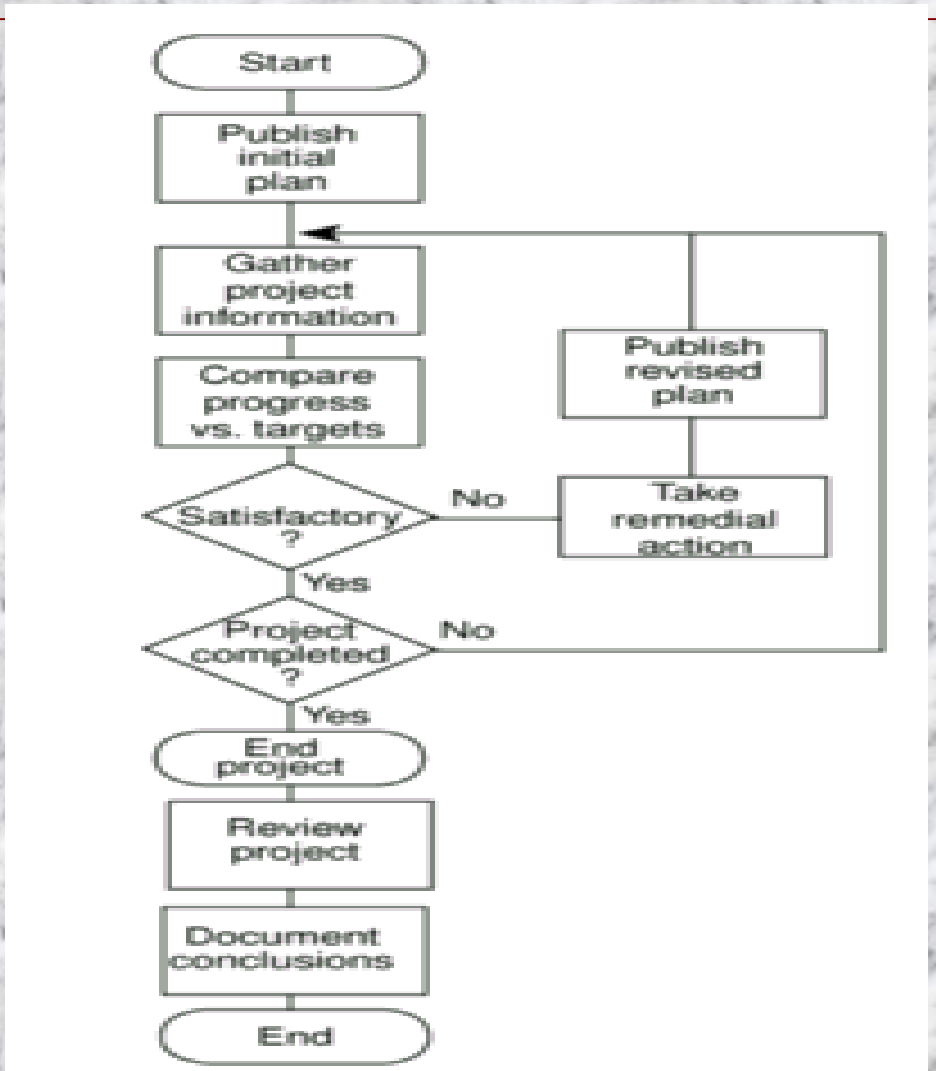
- **监控项目的进展**
- **比较实际进度与计划的差别**
- **修改计划使项目能够返回预定“轨道”**

9.2项目监控框架framework：过程

关心的四个问题：

- 工期拖延
- 质量不过关
- 功能不合适
- 成本超出预算

本章主要关心第一、第四个问题



9.2.1项目监控框架：责任（1）

- **项目指导委员会（Project Steering Committee， Project Board）负责整个项目进度**

项目监控框架：责任（2）

- **项目情况报告的内容**

Table 9.1 *Categories of reporting*

Report type	*Examples*	*Comment*
Oral formal regular	weekly or monthly progress meetings	while reports may be oral formal written minutes should be kept
Oral formal ad hoc	end-of-stage review meetings	while largely oral, likely to receive and generate written reports
Written formal regular	job sheets, progress reports	normally weekly using forms
Written formal ad hoc	exception reports, change reports	
Oral informal ad hoc	canteen discussion, social interaction	often provides early warning; must be backed up by formal reporting

9.2.2项目监控框架：进度评估assessing progress

- 基础：定期信息收集或者发生的特定事件
- 这些信息必须是客观的和可度量的
- 但是并非每一次都能够得到符合要求的信息，因而通常需要项目成员进行主观判断

9.2.3项目进度监控：检查点设置

- **检查点(Checkpoints)包括:**
  - **定期的（如一星期一次，一月一次）**
  - **与特定的事件绑定的，如生成一份报告或者交付部分产品**

9.2.4项目监控框架：监测频率frequency

- 监测的频率依赖于项目的大小和风险情况
  - 团队领导，可能需要每天都了解一下进度
  - 项目经理需要每星期或每月了解情况
- 管理层次越高，频率越低，信息越抽象
- 许多公司利用星期一早晨的短会来激励员工实现短期目标

9.3数据收集

- 尽管整个过程被分成了容易管理的活动，但是项目执行中仍然需要在活动中对任务完成的比例进行评估，这种评估通常是困难的。
- 思考：某一软件开发者完成了一个需要500行代码的软件的250行，请解释一下为什么不能认为他的工作已经完成了一半?

答案

- 许多因素决定了不能用完成的代码行的比例来衡量进度：
  - 对整个软件的代码行的估计可能不准确
  - 写完的代码可能相对容易，或者相对容易
  - 一个软件如果没有通过测试就不能算完成，因而即使代码全部写完了，如果没有测试也不能算完成。
- 对所需完成内容的深入的了解有助于判断进度，如将整个工作细分为子任务，如设计，编码，单元测试等。

9.3.1部分完成报告

- 许多组织采用财务系统中的每周时刻表来记录每个职员在每项工作中花费的时间，但是该表无法告诉项么，进度是否满足要
- 因而可以对每周时刻完成的工作内容

Time Sheet

Staff John Smith Week ending 26/3/99

Rechargeable hours

Project	Activity code	Description	Hours this week	% Complete	Scheduled completion	Estimated completion
P21	A243	Code mod A3	12	30	24/4/99	24/4/99
P34	B771	Document take-on	20	90	1/4/99	29/3/99

Total recharged hours	32

Non-rechargeable hours

Code	Description	Hours	Comment & authorization
Z99	day in lieu	8	Authorized by RB

