软件过程

- 软件系统的规格描述、设计、实现和测试的互相连贯的一系列活动

目标

- 介绍软件过程模型
- 描述不同的过程模型，什么时候应用
- 描述需求工程、软件开发、测试和进化的过程模型的概貌
- 介绍支持软件过程的CASE技术

内容

- 软件过程模型
- 过程反复
- 软件描述
- 软件设计和实现
- 软件有效性验证
- 软件进化
- 自动化的过程支持

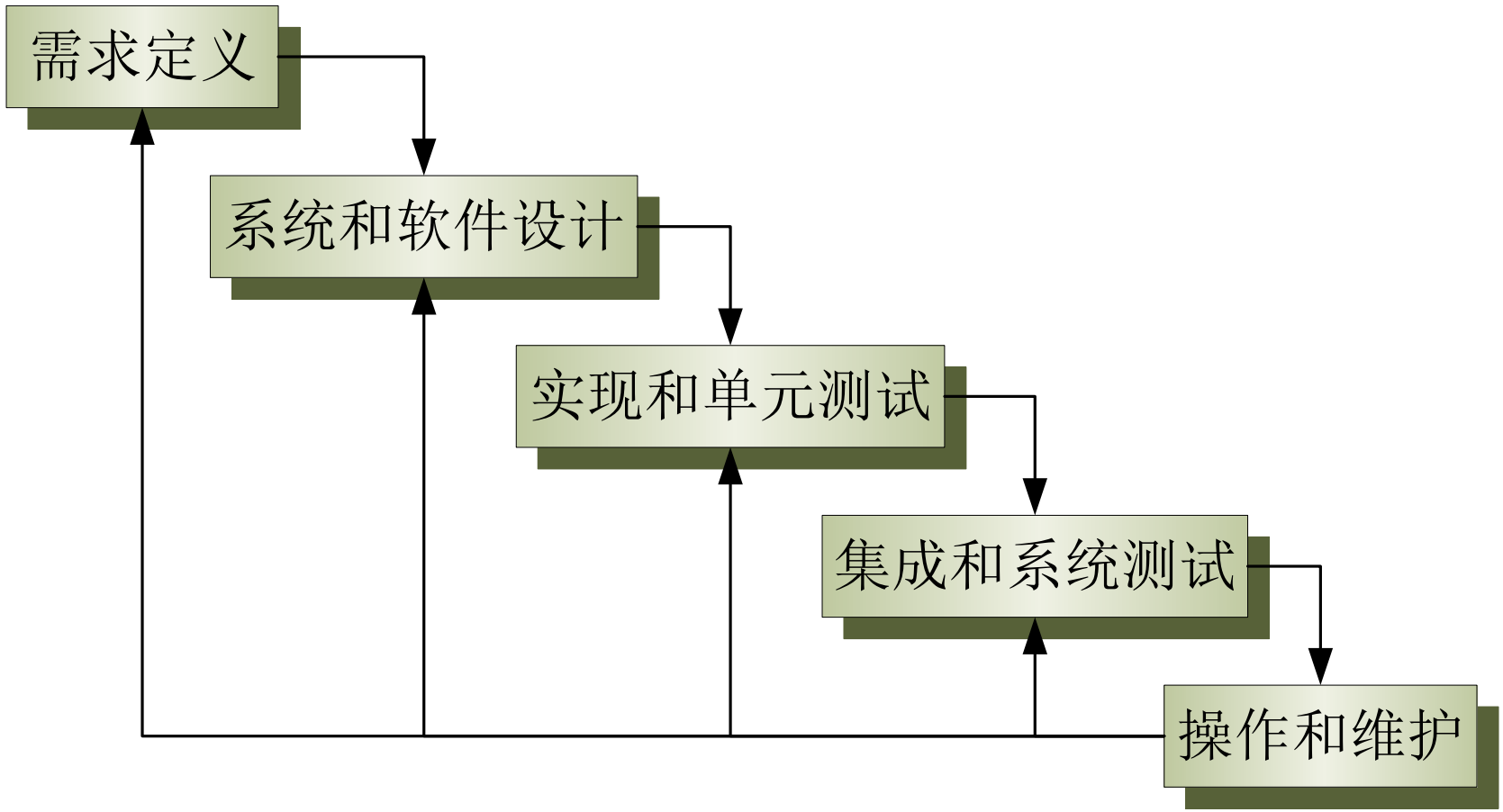
软件过程

- 开发软件系统需要一系列有组织的活动
  - 描述
  - 设计
  - 有效性验证
  - 进化
- 一个软件过程模型是过程的一个抽象表达。它表示从一些特定的视角对过程的描述。

