软件工程

第十四章　软件实施与维护

乔立民

qlm@hit.edu.cn

2011年6月6日

主要内容

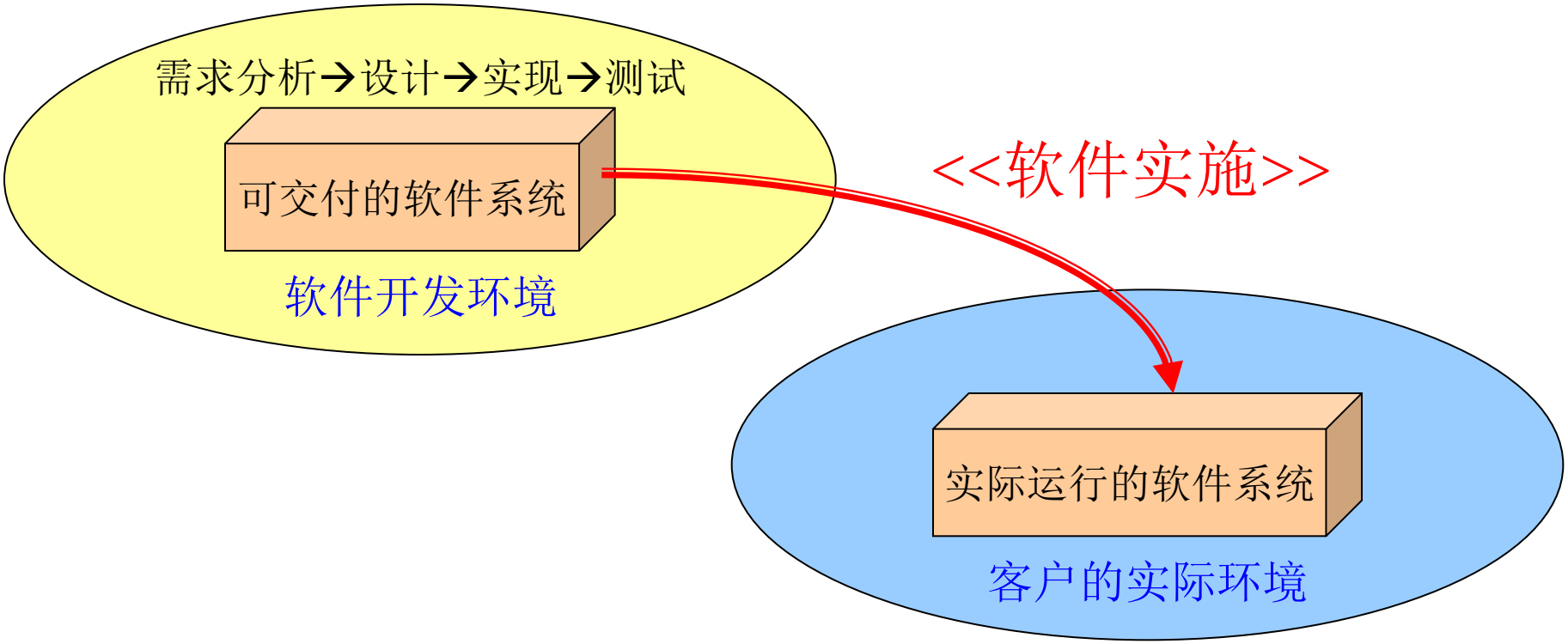
14.1 软件实施

14.2 软件维护

14.3* 软件再工程

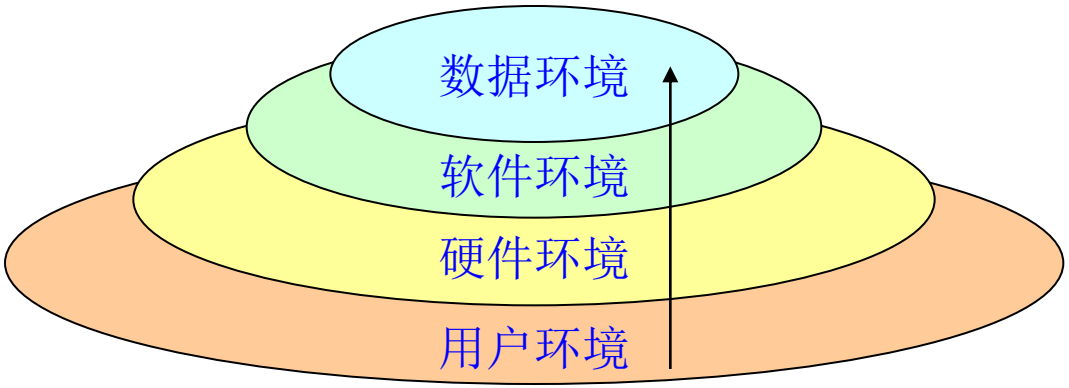
1 软件实施

- **软件实施(Software Implementation)：将新系统设计方案与软件系统转换成实际运行系统的全过程。**

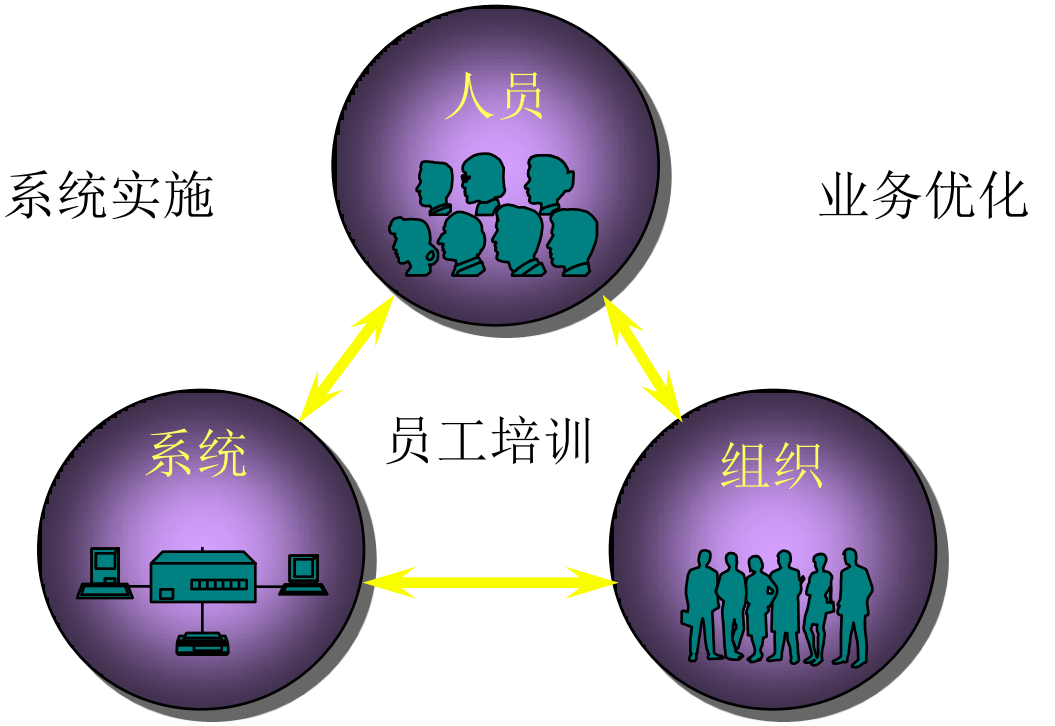


2 软件实施的内容

- 建立硬、软件环境，实现物理系统；
- 装载数据，系统试运行，进行局部调整；
- 用户技术培训和操作培训；
- 进行系统切换和交接；
- 制订系统管理和操作制度，正确运行系统；
- 维护系统，实现设计目标，发挥最大效益。



软件实施：牵动全局



3 实施方法论

- 业务诊断
- IT规划
- 方案

- 项目范围
- 项目组织
- 项目计划
- 项目启动

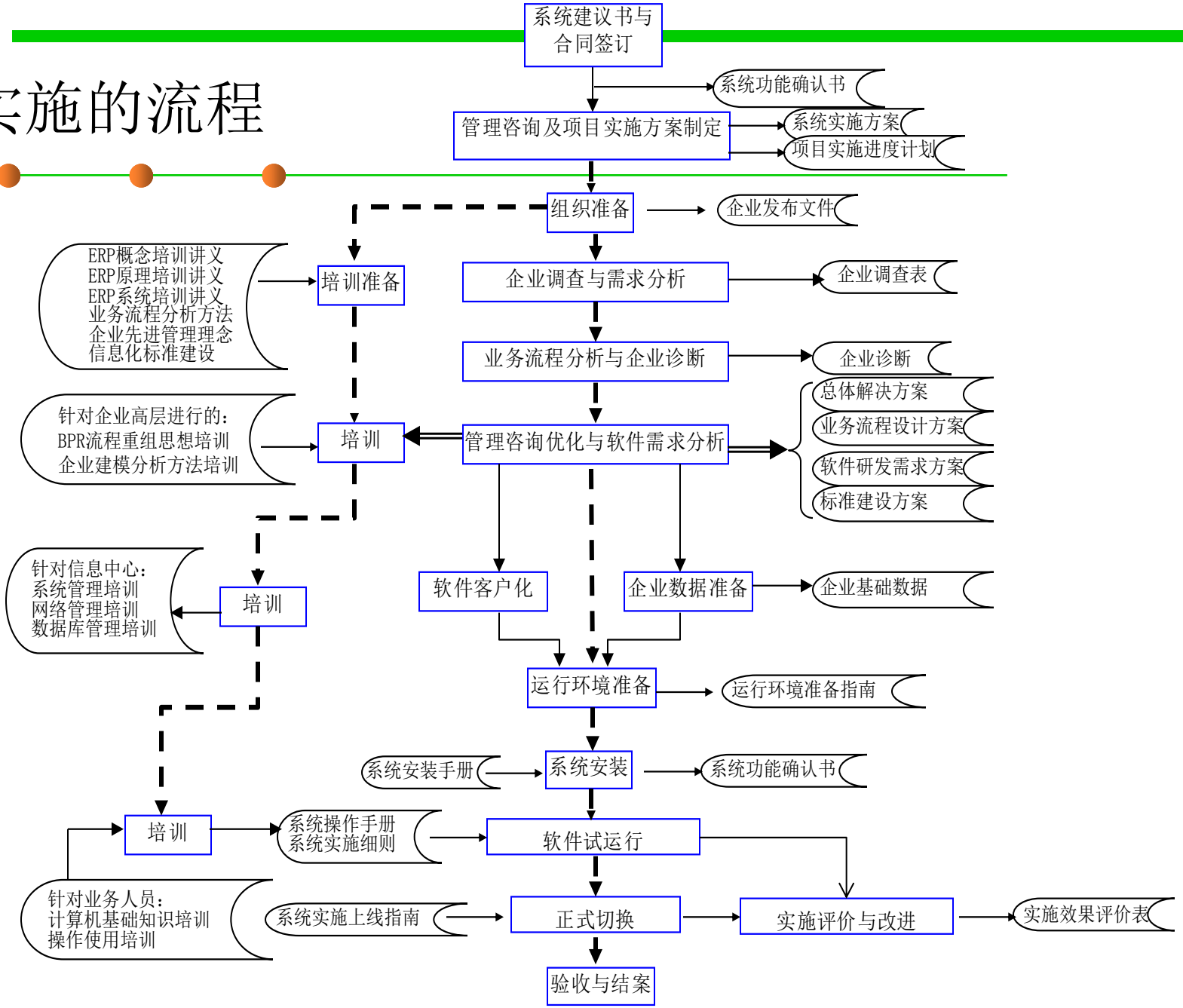
- 流程优化
- 培训
- 系统配置
- 系统测试
- 模拟运行
- 静态数据
- SOP
- 客户化手册