Total non-rechargeable hours	8

9.3.2风险报告

- 询
- 交
- 识
- 将
- 对
  - —
  - —
  - —

Activity Assessment Sheet

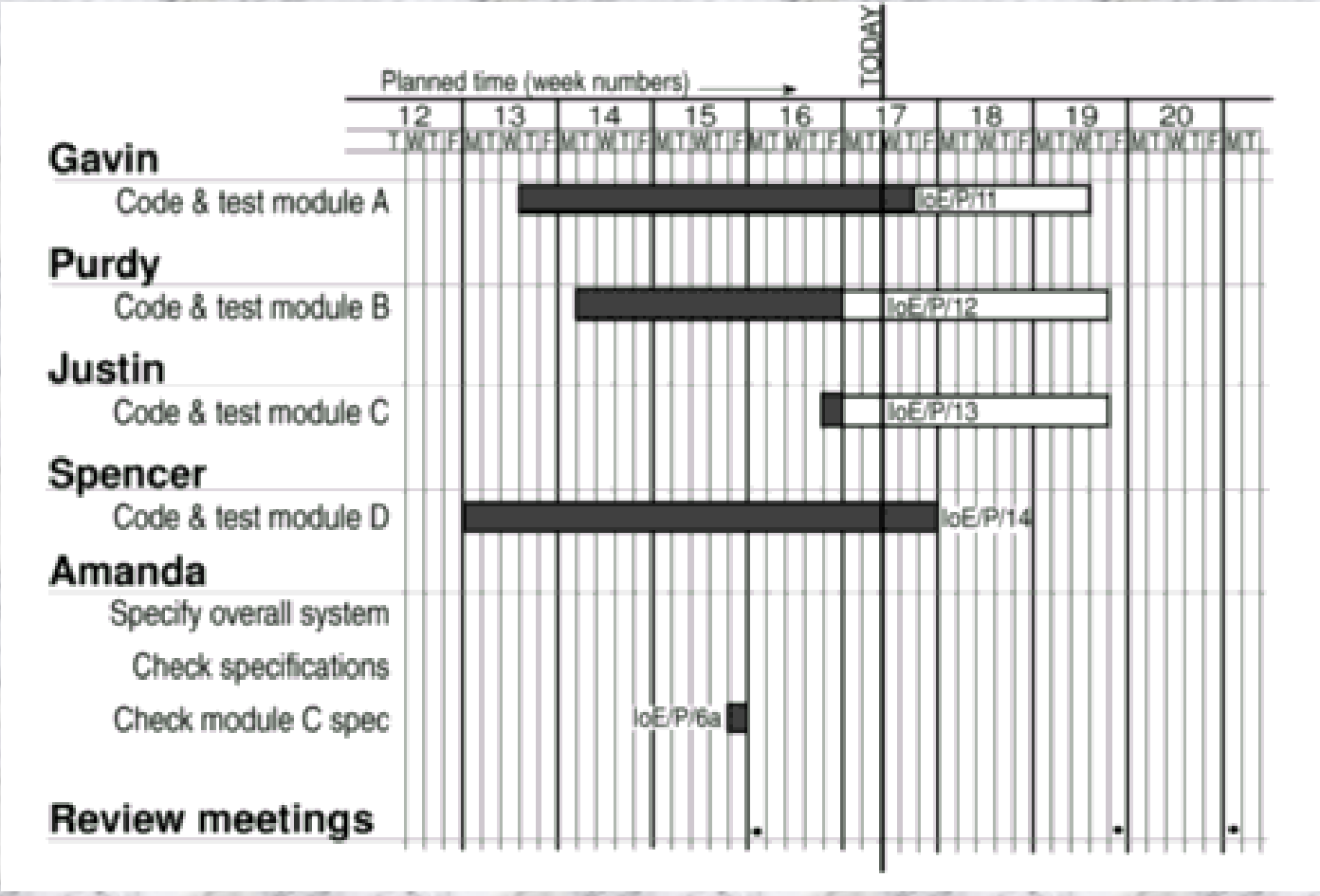
Staff Justin

Ref: IoE/P/13 **Activity: Code & test module C**

Week number	13	14	15	16	17	18	
Activity Summary	G	A	A	R			
Component							*Comments*
Screen handling procedures	G	A	A	G			
File update procedures	G	G	R	A			
Housekeeping procedures	G	G	G	A			
Compilation	G	G	G	R			
Test data runs	G	G	G	A			
Program documentation	G	G	A	R			