通用软件过程模型

- 瀑布模型
  - 独立的过程阶段
- 进化式开发
  - 描述和开发交叉进行
- 形式化系统开发
  - 数学系统模型形式化的转化为一个系统实现
- 面向复用的开发
  - 从已有的组件组建成系统

瀑布模型



瀑布模型的阶段

- 需求分析和定义
- 系统和软件设计
- 实现和单元测试
- 集成和系统测试
- 操作和维护
- 瀑布模型的缺点是过程进行中对应变更比较困难。 需求变更、设计变更、测试变更

瀑布模型问题

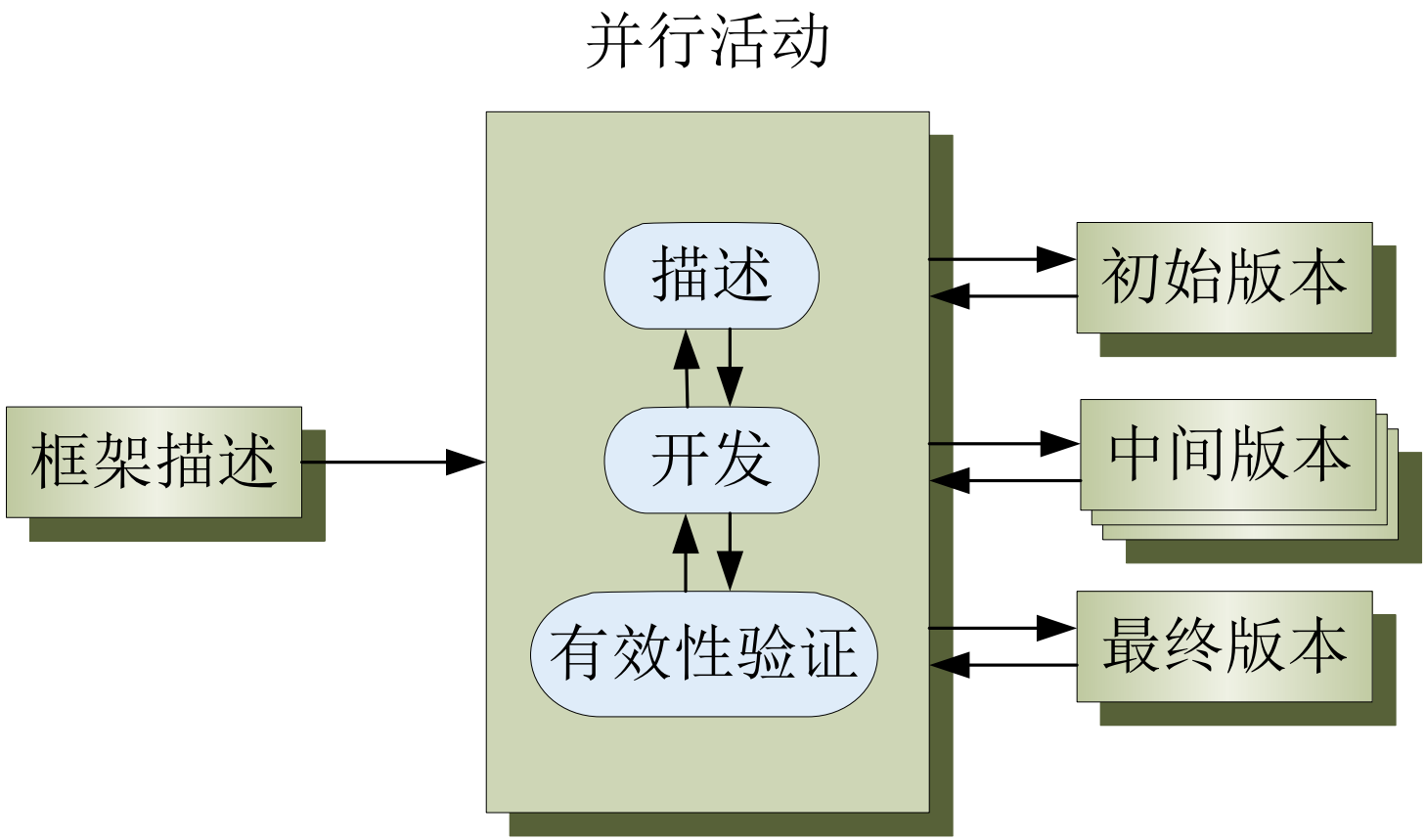
- 将项目生硬地划分成几个明显不同的阶段
- 这使得响应客户的需求比较困难
- 所以这个模型适用于需求能够较好地理解的情况