- 生产系统准备
- 系统权限
- 最终用户培训
- 动态数据
- 切换运行

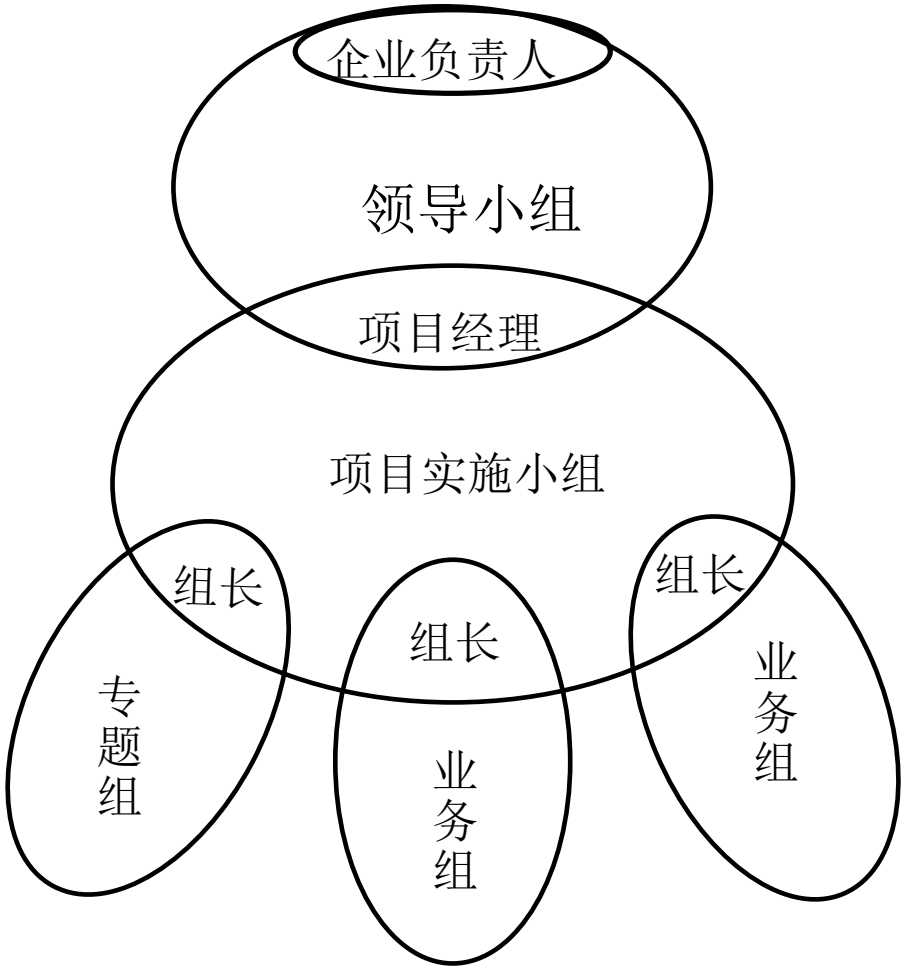
- 现场支持
- 系统验收
- 运行维护
- 周期运行与审查
- 业务优化与改善

培训/教育，项目管理，变革管理

4 软件实施的流程


系统建议书与合同签订
系统功能确认书
管理咨询及项目实施方案制定
系统实施方案
项目实施进度计划
组织准备
企业发布文件
ERP概念培训讲义
ERP原理培训讲义
ERP系统培训讲义
业务流程分析方法
企业先进管理理念
信息化标准建设
培训准备
企业调查与需求分析
企业调查表
业务流程分析与企业诊断
企业诊断
针对企业高层进行的：
BPR流程重组思想培训
企业建模分析方法培训
培训
管理咨询优化与软件需求分析
总体解决方案
业务流程设计方案
软件研发需求方案
标准建设方案
针对信息中心：
系统管理培训
网络管理培训
数据库管理培训
培训
软件客户化
企业数据准备
企业基础数据
运行环境准备
运行环境准备指南
系统安装手册
系统安装
系统功能确认书
培训
系统操作手册
系统实施细则
软件试运行
针对业务人员：
计算机基础知识培训
操作使用培训
系统实施上线指南
正式切换
实施评价与改进
实施效果评价表
验收与结案

关键步骤之一：实施组织与培训



关键步骤之二：数据准备

- **数据整理与规范化：**
  - 软件系统在通常需要特定的基础数据；
  - 软件的成功实施，依赖于客户是否能够准确、全面、规范化的提供软件所需的基础数据。
  - 软件是一个数据加工厂，没有高质量的数据原材料，不可能制造出高质量的功能与信息输出。
- **包括：**
  - 历史数据的整理
  - 数据资料的格式化
  - 各类基础数据的收集与电子化

ERP软件实施格言：三分技术、七分管理、十二分数据！

关键步骤之三：模拟运行

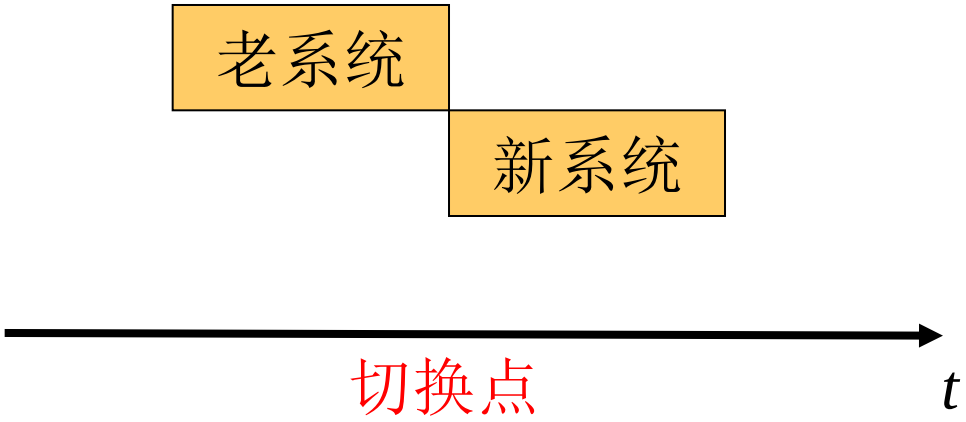
- **从客户的各主要业务数据中抽取一些典型数据，作为系统模拟运行阶段的测试数据**
  - 静态数据：在任何时间点都不变的数据，如客户信息；
  - 动态数据：随时间动态产生、动态处理和动态流转的数据，如订单、库存、计划等数据。

关键步骤之四：系统切换

- **系统转换**
  - 系统转换是由现行系统的工作方式向所开发的管理信息系统工作方式的转换过程，也是系统的设备、数据、人员等的转换过程。
- **系统转换的基本条件**
  - 系统设备：系统实施前购置、安装、调试完毕
  - 系统人员：系统转换前配齐并参与各管理岗位工作，并进行相关培训
  - 系统数据：系统转换所需各种数据按照要求各式输入到系统之中
  - 系统文件资料：用户手册、系统操作规程、系统结构与性能介绍手册

系统切换方法

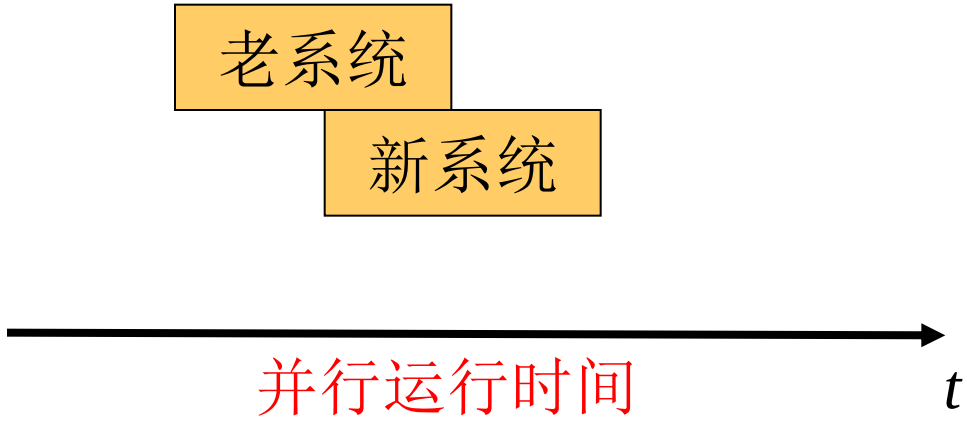
- **直接切换法**
  - 在某一确定的时刻，老系统停止运行， 新系统投入运行，新系统一般要经过较详细的测试和模拟运行。