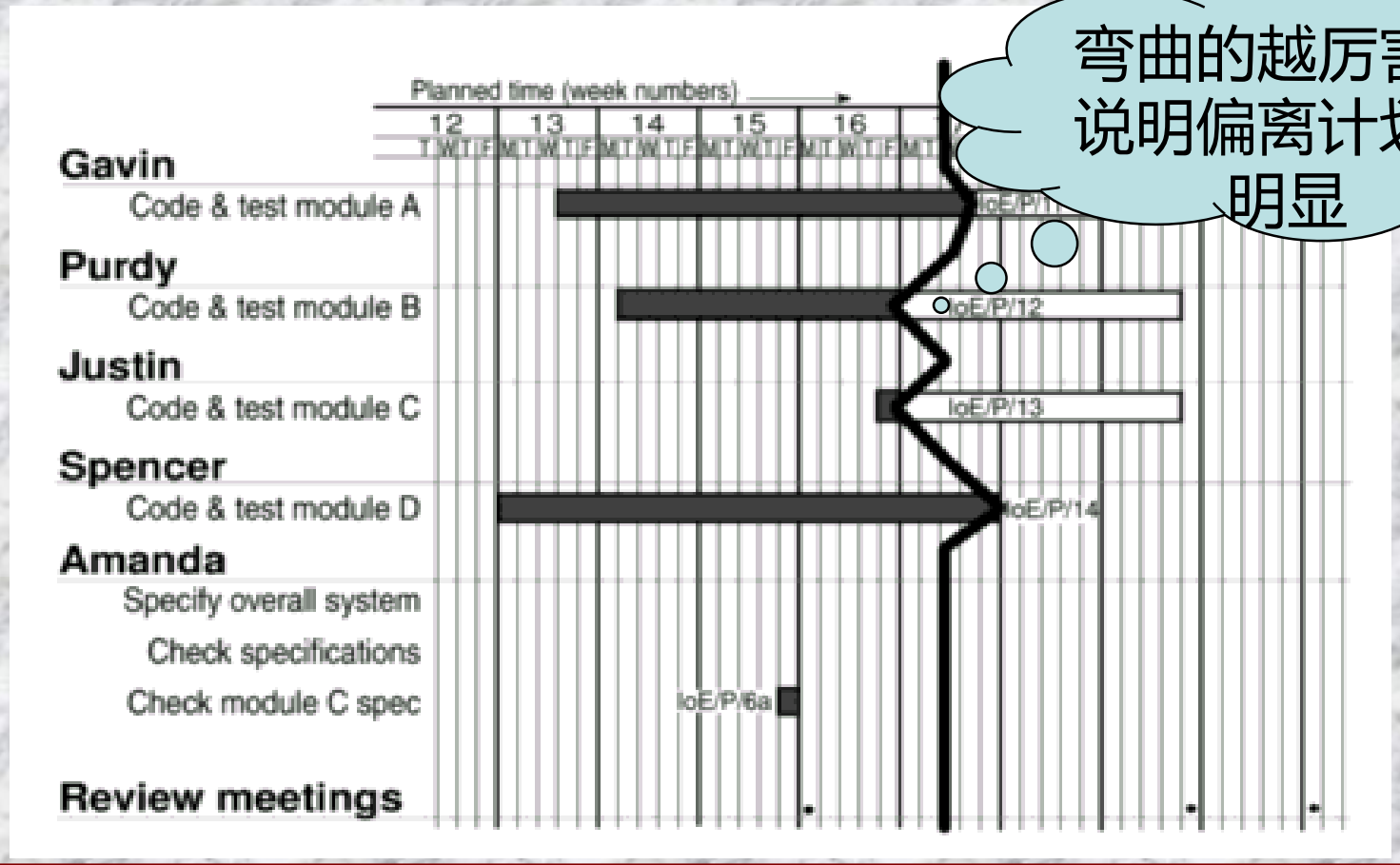
9.4进度可视化

- **Gantt图**



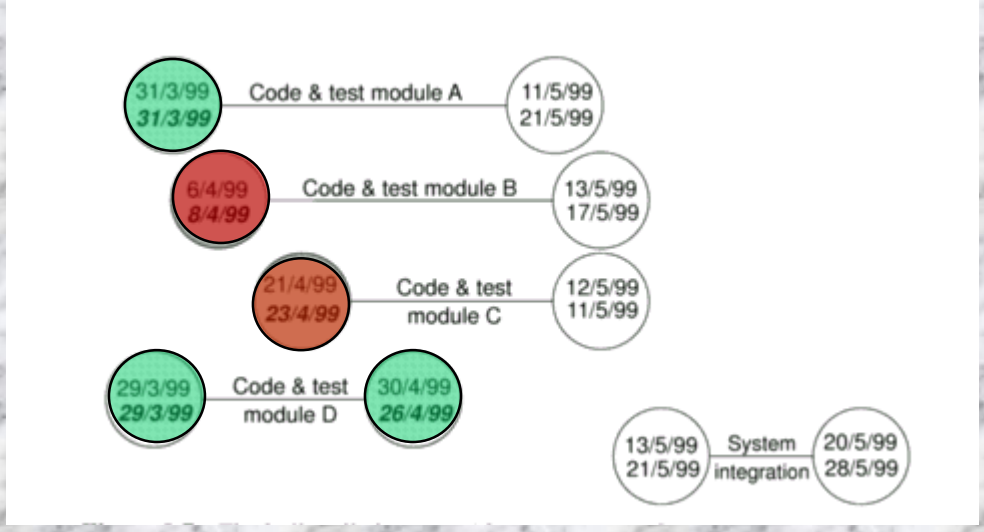
进度可视化

- 滑动图(slip chart)



进度可视化

- **球图：计划开始，计划结束作为两个球，每次计划改变后，日期添加到球中，如果时间是按计划的，球被填为绿色，否则被填为红色。**
- **每次更新后，图不需重画。**

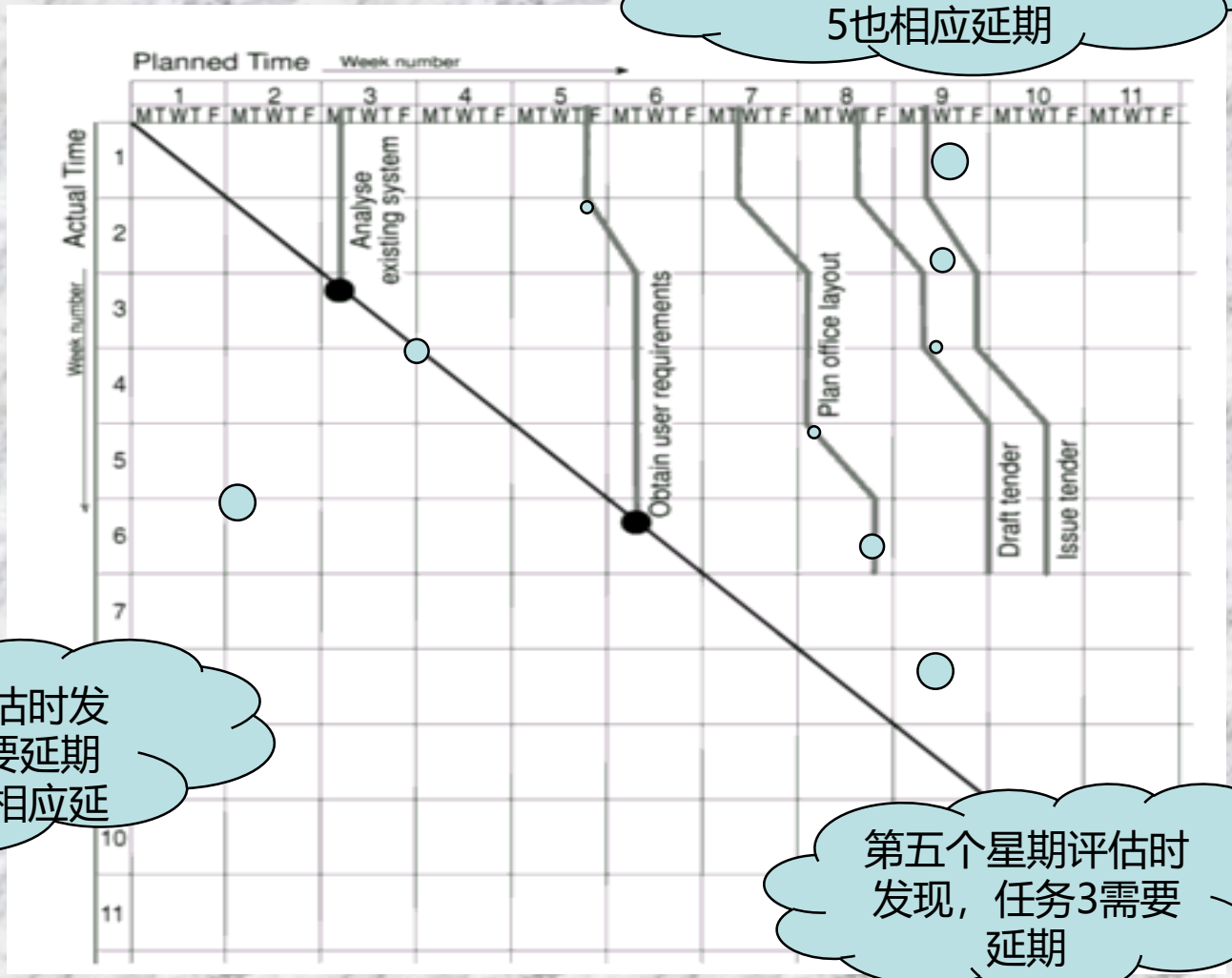


进度可视化

- **前面的方法不能表示出项目生命周期中偏离计划的情况。**
- **对计划偏离的趋势分析能够避免将来的项目偏离。**
- **时间线图（timeline）**