进化式开发

- 探索式开发
  - 其目标是与客户一起工作，从最初的需求草案进化到最后的系统。应该从理解最清楚的需求开始。
- 抛弃式原型
  - 目的是理解系统需求。从理解较差的需求开始。

用户界面适合于抛弃式原型开发

进化式开发



进化式开发

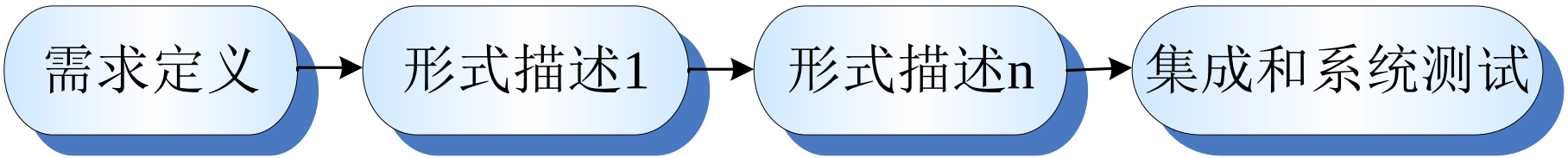
- 问题
  - 缺乏过程可见性
  - 系统结构通常较差
  - 需要特殊的工具和技术 (例如采用快速建立原型的语言)
- 适用性
  - 中小型交互式系统
  - 大型系统的一部分 (如用户界面)
  - 生命周期短的系统