- **并行切换法**
- **试点过渡法(分段转换)**

系统切换方法

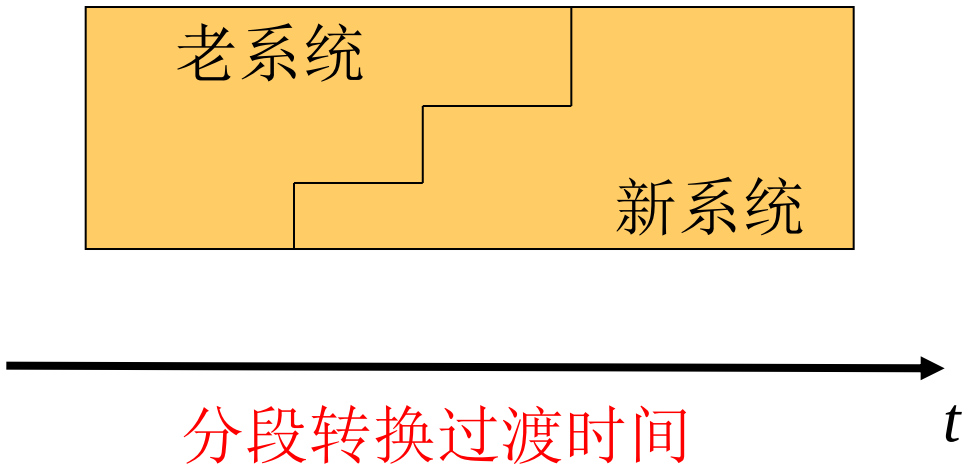
- **直接切换法**
- **并行切换法**
  - 新系统投入运行时，老系统并不停止运行，而是与新系统同时运行一段时间，对照两者的输出，利用老系统对新系统进行检验。



- **试点过渡法(分段转换)**

系统切换方法

- **直接切换法**
- **并行切换法**
- **试点过渡法(分段转换)**
  - 先选用新系统的某一部分代替老系统，作为试点，逐步的代替整个老系统。



[课堂讨论]

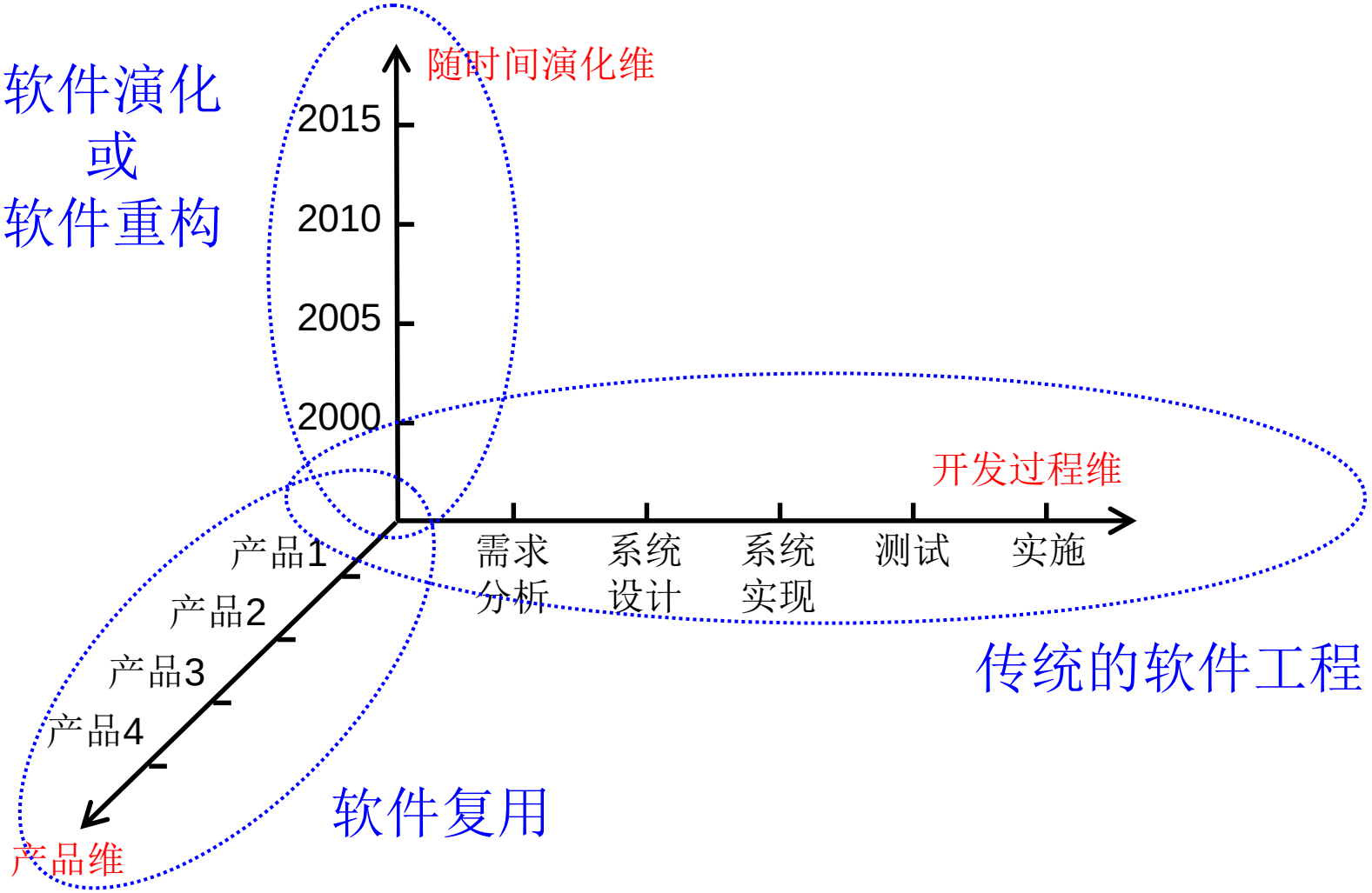
- 这三种系统切换方法各有哪些优缺点？

三种切换方法的对比

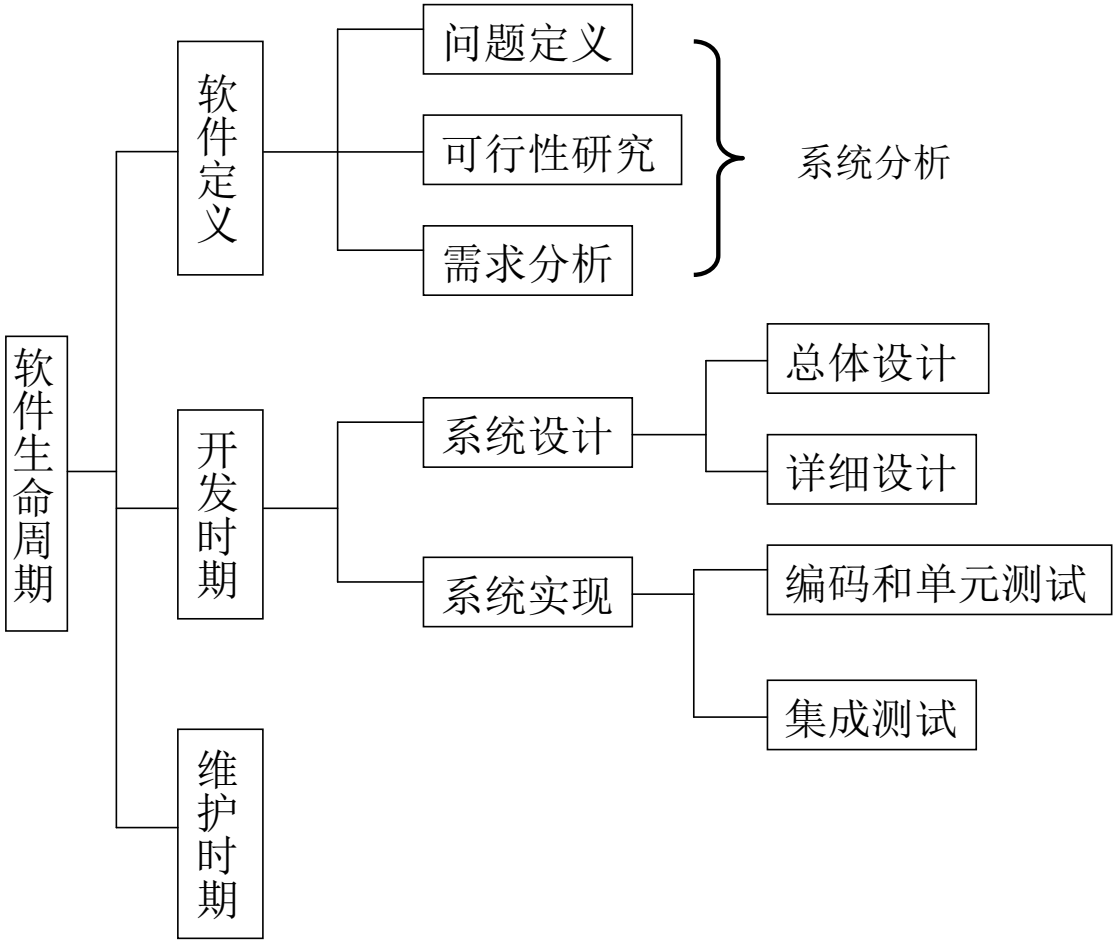
- **直接切换法**
  - 简单、费用低。
  - 风险大，应有一定的保护措施。
- **并行切换法**
  - 可保证系统的延续性，可进行新老系统的比较，平稳可靠的过渡。
  - 费用高，易延长系统转换的时间。
- **试点过渡法(分段转换)**
  - 避免了直接转换的风险，及并行转换的双倍费用，但会出现接口问题。
  - 适于大型系统，可保证平稳、可靠。