进度可视化

计划的时间



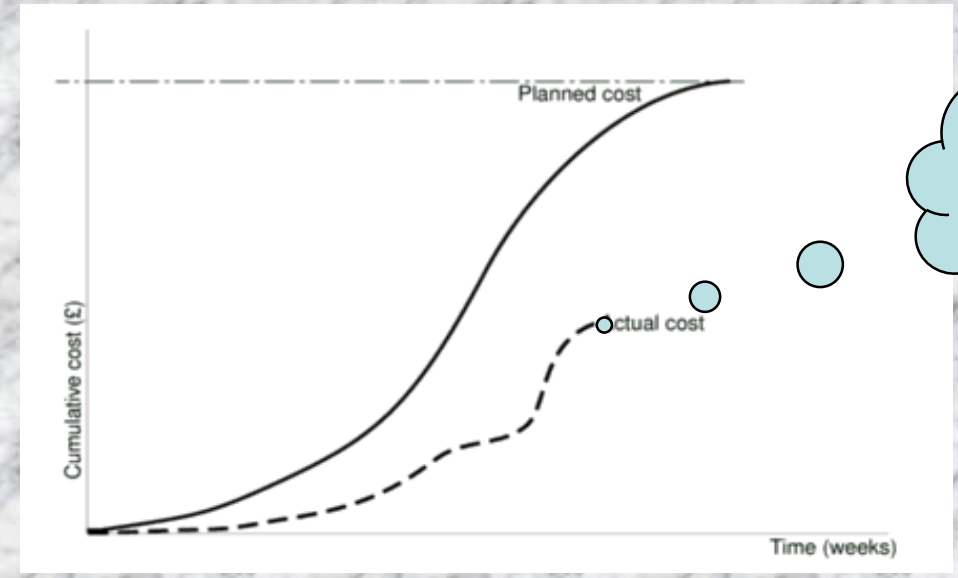
第四个星期评估时发现，任务4需要延期，任务5也相应延期

第二个星期评估时发现，任务2需要延期，其它任务也相应延期

第五个星期评估时发现，任务3需要延期

9.5成本监控

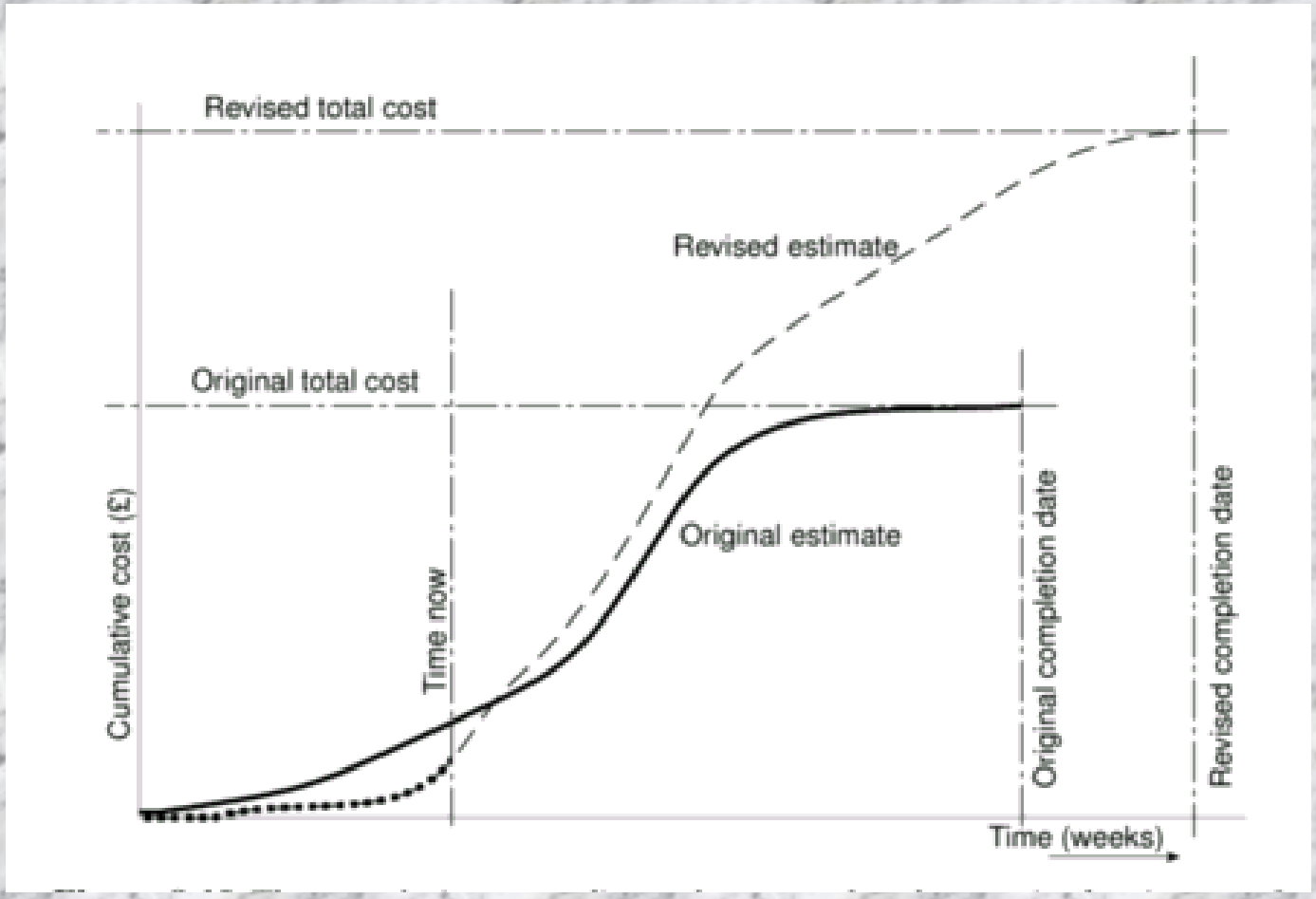
- **监控的意义**
  - **成本本身是项目中的重要元素**
  - **成本监控也能展示已经花费了多少劳力**
- **简单的监控方法：累积消耗图**



不能说明项目进展情况

累积消耗图

- **对普通的累积消耗图上加上项目时间信息**



9.6赢得值(挣值,盈余值) Earned Value

- 赢得值（Earned Value)分析:建立在对每个任务或工作包的原始消耗预测的基础上给其分配一个值。
- 这个分配的值是对每一项内容的原始预算成本，被称为基线预算(baseline budget)或计划工作的预算成本（budgeted cost of work scheduled, BCWS)。
- 未开始的任务被赋值为0，当任务被完成后，将被赋成本值。对项目中的任一点上，相关任务全部的值将被称为赢得值或完成工作的预算成本（budgeted cost of work performed, BCWP)，这个值可以表示为BCWS或BCWS的一个百分比.

挣值

当任务已开始但未完成时，需要分配一个赢得值给该任务，方法为：

- 0/100技术：任务被分配值0直到任务完成后，被分配预算值的100%
- 50/50技术：任务一开始后，就赋予50%,直到项目结束后赋值100%
- 里程碑方法：对任务中的一系列里程碑赋予特定值。
- 建议用0/100方法，因为50/50方法由于活动开始时报告的估值过高，容易给人一种错误的安全感，而里程碑方法对长周期活动或许是合适的,但最好将该任务细分为多个子任务。

9.6.1基线预算

- 建立赢得值分析的第一步是为项目建立一个基线预算 (baseline budget)
- 预算基线是建立在项目计划的基础上的，它是随时间显示挣值值的预测。
- 挣值可以用货币单位来衡量，在人员密集的案例里,如软件开发,通常可以用人员工作量(人-时或工作日)来衡量。