形式化系统开发

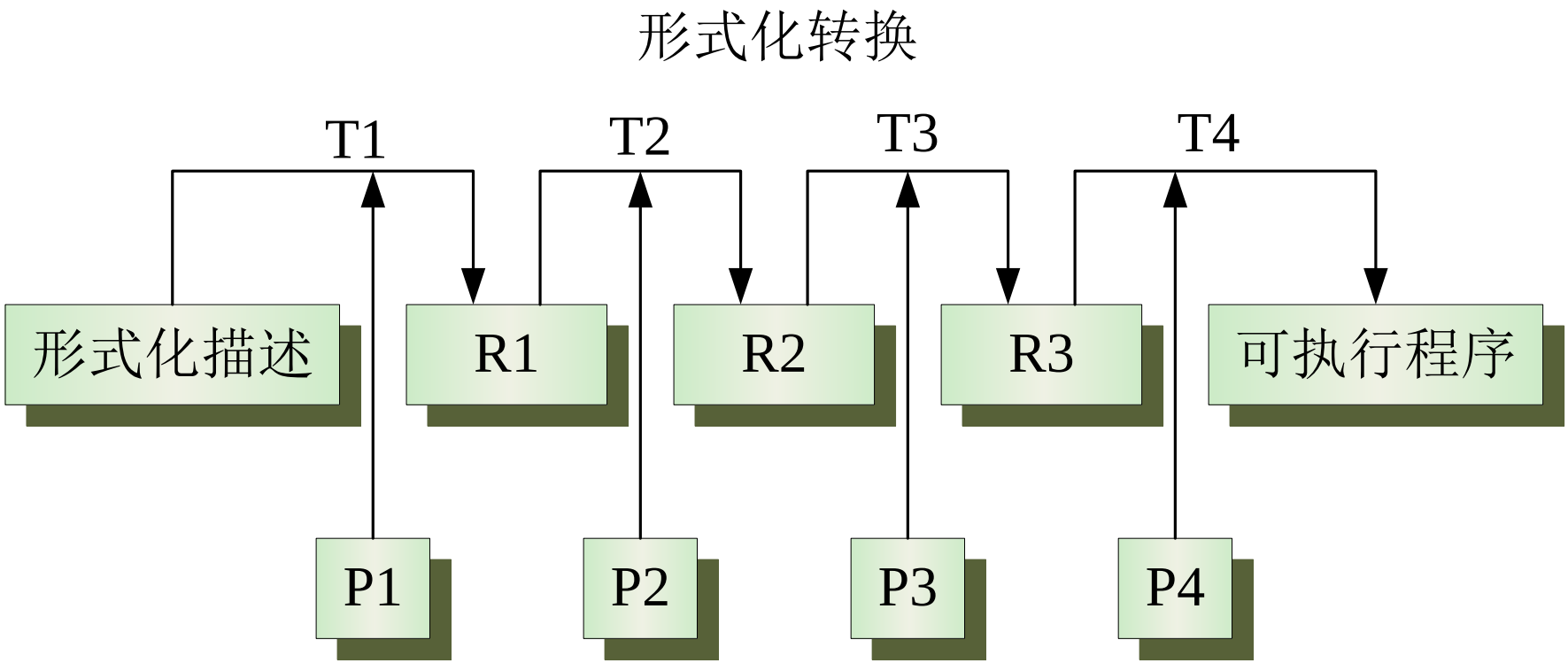
形式化编程语言：B语言、Z语言

- 基于用形式化数学转换将系统描述转换成一个可执行程序
- 转换是保证正确性的，所以程序是符合描述的。
- 具体实现如IBM的净室(Cleanroom)过程软件开发

形式化系统开发



形式化转换



形式化转换的正确证明

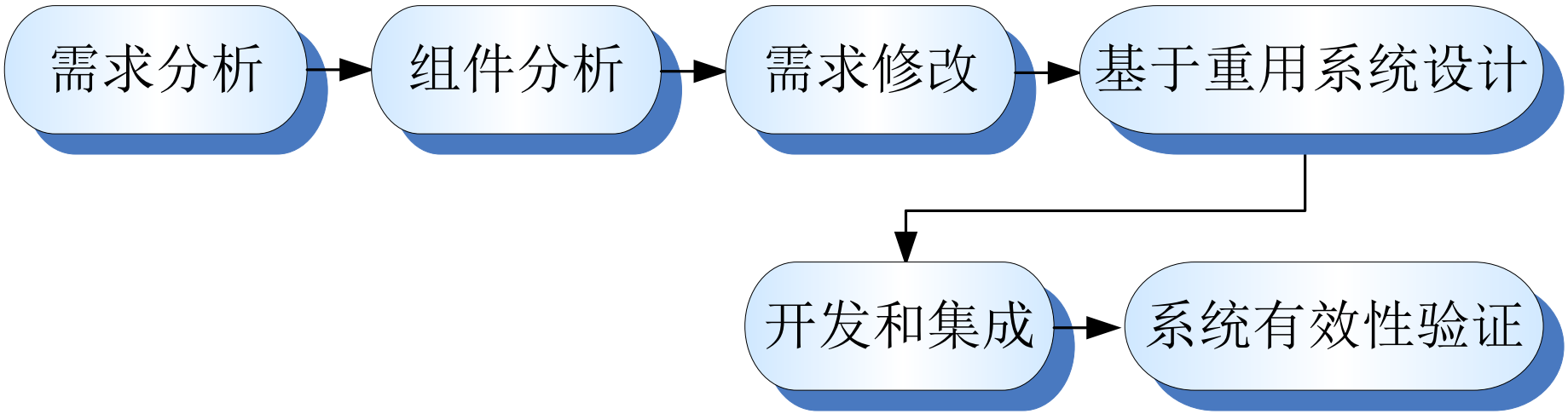
形式化系统开发

- 问题
  - 应用这项技术需要专门的技能和训练
  - 系统的某些方面难以形式化描述，例如用户界面
- 适应性
  - 适合于严格的系统，特别是安全性和保密性要求极高的系统

面向复用的开发

- 基于已有的组件或者COTS系统的系统重用
- 过程阶段
  - 组件分析
  - 需求修正
  - 基于重用的系统设计
  - 开发和集成
- 这种方法正变得越来越重要，但经验还是不足。