主要内容

软件工程的三个维度

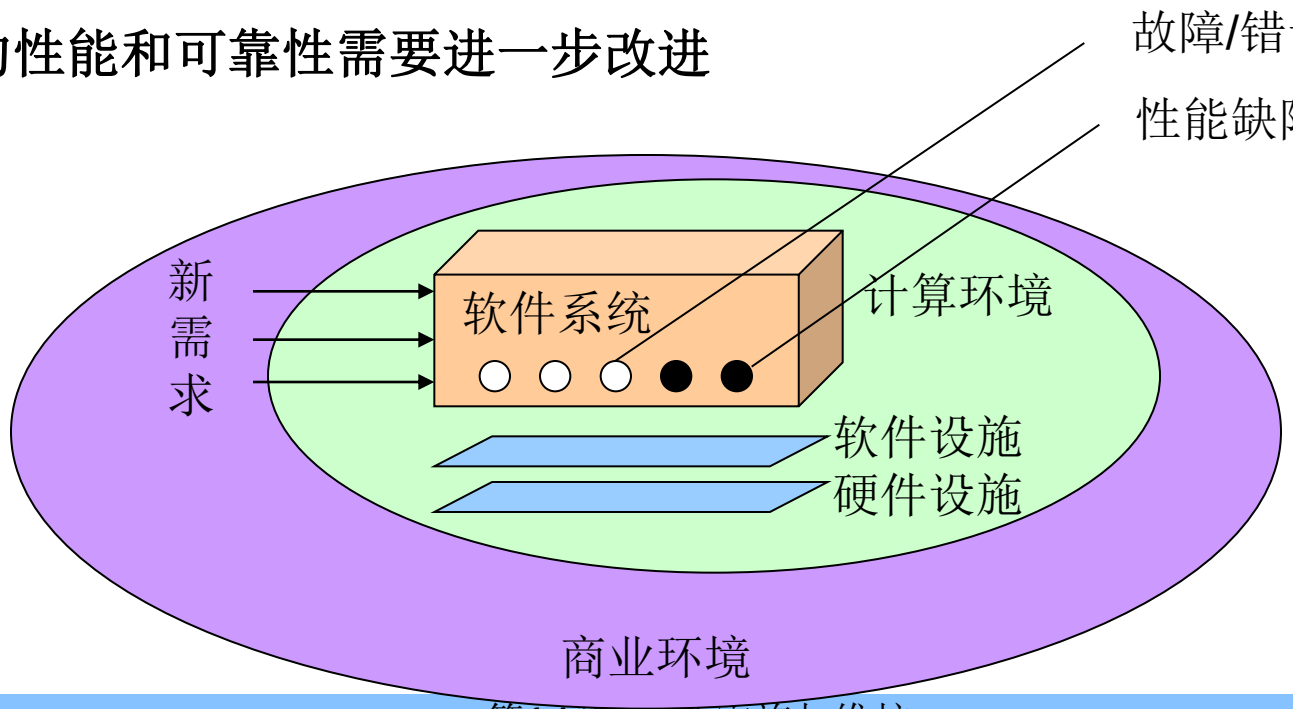

软件演化
或
软件重构
随时间演化维
2015
2010
2005
2000
开发过程维
需求分析
系统设计
系统实现
测试
实施
传统的软件工程
产品1
产品2
产品3
产品4
产品维
软件复用

软件生命周期


软件生命周期
软件定义
问题定义
可行性研究
需求分析
系统分析
开发时期
系统设计
总体设计
详细设计
系统实现
编码和单元测试
集成测试
维护时期

1 软件演化

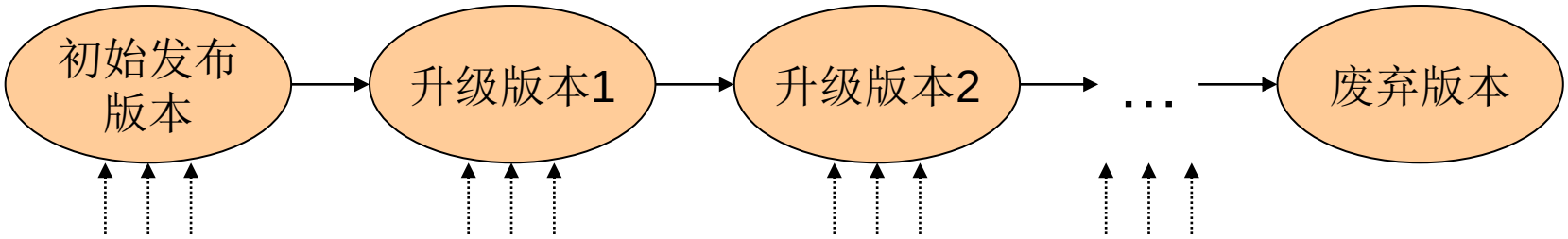
- 软件在使用过程中，新的需求不断出现
- 商业环境在不断地变化
- 软件中的缺陷需要进行修复
- 计算机硬件和软件环境的升级需要更新现有的系统
- 软件的性能和可靠性需要进一步改进



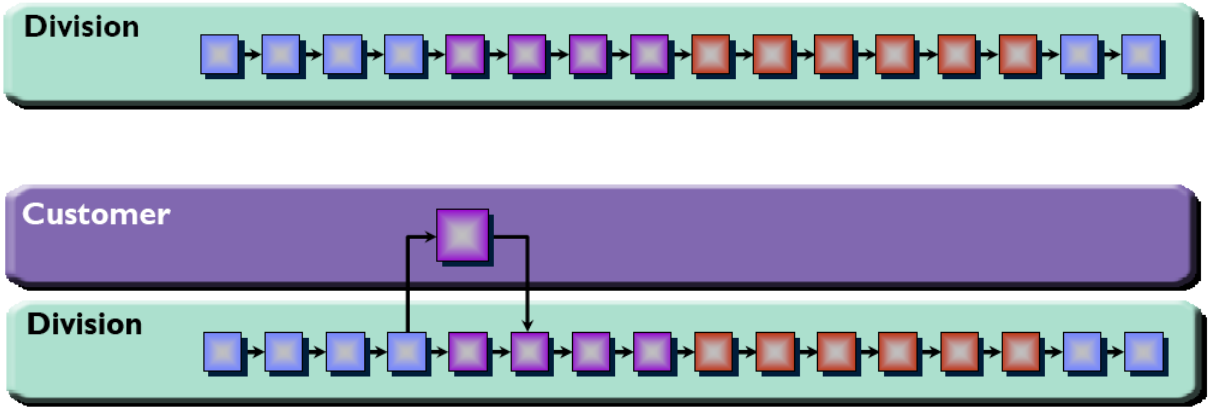
软件演化的Lehman定律

- **持续变化(continuing change)**
  - 现实世界的系统要么变得越来越没有价值，要么进行持续不断的变化以适应环境的变化；
  - 环境变化产生软件修改，软件修改又继续促进环境变化；
- **复杂度逐渐增大(increasing complexity)**
  - 当系统逐渐发生变化时，其结构和功能将变得越来越复杂，并逐渐难以维护并失去控制，直至无法继续演化，从而需要大量额外的资源和维护工作来保持系统的正常运行。
  - 软件修改会引入新的错误，造成故障率的升高；

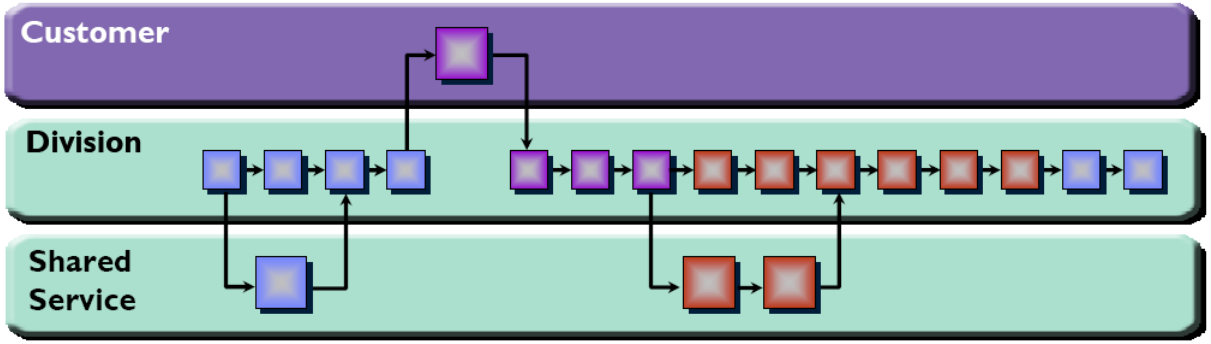
热力学第二定律(熵值理论)



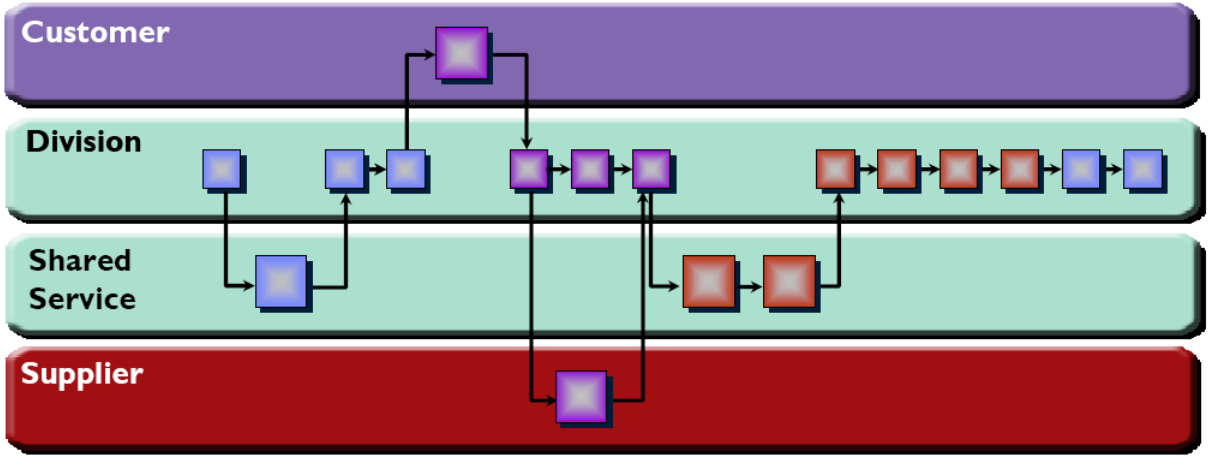
软件演化的典型例子



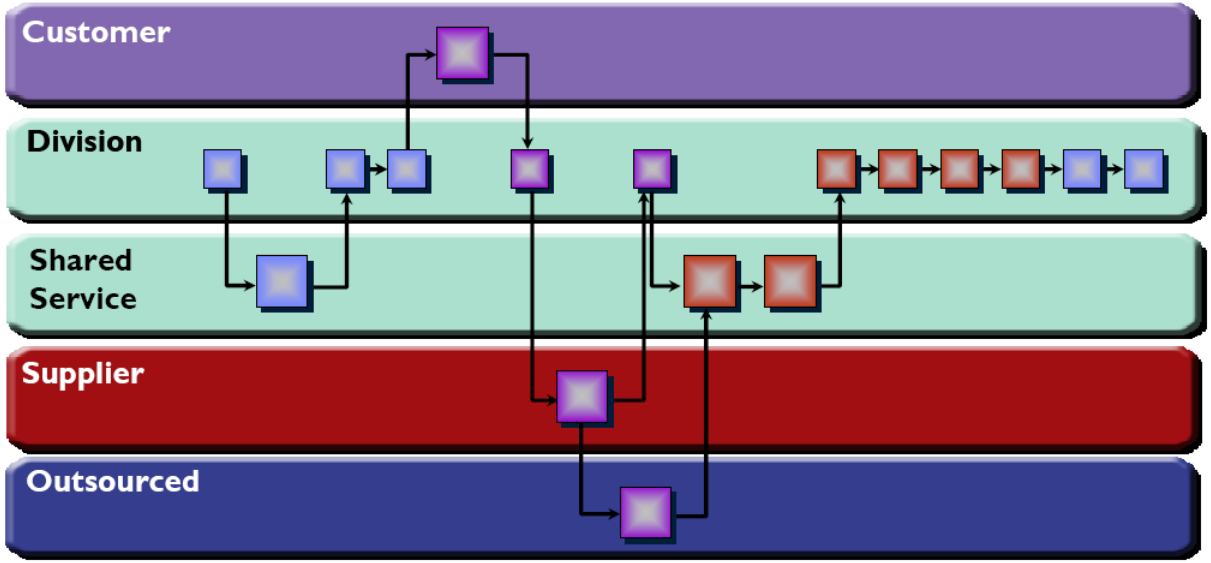
变化1：客户订单入口



变化2：提取公共服务



变化3：供应商管理库存



变化4：物流业务外包出去

银行业的困境

- 目前不少银行/保险公司所使用的核心业务处理系统甚至仍在使用1980年代所开发的COBOL语言书写的程序，用的还是VSAM文件系统。

- 随着银行/保险行业标准和会计准则的更新和新产品的推出，这些原有的系统已经无法支持这些新变化。

- 但由于现在已经无法找到能够完全理解这些核心系统的程序人员(例如COBOL早已很少使用)，所以不少银行/保险公司往往受困于此却无计可施。

2 软件演化的处理策略

- **软件维护(Software Maintenance)**
  - 为了修改软件缺陷或增加新的功能而对软件进行的变更
  - 软件变更通常发生在局部，不会改变整个结构