例子

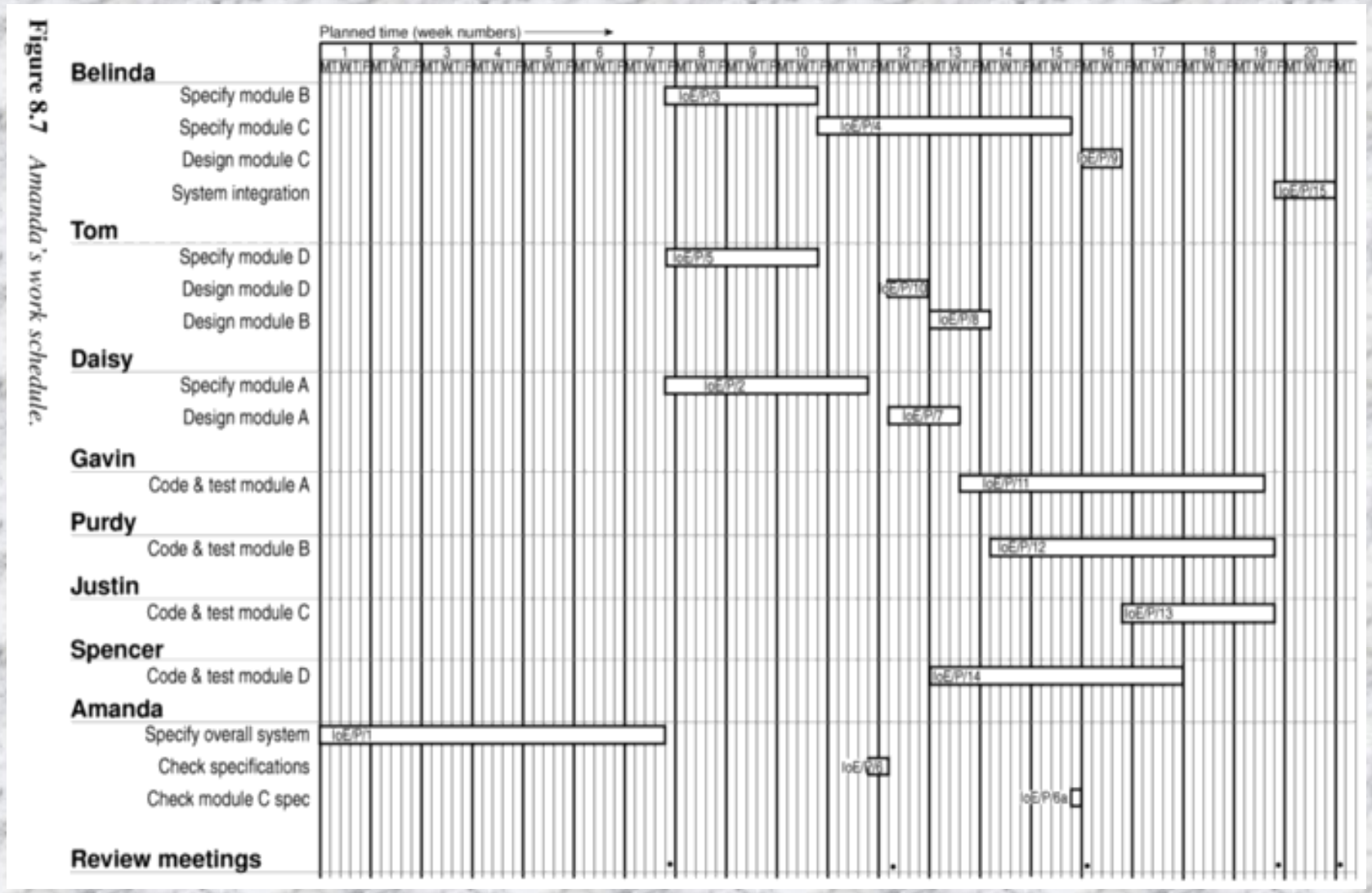
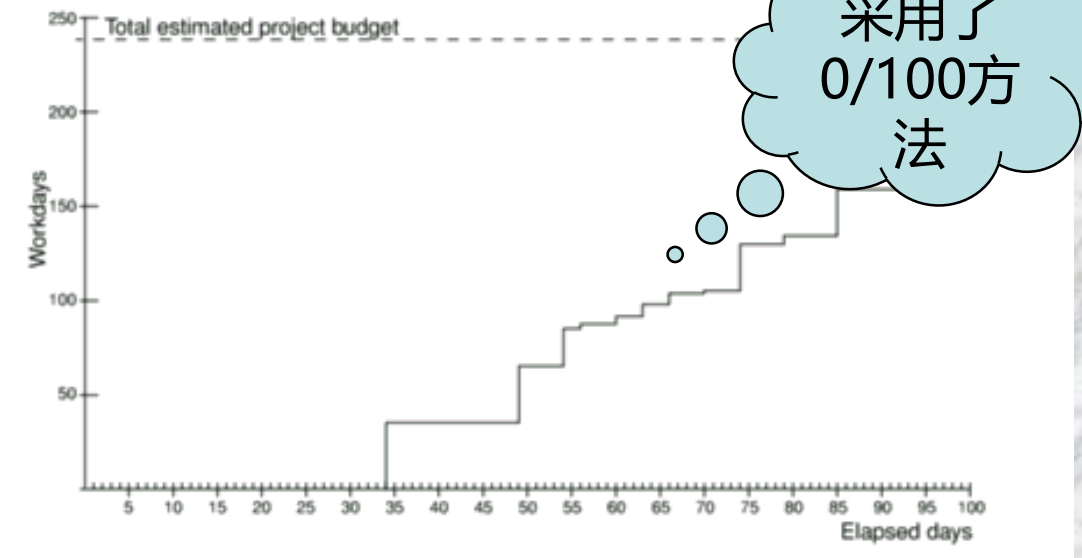


Figure 8.7 *Amanda's work schedule.*

例子

Table 9.2 *Amanda's baseline budget calculation*

Task	*Budgeted workdays*	*Scheduled completion*	*Cumulative workdays*	*% cumulative earned value*
Specify overall system	34	34	34	14.35
Specify module B	15	49	64	27.00
Specify module D	15	49		
Specify module A	20	54	84	35.44
Check specifications	2	56	86	36.28
Design module D	4	60	90	37.97
Design module A	7	63	97	40.93
Design module B	6	66	103	43.46
Check module C spec	1	70		
Specify module C	25	74		
Design module C	4	79		
Code & test module D	25	85		
Code & test module A	30	93		
Code & test module B	28	94		
Code & test module C	15	94		
System integration	6	100		