面向复用的开发



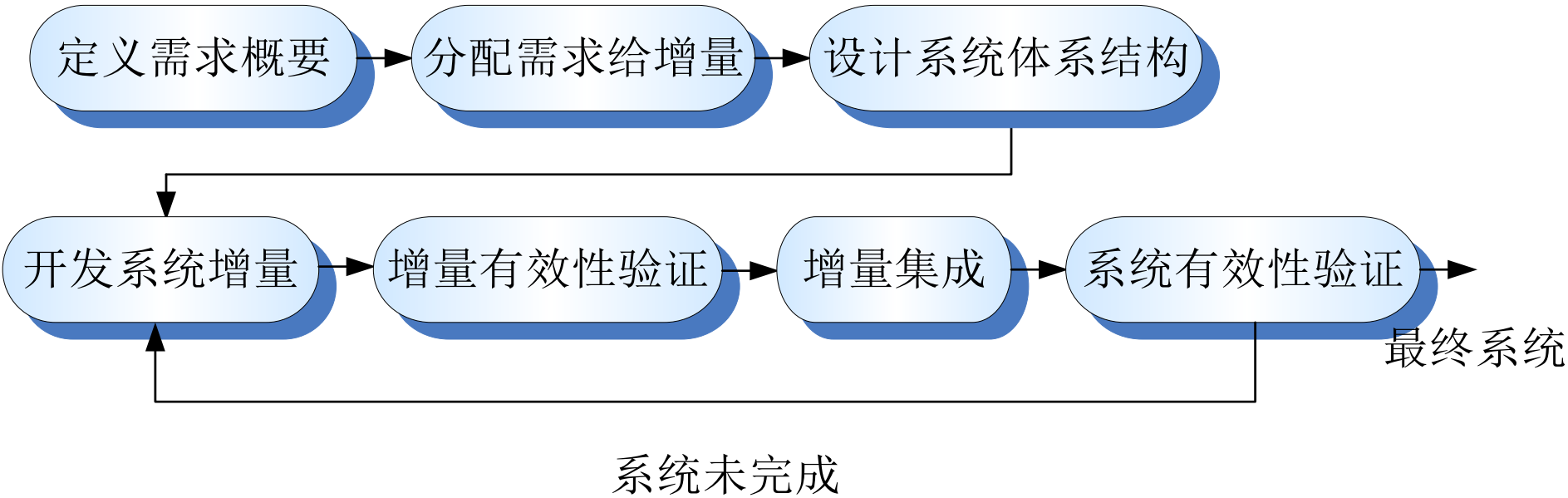
过程反复

- 系统需求在项目进行期间总是进化的，所以对大型系统来说经常是早期阶段反复的过程
- 反复可以应用于任何通用过程模型
- 两个混合模型
  - 增量式开发
  - 螺旋式开发

增量式开发

- 系统不是一次交付，而是将要求的功能分成多次增量进行开发和交付
- 用户需求是有优先顺序的。优先级最高的需求包含在最初的增量中。
- 一旦一个增量的开发开始时，需求就要冻结。，虽然后面的增量可以继续进化

增量式开发


定义需求概要
分配需求给增量
设计系统体系结构
开发系统增量
增量有效性验证
增量集成
系统有效性验证
最终系统
系统未完成

增量式开发的优势

- 客户要求的功能随着每次增量被交付，所以系统功能可以较早看到　前期宣传
- 较早的增量可以作为原型帮助引出后面增量的需求　引导需求
- 项目总体失败的风险降低
- 最高优先级的系统服务由于是最早增量，得到最多的测试　优先级高的服务往往是关键的服务