- **软件再工程(Software Re-engineering)**
  - 为了避免软件退化而对软件的一部分进行重新设计、编码和测试，提高软件的可维护性和可靠性等

- **前者比后者的力度要小。**

主要内容

14.1 软件实施

14.2 软件维护

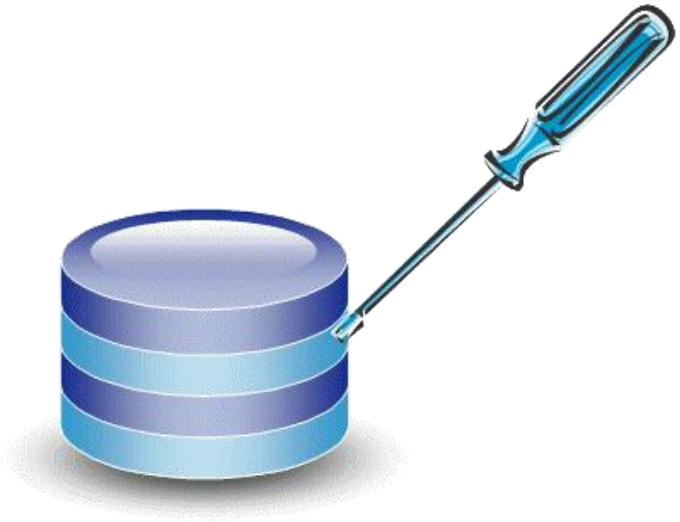
14.2.1 软件演化

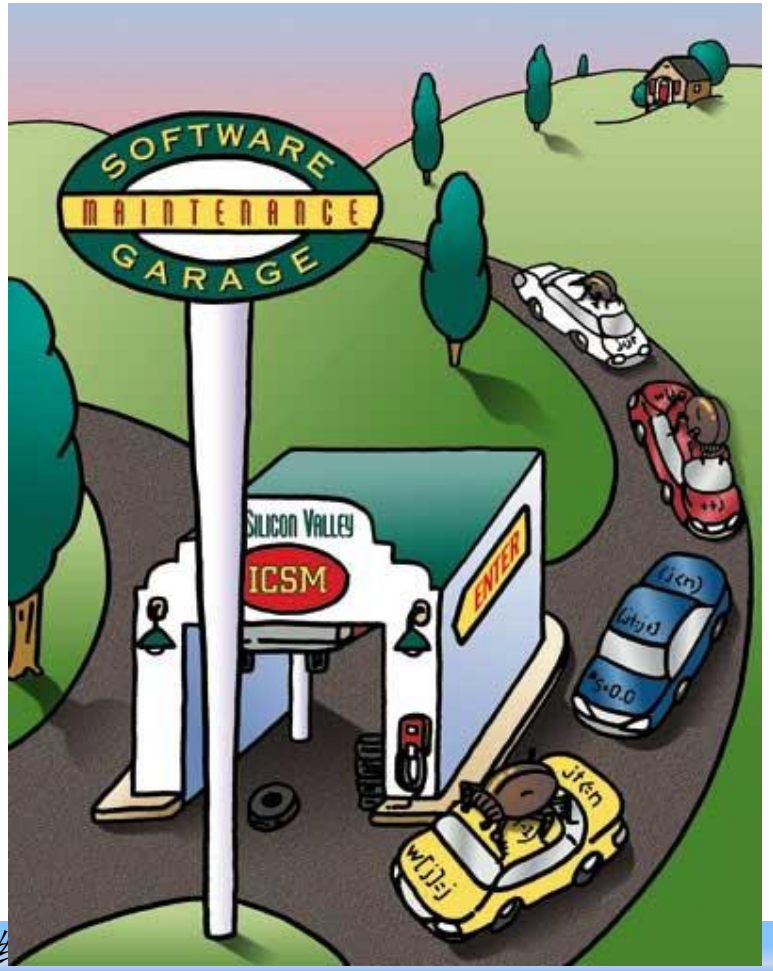
14.2.2 软件维护

14.3* 软件再工程

1 软件维护

- **软件维护**
  - (ANSI/IEEE) 在软件产品发行和被投入运行使用之后对其的修改，以改正错误，改善性能或其他属性，从而使产品适应新的环境或新的需求。





2 软件维护的类型

- 纠错性维护：修改软件中的缺陷或不足
- 适应性维护：修改软件使其适应不同的操作环境，包括硬件变化、操作系统变化或者其他支持软件变化等
- 完善性维护：增加或修改系统的功能，使其适应业务的变化
- 预防性维护：为减少或避免以后可能需要的前三类维护而提前对软件进行的修改工作

纠错性维护

- **纠错性维护(Corrective Maintenance)：**
  - 在软件交付使用后，因开发时测试的不彻底、不完全，必然会有部分隐藏的错误遗留到运行阶段。
  - 这些隐藏下来的错误在某些特定的使用环境下就会暴露出来。
  - 为了识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误使用，应当进行的诊断和改正错误的过程就叫做改正性维护。

适应性维护

- **在使用过程中，**
  - 外部环境：新的硬、软件配置
  - 数据环境：数据库、数据格式、数据输入/输出方式、数据存储介质

  可能发生变化。

- **为使软件适应这种变化，而去修改软件的过程就叫做适应性维护(Adaptive Maintenance)。**

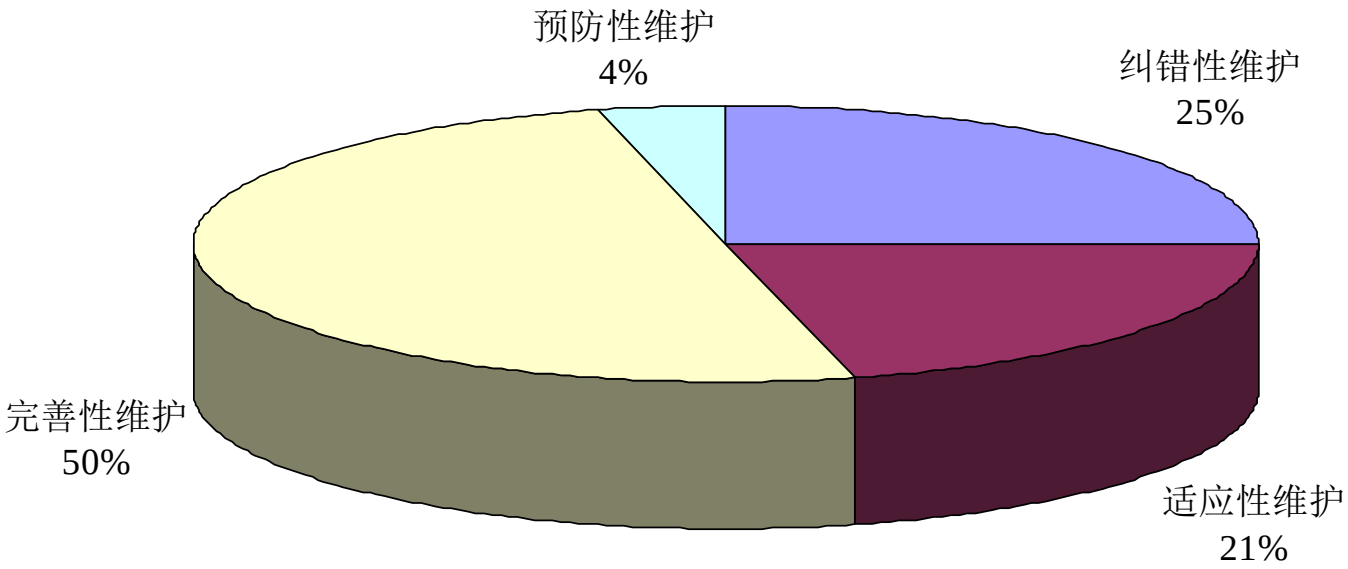
完善性维护

- **完善性维护(Perfective Maintenance)**
  - 在软件的使用过程中，用户往往会对软件提出新的功能与性能要求。
  - 为了满足这些要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。
  - 这种情况下进行的维护活动叫做完善性维护。

预防性维护

- **预防性维护(Preventive Maintenance)：为了提高软件的可维护性、可靠性等，为以后进一步改进软件打下良好基础。**
  - 定义为“采用先进的软件工程方法对需要维护的软件或软件中的某一部分(重新)进行设计、编制和测试”。