采用了0/100方法

9.6.2挣值监控

- **随着项目的进行，可以不断进行挣值监控，判断项目的进度。**

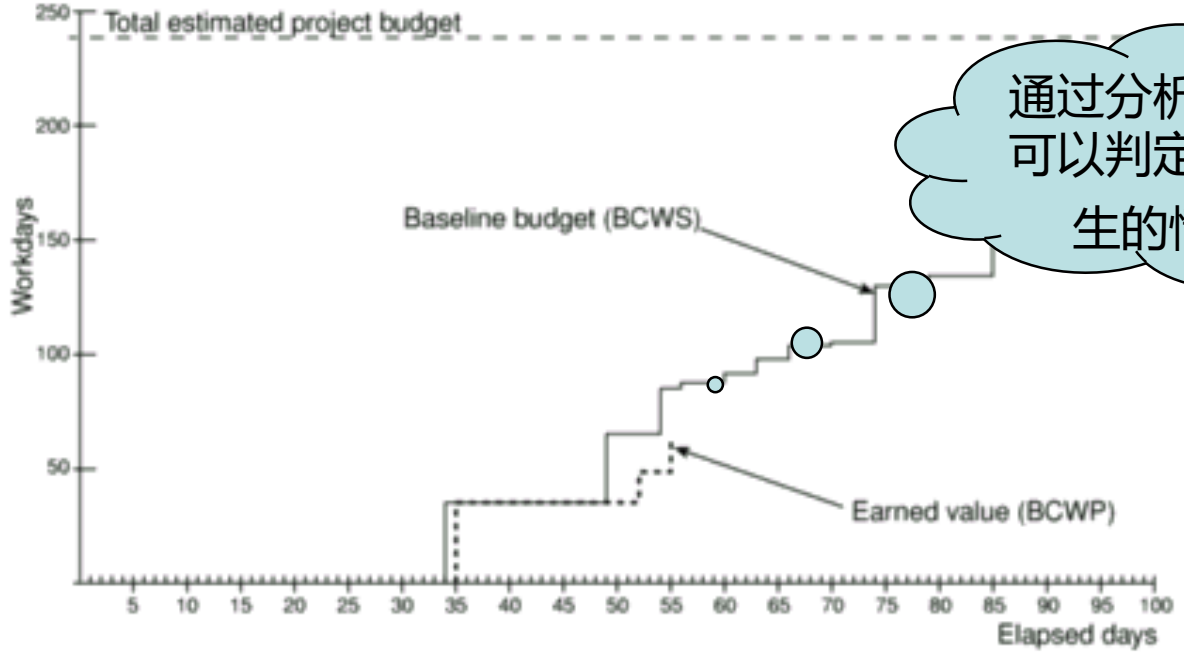
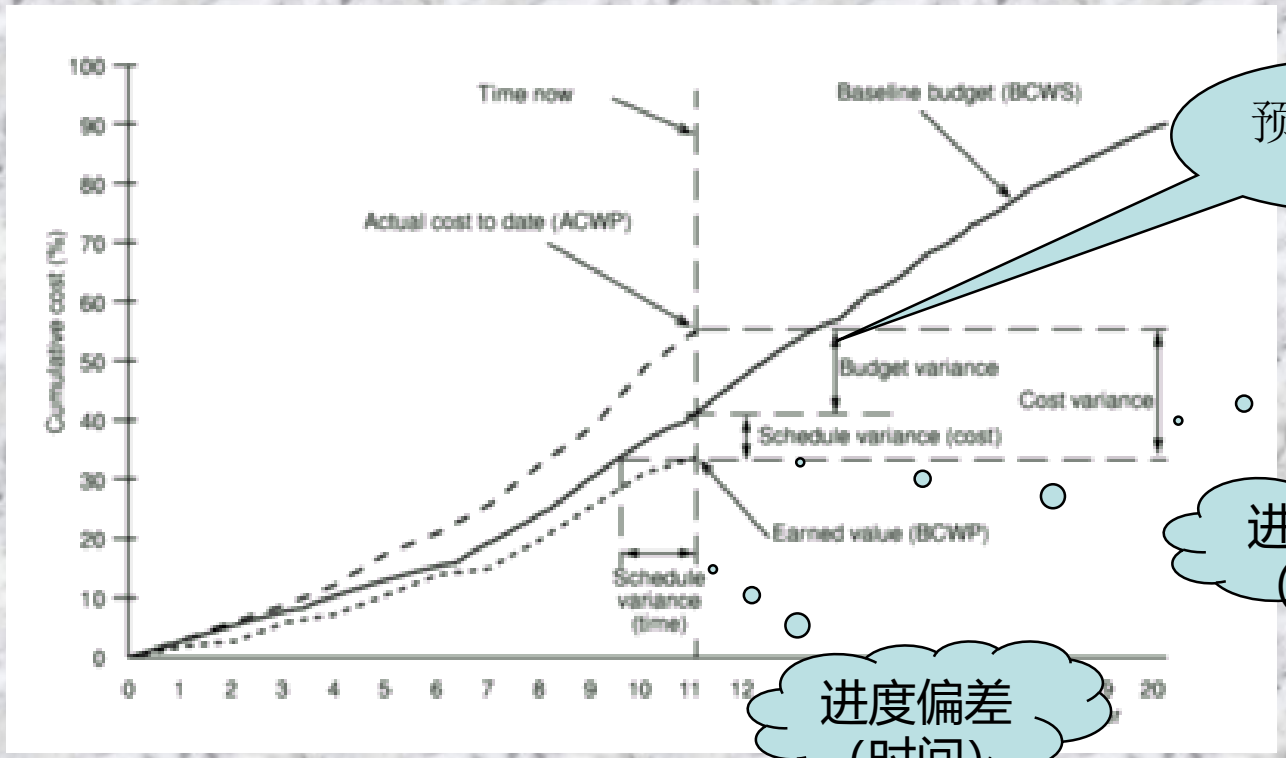


Figure 9.12 *Amanda's earned value analysis at week 12.*

挣值监控

- **每一项任务的实际成本消耗为（Actual Cost work performed, ACWP)**



基本公式

简称	说明	公式说明
PV	计划值	计划完成工作的预算值
EV	挣值	实际完成工作的预算值
AC	实际成本	实际花费成本
BAC	完工预算	整个项目的预算
EAC	完工估算	实际成本+完工尚需估算 AC+ETC
ETC	完工尚需估算	剩下的工作还需多少钱
SV	进度偏差	挣值-计划值 EV-PV
SPI	进度偏差指数	挣值/计划值 EV/PV
CV	成本偏差	挣值-实际成本 EV-AC
CPI	成本偏差指数	挣值/实际成本 EV/AC
PC	完工百分比	挣值/完工预算 EV/BAC
PS	花费百分比	实际成本/完工预算 AC/BAC
TCPIC	尚需竣工绩效指数(成本的)	（BAC-EV）/（BAC-AC）
TCPIS	尚需竣工绩效指数(进度的)	（BAC-EV）（BAC-PV）

赢得值监控

- 预算变动(Budget variance):ACWP-BCWS
- 进度变动(Schedule variance)
- 成本变动(Cost variance):BCWP-ACWP
- 性能比例(Performance ratios):
  - 成本性能指数：CPI = BCWP（赢得值）/ACWP（实际的成本消耗）
  - 调度性能指数：SPI=BCWP/BCWS(预算成本)
  - 值越大，工作完成得越好