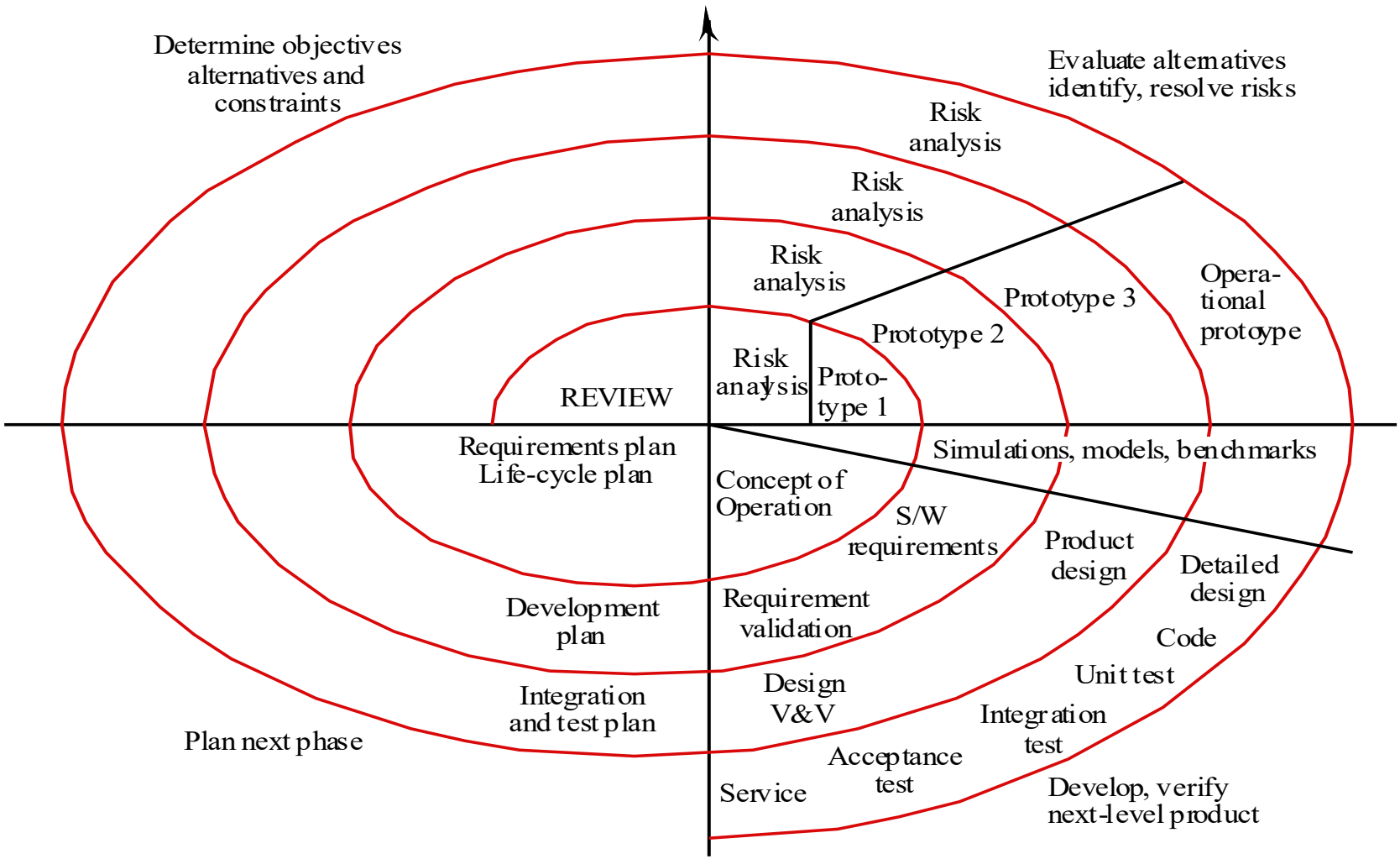
极限编程

- 基于非常小的功能增量的开发和交付的一种新的方法
- 依靠连续的代码改进，用户参与到开发团队一起开发

螺旋式开发

- 将过程表示为螺旋线，而不是用一系列活动和活动间的回溯来表示
- 螺旋线中的每个回路表示过程中的一个阶段
- 没有固定的阶段。螺旋线中的回路根据需要选取
- 风险需要明确评估、然后在过程中解决

软件过程的螺旋式模型


Determine objectives
alternatives and
constraints
Evaluate alternatives
identify, resolve risks
Risk
analysis
Risk
analysis
Risk
analysis
Prototype 3
Opera-
tional
protoype
Prototype 2
Risk
analysis
Proto-
type 1
REVIEW
Requirements plan
Life-cycle plan
Simulations, models, benchmarks
Concept of
Operation
S/W
requirements
Product
design
Detailed
design
Development
plan
Requirement
validation
Code
Unit test
Integration
and test plan
Design
V&V
Integration
test
Plan next phase
Acceptance
test
Service
Develop, verify
next-level product