软件维护的类型



小结

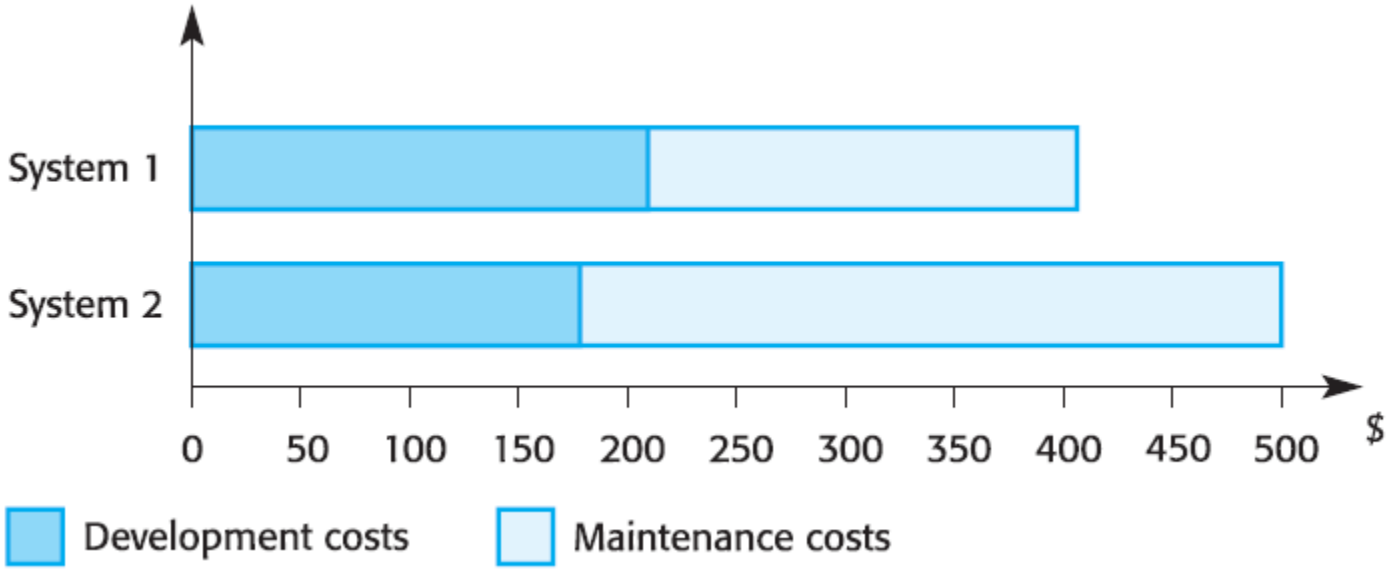
- **实践表明，在几种维护活动中，完善性维护所占的比重最大，即大部分维护工作是改变和加强软件，而不是纠错。**
  - 完善性维护不一定是救火式的紧急维修，而可以是有计划、有预谋的一种再开发活动。
  - 来自用户要求扩充、加强软件功能、性能的维护活动约占整个维护工作的50%。
- **软件维护活动所花费的工作占整个生存期工作量的70%以上，这是由于在漫长的软件运行过程中需要不断对软件进行修改，以改正新发现的错误、适应新的环境和用户新的要求，这些修改需要花费很多精力和时间，而且有时会引入新的错误。**

3 软件维护的内容

- **程序维护**
  - 根据使用的要求，对程序进行全部或部分修改。修改以后，必须书写修改设计报告。
- **数据维护**
  - 数据维护指对数据有较大的变动。如安装与转换新的数据库；或者某些数据文件或数据库出现异常时的维护工作，如文件的容量太大而出现数据溢出等。
- **硬件维护**
  - 硬件人员应加强设备的保养以及定期检修，并做好检验记录和故障登记工作。

4 软件维护的成本

- **软件的维护成本极其昂贵**
  - 业务应用系统：维护费用与开发成本大体相同
  - 嵌入式实时系统：维护费用是开发成本的四倍以上



影响维护成本的因素

- **团队稳定性**
  - 系统移交后开发团队会解散，人员分配到其他项目中，负责维护的人员通常不是原开发人员，需要花时间理解系统。
- **合同责任**
  - 维护合同一般独立于开发合同，这样开发人员有可能缺少为方便维护而写软件的动力。
- **人员技术水平**
  - 维护人员有可能缺乏经验，而且不熟悉应用领域。
- **程序年龄与结构**
  - 程序结构随年龄的增加而受到破坏，不易理解和变更。

软件维护的典型困难

- **软件维护中出现的大部分问题都可归咎于软件规划和开发方法的缺陷：**
  - 软件开发时采用急功近利还是放眼未来的态度，对软件维护影响极大，软件开发若不严格遵循软件开发标准，维护就会遇到许多困难。

- **例如：**

1. 读懂原开发人员写的程序通常相当困难
2. 软件人员的流动性，使得软件维护时，很难与原开发人员沟通
3. 没有文档或文档严重不足
4. 软件设计时，欠考虑软件的可修改性
5. 频繁的软件升级，要追踪软件的演化变得很困难，使软件难以修改

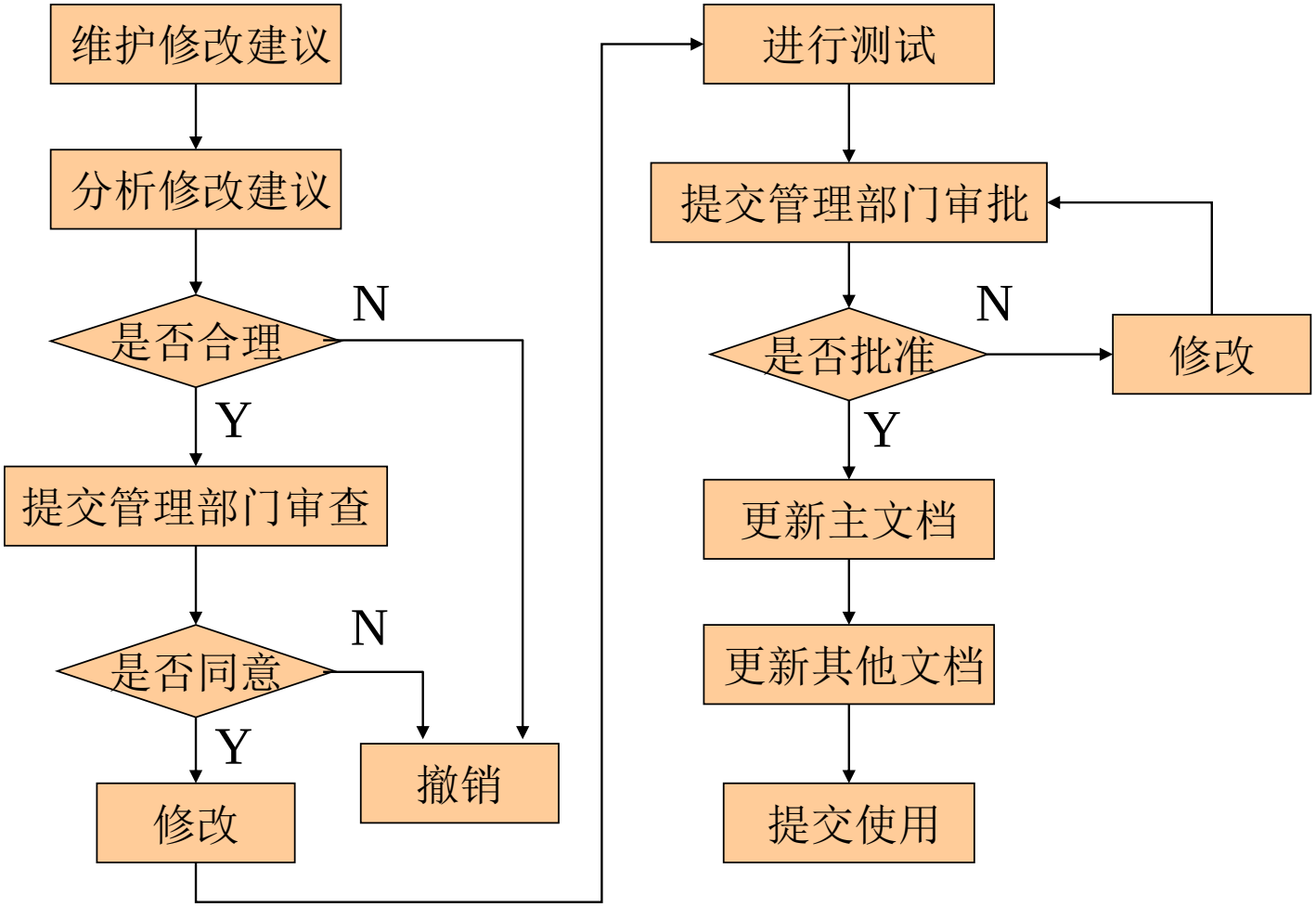
造成困难的根本原因

- 很难甚至不可能追踪软件版本的进化过程，软件的变化没在相应文档中反映出来；
- 很难甚至不可能追踪软件的整个创建过程；
- 理解他人的程序非常困难，当软件配置不全、仅有源代码时问题尤为严重；
- 软件人员流动性很大，维护他人软件时很难得到开发者的帮助；
- 软件没有文档、或文档不全、或文档不易理解、或与源代码不一致；
- 多数软件设计未考虑修改的需要(有些设计方法采用了功能独立和对象类型等一些便于修改的概念)，软件修改不仅困难而且容易出错；
- 软件维护不是一项有吸引力的工作，从事这项工作令人缺乏成就感。