例子

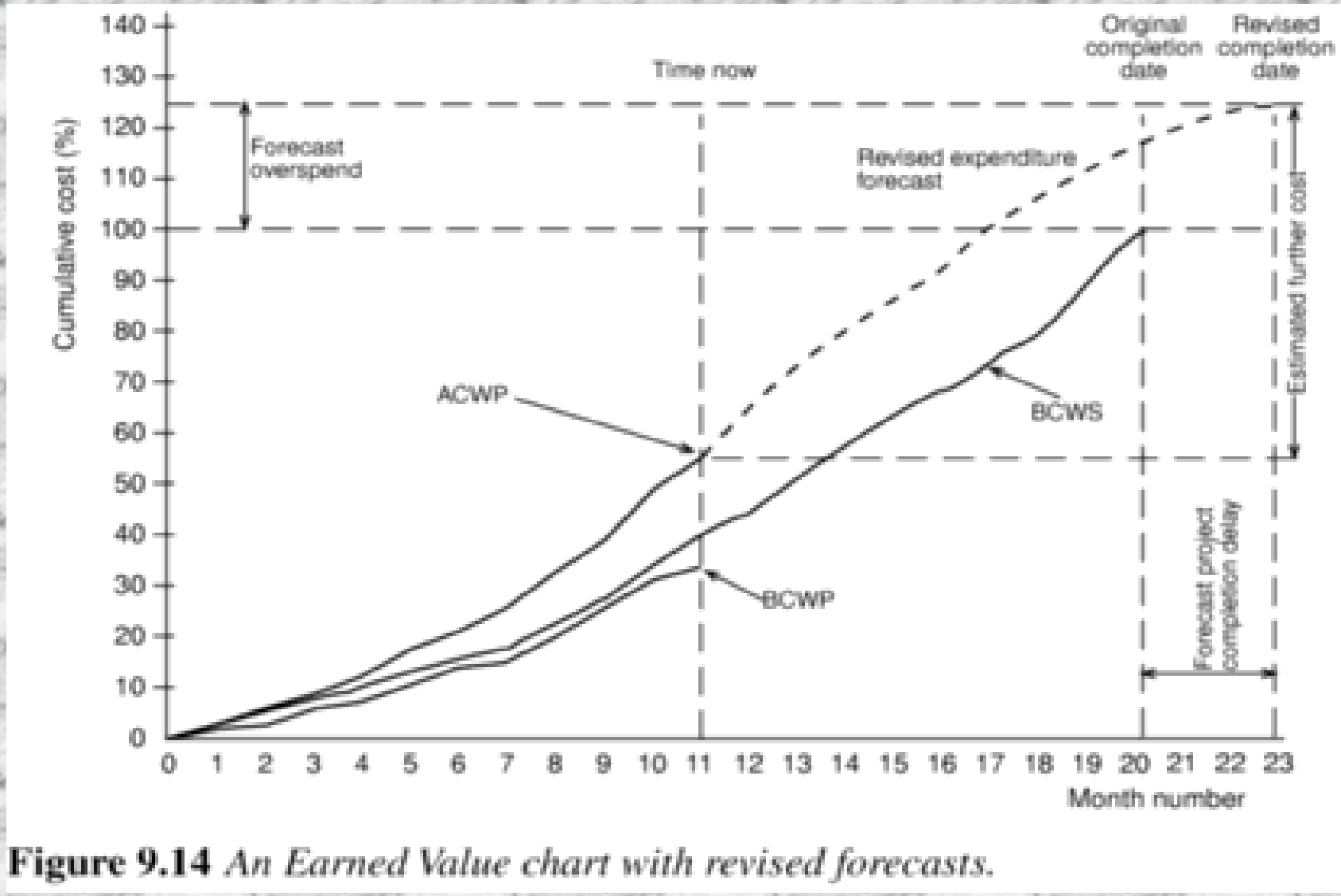
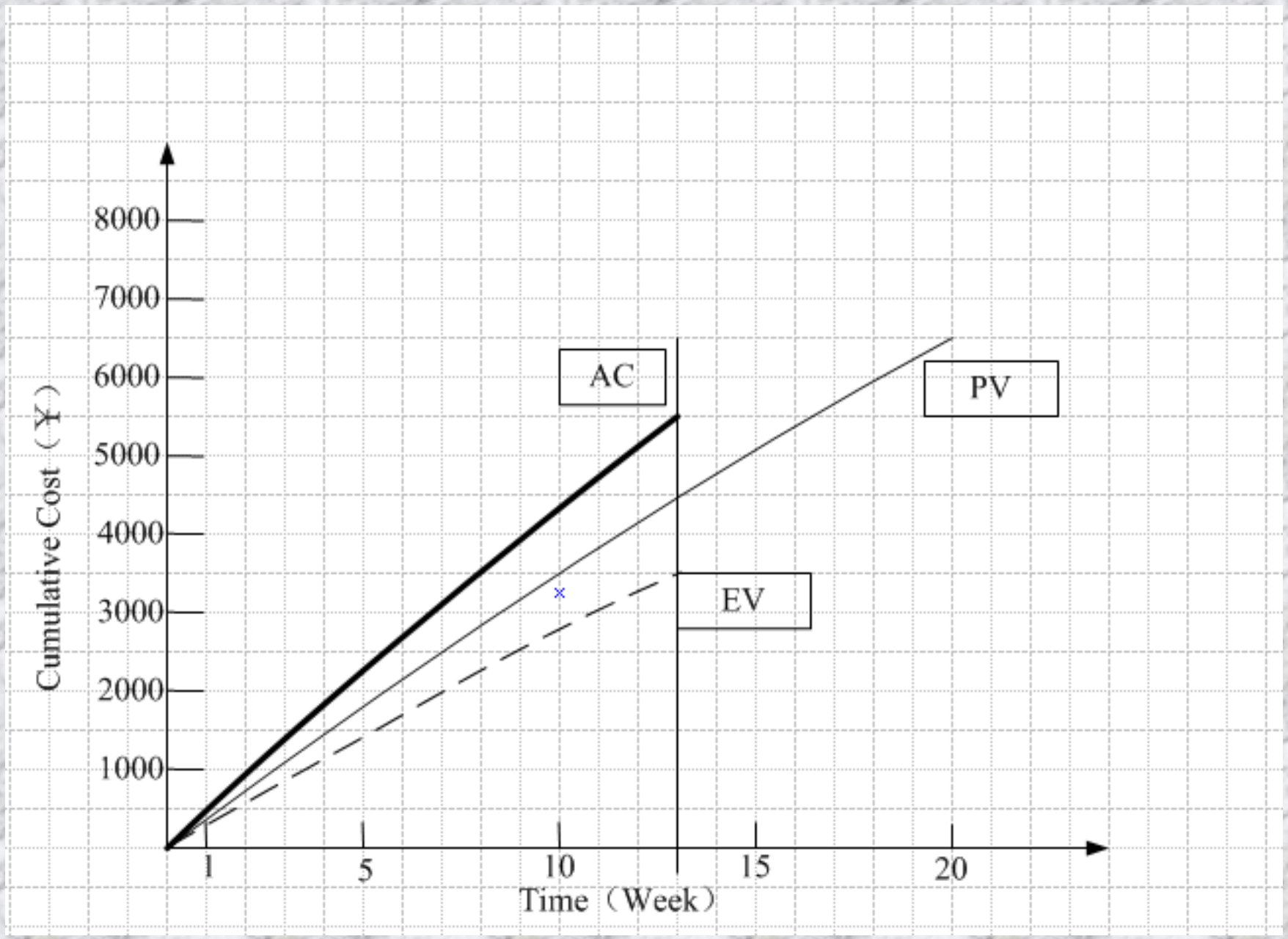
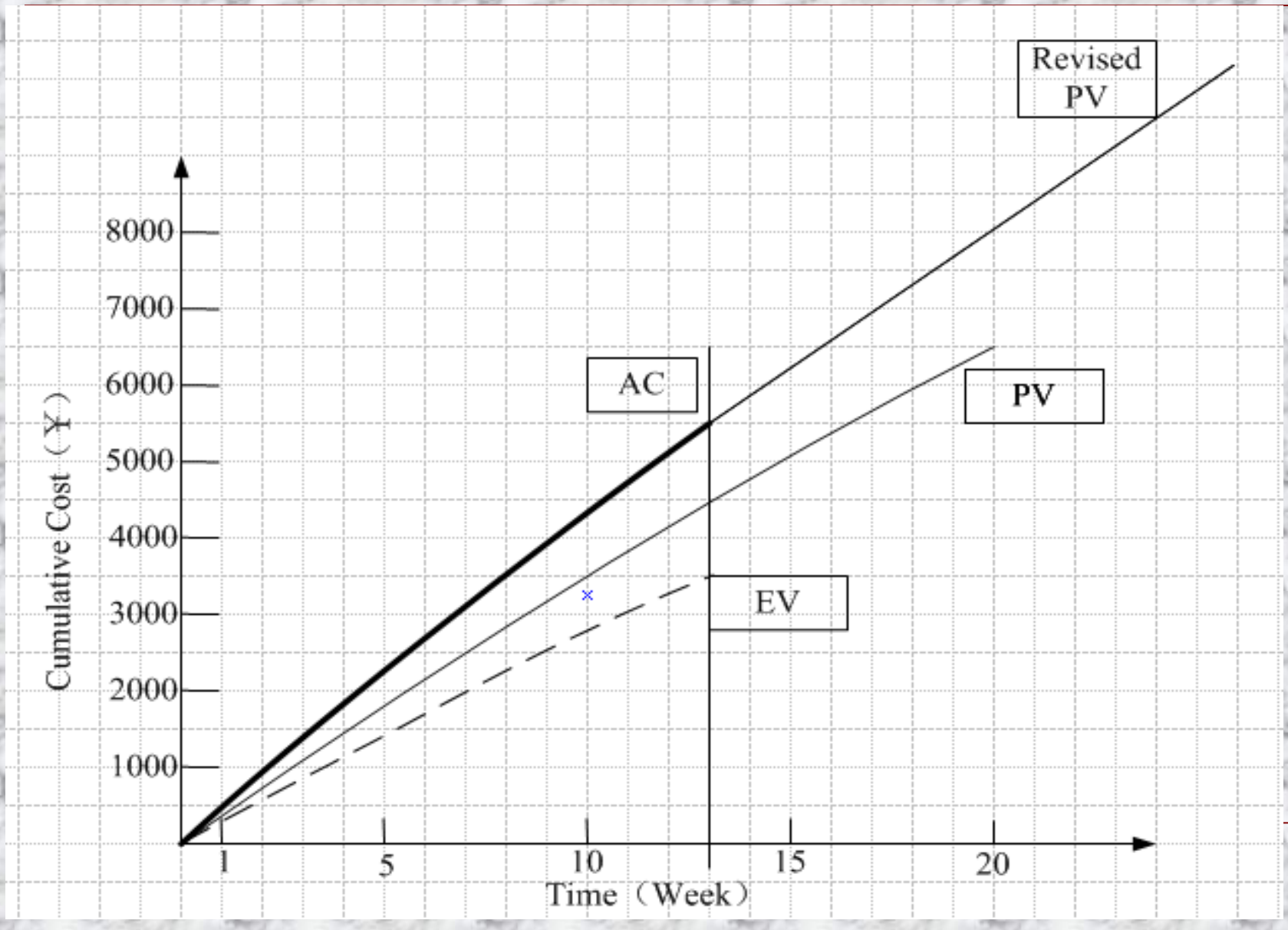