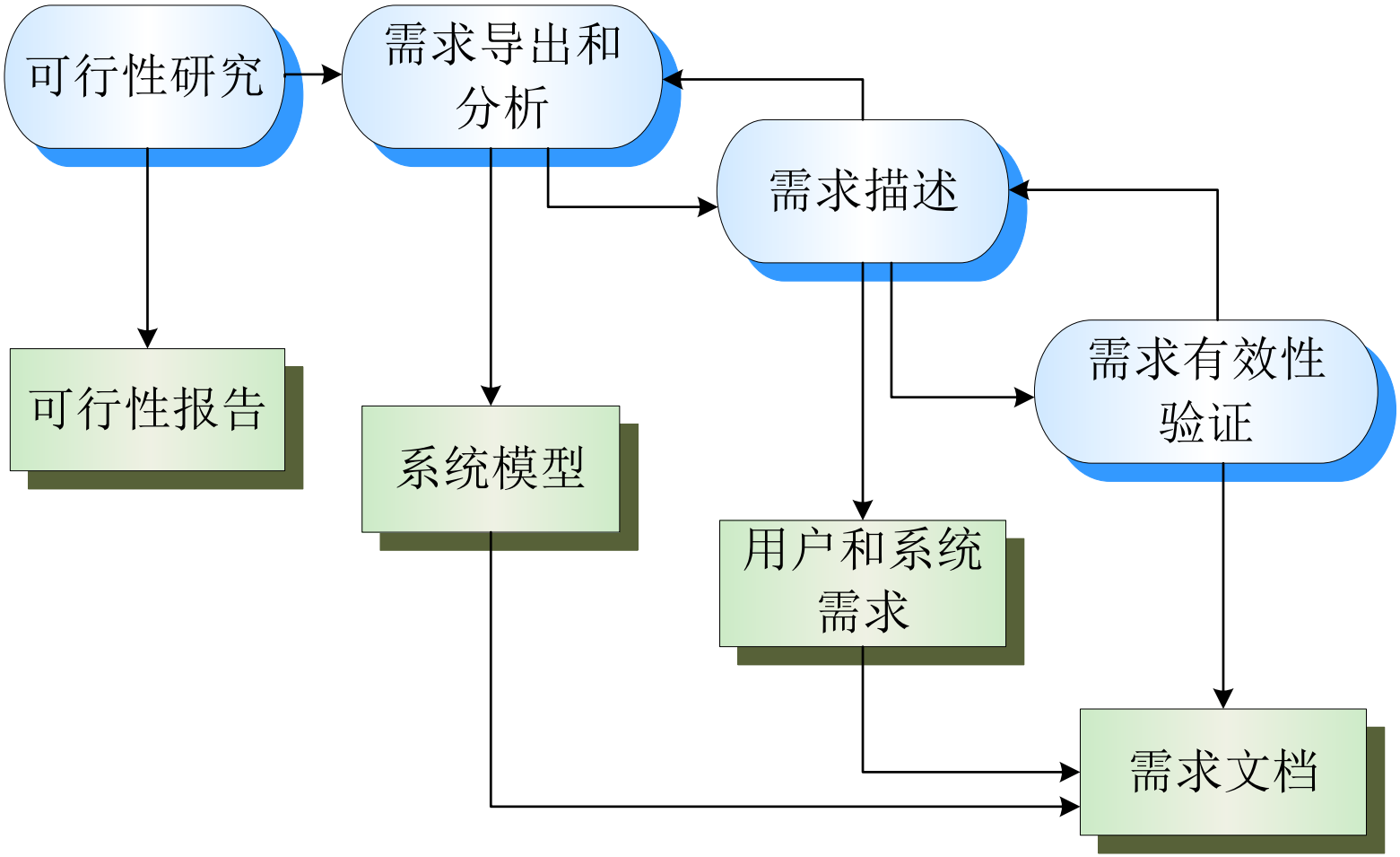
螺旋式模型的扇区

- 目标设置
  - 确定阶段的目标
- 风险评估和规避
  - 分析风险、采取行动规避风险
- 开发和有效性验证
  - 选择一个系统开发模型，可以是任何通用模型
- 规划
  - 对项目进行评审、规划螺旋线的下一个阶段