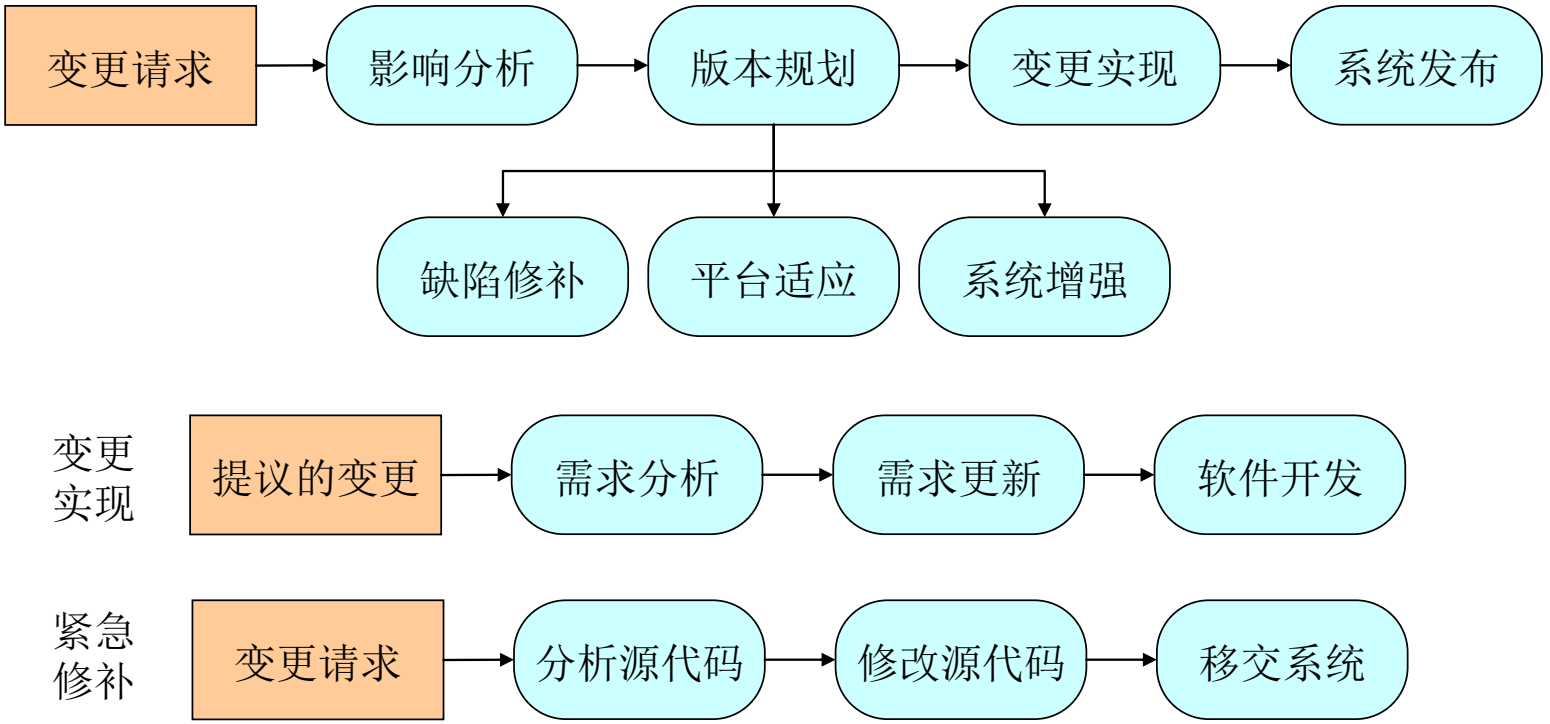
不好的维护所造成的代价

- 因为可用的资源必须供维护任务使用，以致耽误甚至丧失了开发的良机；
- 当看来合理的有关改错或修改的要求不能及时满足时将引起用户不满；
- 由于维护时的改动，在软件中引入了潜伏的故障，从而降低了软件的质量；
- 当必须把软件工程师调去从事维护工作时，将在开发过程中造成混乱；
- 生产率的大幅度下降。

5 软件维护的管理流程

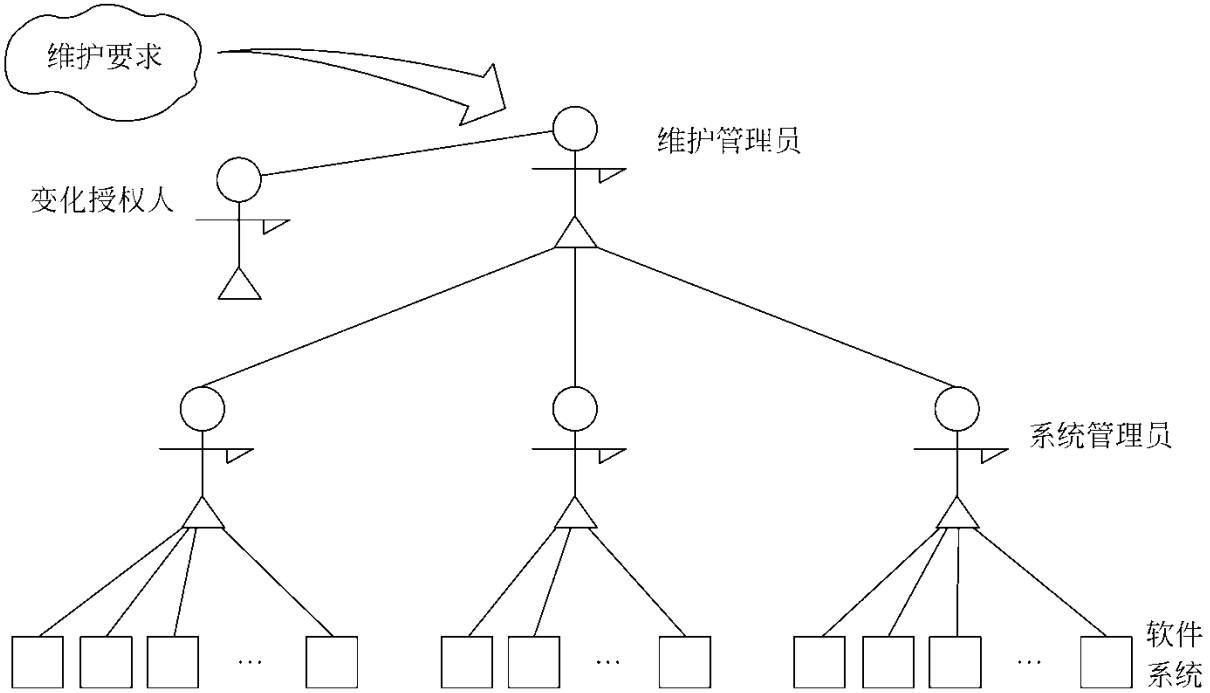

维护修改建议
分析修改建议
是否合理
N
Y
提交管理部门审查
是否同意
N
Y
撤销
修改
进行测试
提交管理部门审批
是否批准
N
Y
修改
更新主文档
更新其他文档
提交使用

软件维护过程


变更请求
影响分析
版本规划
变更实现
系统发布
缺陷修补
平台适应
系统增强
变更实现
提议的变更
需求分析
需求更新
软件开发
紧急修补
变更请求
分析源代码
修改源代码
移交系统

建立维护组织

- 现实：一般软件公司没有专门的维护机构。
- 维护机构成员一般包括：配置管理员、维护控制员、系统管理员、一般维护工作人员。



安排计划

- 维护工作不应该采用“一次改一个错”的零打碎敲的方法，而应当有计划有步骤地统筹安排。

- 维护报告应包括的内容：该维护任务的范围，所需资源，确认的要求，维修费用及维修进度安排。

维护实施

- **软件维护任务与新软件开发的过程基本上一致并且是并行的。**
- **软件修改完成后，由维修主管进行验收，验收标准如下：**
  - 全部软件文档已准备齐全，并已更新好
  - 所有测试用例和测试结果已经正确记录下来
  - 记录和所有寻找软件配置的工序已建立
  - 维护工序和责任已经确定

6 软件维护文档

- **软件问题报告**
  - 由要求一项维护活动的用户填写，用标准化的格式表达所有软件维护要求
  - 如果遇到了一个错误，那么必须完整描述导致出现错误的环境(包括输入数据，全部输出数据，以及其他有关信息)
  - 对于适应性或完善性的维护要求，应该提出一个简短的需求说明书
- **软件变动报告**
  - 满足维护要求表中提出的要求所需要的工作量
  - 维护要求的性质
  - 这项要求的优先次序
  - 与修改有关的事后数据

软件维护文档

- **软件维护记录**

Swanson提出的项目表

(1)程序名称；
(2)源程序语句条数；
(3)机器代码指令条数；
(4)使用的程序设计语言；
(5)程序的安装日期；
(6)程序安装后的运行次数；
(7)与程序安装后运行次数有关的处理故障的次数；
(8)程序修改的层次和名称；
(9)由于程序修改而增加的源程序语句条数；
(10)由于程序修改而删除的源程序语句条数；
(11)每项修改所付出的“人时”数；
(12)程序修改的日期；
(13)软件维护人员的姓名；
(14)维护申请报告的名称；
(15)维护类型；
(16)维护开始时间和维护结束时间；
(17)用于维护的累计“人时”数；
(18)维护工作的净收益。

软件维护的副作用

- 由于软件被修改而导致的错误或其他多余动作的发生，称为软件维护的副作用。
- 为确保编码修改没有引入新的错误，应进行严格的回归测试。一般情况下，通过回归测试，可以发现并纠正修改编码所带来的副作用。

7 软件的可维护性

- **软件的可维护性**
  - 软件能够被理解、改正、适应和完善以适应新的环境的难易程度。
- **控制因素**
  - 与开发方法有关的因素，如采用什么方法
  - 与开发环境有关的因素
    1. 合格的软件开发人员
    2. 可理解的系统结构
    3. 系统处理容易
    4. 使用标准的编程语言
    5. 使用标准的操作系统
    6. 标准化的文档结构
    7. 测试用例的有效性
    8. 系统自身拥有的纠错工具
    9. 易于维护的计算机
    10. 开发该软件的个人或组织