Figure 9.14 *An Earned Value chart with revised forecasts.*


8000
7000
6000
5000
4000
3000
2000
1000
Cumulative Cost（¥）
AC
PV
EV
1
5
10
15
20
Time（Week）

CPI＝EV/AC=3500/5500=0.636　　SPI=EV/PV=10/13=0.769

EAC=6500/CPI=10214　　TEAC=20/SPI=26



挣值监控

- **挣值概念还没有被软件界全面接受，原因可能在于建了一半的房屋可以有反映人力和材料消耗的记录，而完成一半的软件项目却没有任何数据。这是对挣值值分析的误解。实际上挣值分析是一项跟踪项目进度的方法。**

9.7监控的优先级Prioritizing monitoring

列出优先级以决定监控的层次

- **关键路径活动**
- **没有自由浮动的活动**
- **小自由浮动时间的活动**
- **高风险的活动**
- **使用关键资源的活动**

9.8使项目回到正规Getting the project back to target

- 几乎所有的项目都会遇到延误和意外事件。项目经理的一项任务就是识别这些事件发生的时间，在最小延迟时间和对项目团队有最小的影响的情况下，消除问题的影响。

9.8.1缩短关键路径

- 要求项目组人员“Work harder”有一些效果，但是不能轻易使用。
- 分配额外的资源可以加快进度，但是并不总是奏效，例如分配给某一人员的小模块，再增加一个人员并不一定能够缩短时间。
- 将非关键路径上的资源调整到关键路径上
- 注意：缩短关键路径可能使其它路径成为关键路径。

9.8.2重新考虑任务优先关系

- 网络计划考虑的是理想情况和普通工作情况，因而在无法缩短关键路径时，可以重新考虑任务优先关系。
- 另一种方法是将活动再进行划分，从而一部分可以与其它活动并行。
- 重新考虑任务优先关系可能带来风险或者质量上的影响。

9.9变更控制（Change Control)

- 用户的需求可能变化，项目内部可能变化……
- 变更需要仔细考虑，因为一个部分的变化可能会对另外部分的造成影响
- 问题：对程序描述的改变将引起软件的设计和代码的改变，还有什么其它产品可能需要修改?
- 答案：测试数据，期待结果和用户手册等

9.9.1变更管理员角色

- **Configuration Librarian**
- **责任:**
  - **识别所有需要变更控制的内容**
  - **建立一个保存所有项目文档和代码的中央库**
  - **建立一套管理变更的正式过程**
  - **维护读取库中内容的记录和库中每一项的状态**

9.9.2变更控制过程

- 用户意识到需要对系统进行修改，考虑将修改请求提交给开发人员
- 用户端的管理者考虑是否将该修改请求提交给项目承担者
- 项目承担端的管理者将该任务指派给一个成员，该成员将判断该修改的成本以及修改的影响，并提交一个报告
- 该报告被提交给用户，用户将考虑是否能够承受额外的成本

变更控制过程(续)

- **用户同意后，一个获多个开发者被授权从主产品上取出要修改的部分的拷贝**
- **拷贝被修改。**
- **新版本开发出来后，将通知用户，用户进行接受测试**
- **当用户满意后，产品的配置项被新版本所代替。**

9.10项目修复

- **需要修复的项目**
  - **没有人对项目何时结束有一点点概念**
  - **产品满目疮痍。**
  - **开发组人员工作超时，每周多于60小时**
  - **管理层已经无法控制进度，而评估项目状态的准确性丧失殆尽**
  - **客户对开发组能否按承诺交付软件不再抱有信心**
  - **开发人员，市场人员，项目经理，客户之间关系紧张**
  - **开发组士气低落**
  - **……**

修复方案

- **问题：如何挽救项目**
  - **缩减项目规模，以便在计划的时间与工作量内完成项目**
  - **把注意力放在短期的改善上，以提高过程的生产率**
  - **面对现实，放弃计划**
- **有没有其它方法?**

重新获取控制权

修复计划

- 第一步
- 评估你的处境
- 应用W-理论分析
- 作好修复项目的思想准备
- 向开发组成员探问拯救项目的方法
- 变得现实一些

项目修复：人员

- 采取一切措施恢复开发组的士气
  - 采取一个象征性的行动，如给他们特许的条件（允许他们上班晚些，提供更好的工作环境），也可以放一次假
- 确保为开发组创造了条件
  - 如去掉了过多的进度压力，改善了恶劣的工作条件，剔除了管理上的不当做法
- 消除重大的人员问题
  - 勇敢地面对问题，该调整的要调整

项目修复：人员

- 消除重大领导问题
  - 更换子项目经理
  - 为经理配备助手
- 增加新手一定要慎重
- 充分利用开发人员的时间
  - 减轻他们其它的负担
  - 为他们处理一些日常工作
- 允许开发组人员各有不同
  - 绝不允许破坏士气
- 观察开发人员的节奏
  - 不断根据修复的情况来调整安排

项目修复：过程

- 修正软件开发过程中出问题的环节
  - 出问题的环节必然是项目有意或无意忽略了软件开发的基本原则
- 创建详细的小型里程碑
  - 小型的（一、两天规模的）
  - 二元性（要么完成，要么没有完成）
  - 彻底性（所有里程碑完成，项目就完成了）
- 依据里程碑的完成来安排进度
  - 为每个里程碑设置完成的时间，里程碑的设置必然是考虑了人员正常的工作时间

项目修复:过程

- 细致地最终进度进展情况
  - 每天检查小型里程碑的完成情况
  - 误了某个里程碑，就要求他加班加点完成
- 记录里程碑未完成的原因
- 短期后再调整——一周或两周
- 慎重提交进度计划
  - 经过一两周的实际检验
- 严格的风险管理

项目修复：产品

- 稳定需求
- 修正特性集
- 删除优先级低的需求
- 评估你的政治地位
  - 是否本身产品就是不重要的?
  - 目前比产品更重要的是什么?
- 去除产品中没用的“垃圾”
- 降低缺陷数目，并要持续降低
  - 公开缺陷图