提高软件可维护性的方法

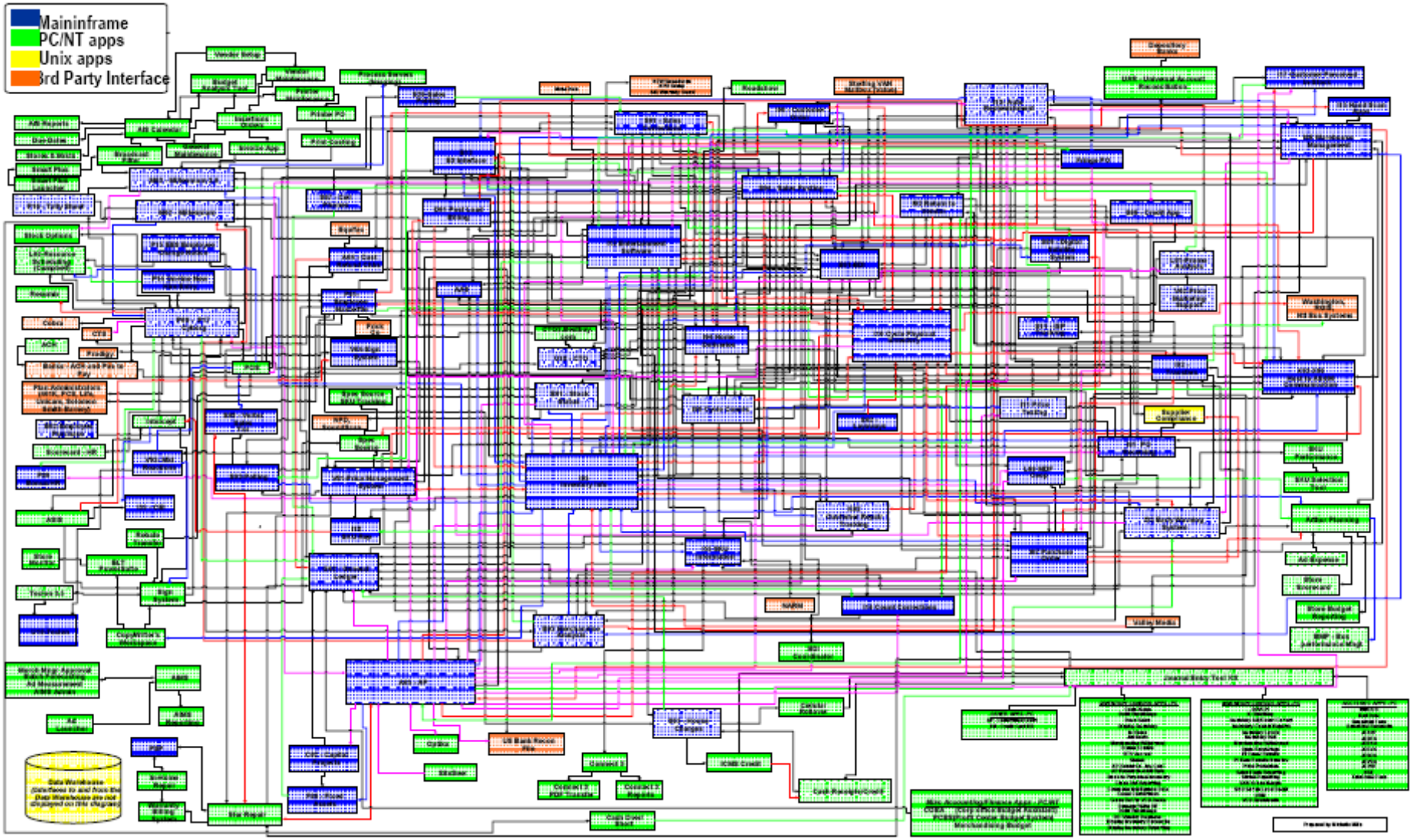
- 建立明确的软件质量标准
- 利用先进的软件技术和工具
- 建立明确的质量保证制度
- 选择可维护的程序设计语言
- 改进软件的文档

8 遗留系统

- **遗留系统(legacy system)**
  - “已经运行了很长时间的、对用户来说很重要的、但是目前已无法完全满足要求却不知道如何处理的软件系统”。

- **特点**
  - 现有维护人员没有参与开发
  - 不具备现有的开发规范
  - 文档不完整，修改记录简略

遗留系统



遗留系统

- **更换遗留系统(Legacy System)是有风险的**
  - 遗留系统几乎没有完整的描述
  - 业务过程和遗留系统的操作方式紧密地“交织”在一起
  - 重要的业务规则隐藏在软件内部
  - 开发新软件本身是有风险的
- **变更遗留系统的问题**
  - 系统的不同部分是由不同的团队实现的
  - 系统的部分或全部是用一种过时不用的语言编写
  - 文档不充分或过时
  - 经过多年维护，系统结构可能已经破坏，理解设计难度大

如何维护遗留系统？

- **[课堂讨论]你有什么好的想法？**

- **一种可行的解决方案：软件再工程**
  - 采用先进的软件工程方法对整个软件或软件中的一部分重新进行设计、编写和测试，以提高软件的可维护性和可靠性，保证系统的正常运行。
  - “把今天的SE方法学用于昨天的系统以满足明天的需要”

主要内容

14.1 软件实施

14.2 软件维护

14.2.1 软件演化

14.2.2 软件维护

14.3* 软件再工程

重建一所房子…

- **你购买了一所二手房，但你不满意目前房子的装修情况…**
- **为此，你考虑重建这所房子 。**
- **怎么做？**
  - 根据一组标准，对房子进行检查
  - 确认其结构是否良好
  - 了解房子最初是如何建造的
  - 采用何种材料进行重建
  - 采用何种方式进行重建

软件再工程

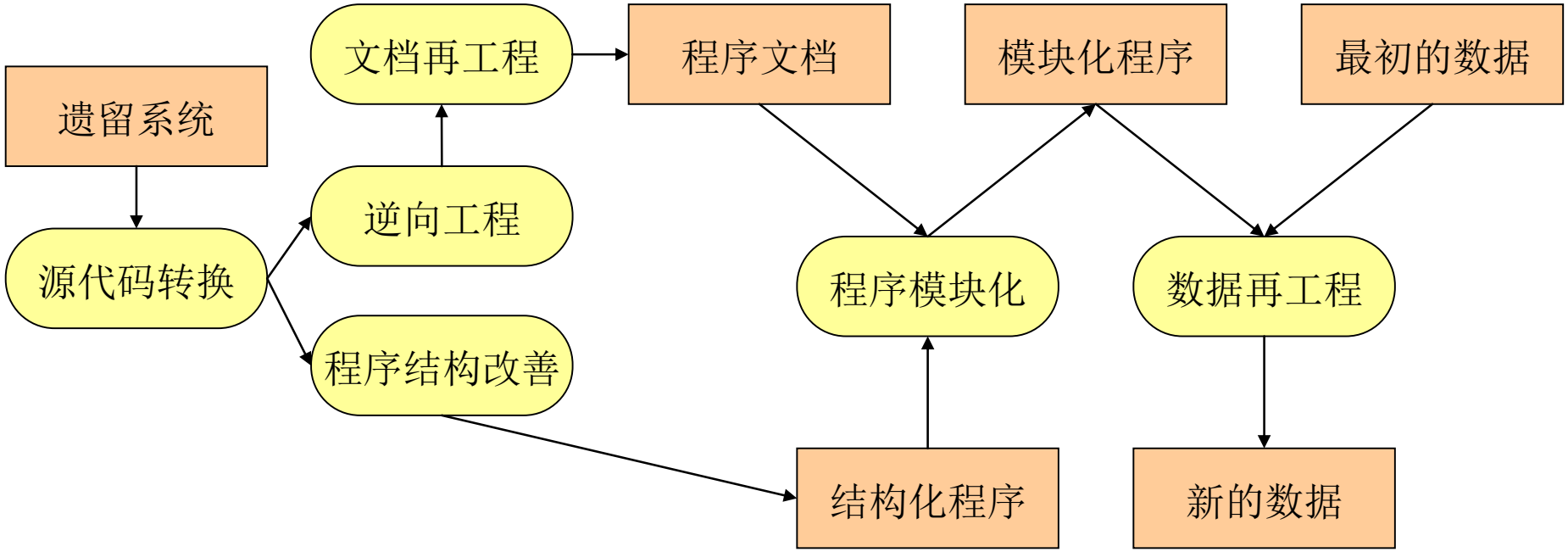
- **软件再工程(Software Re-engineering)**
  - 重新构造或编写现有系统的一部分或全部，但不改变其功能
  - 在大型系统中某些部分需要频繁维护时，可应用软件再工程
  - 目的：努力使系统更加易于维护，系统需要被再构造和再文档化
- **优势**
  - 减少风险：重新开发一个在用的系统具有很高的风险，可能会有开发问题、人员问题和规格说明问题
  - 降低成本：再工程的成本比重新开发软件的成本要小得多

正向工程与再工程

正向工程

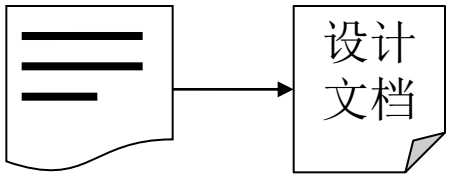
软件再工程

软件再工程的过程

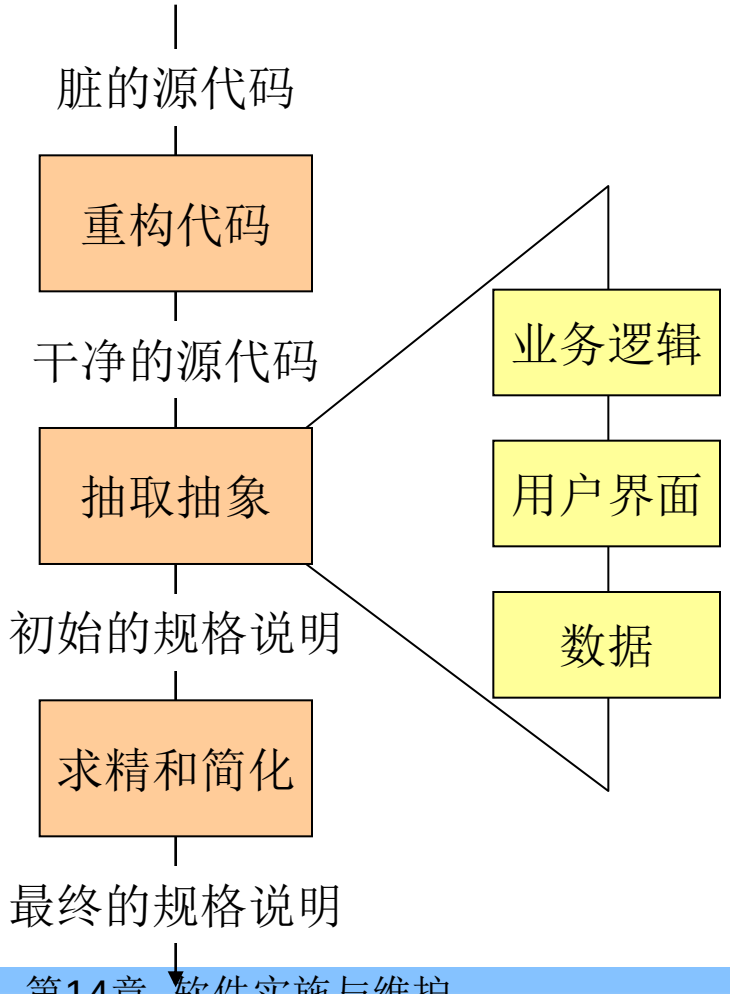

遗留系统
源代码转换
逆向工程
文档再工程
程序结构改善
程序文档
结构化程序
程序模块化
模块化程序
数据再工程
最初的数据
新的数据

再工程的基本活动

- **源代码转换(source code transformation)**
  - 代码从原有的程序设计语言转换到一种新语言
- **逆向工程(reverse engineering)**
  - 分析程序并抽取信息记录其结构和功能
- **程序结构改善(program restructuring)**
  - 分析和修改程序的控制结构，使其更易读和好理解
- **程序模块化(program modulization)**
  - 重新组织程序的结构
- **数据再工程(data restructuring)**
  - 改变程序处理的数据以反映程序的变更(文件存储→数据库存储，数据格式改变，等等)



逆向工程



结束

2011年6月6日