软件描述

- 建立系统需要哪些服务以及系统操作和开发的约束的过程
- 需求工程过程
  - 可行性研究
  - 需求导出和分析
  - 需求描述
  - 需求有效性验证

需求工程过程



软件设计和实现

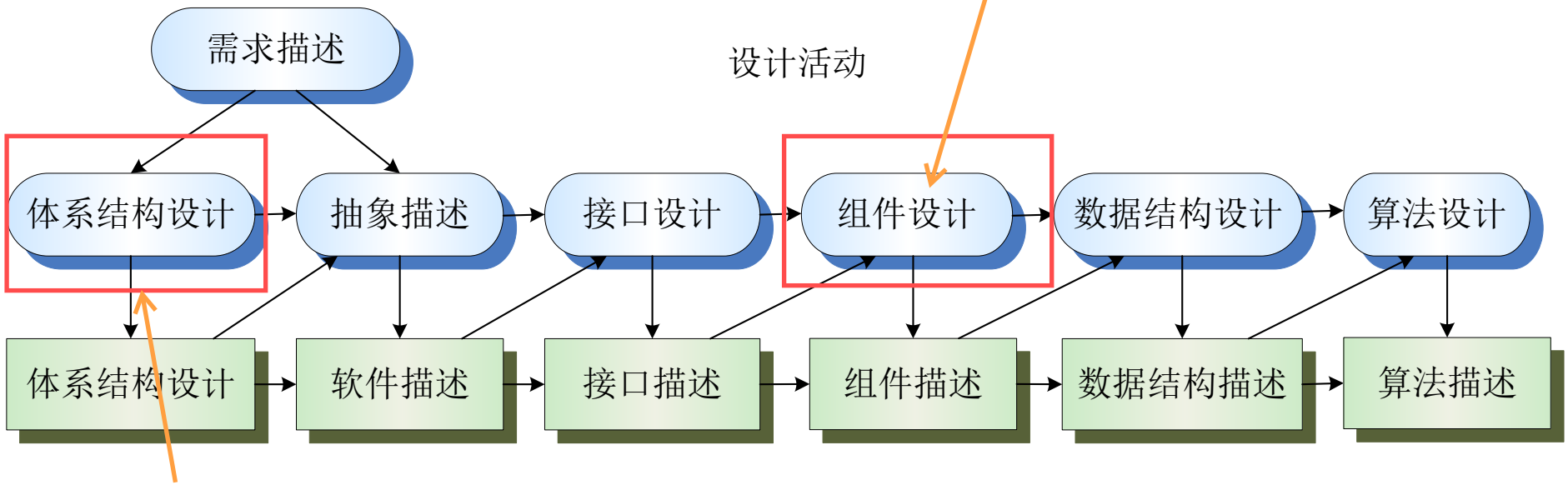
- 将系统描述转化为可执行的系统的过程
- 软件设计
  - 设计实现描述的软件结构
- 实现
  - 将这个结构转化为可执行程序
- 设计和实现活动是紧密联系的，可能是交叉进行的。

设计过程活动

- 体系结构设计
- 抽象描述
- 接口设计
- 组件设计
- 数据结构设计
- 算法设计

软件设计过程

设计的详细程度视具体情况（如组员人数、能力水平等因素）而定



图：体系结构图（组件数量7-20个为宜）设计产品
表：对图中组件的描述
文字：补充描述

设计方法

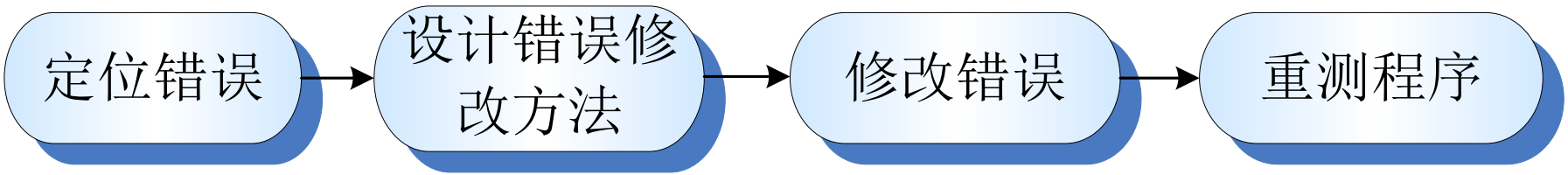
- 开展软件设计的系统方法
- 设计通常是图形化模型的文档
- 可能的模型
  - 数据流模型
  - 实体-关系模型
  - 结构模型
  - 对象模型

编程和调试

- 将设计转化为一个程序，从程序中排除错误
- 编程是个人行为，没有通用的编程过程
- 程序员完成一些程序测试以发现程序中的缺陷，在调试过程中排除缺陷。

调试：使程序能够运行
测试：使程序能够正确的运行

调试过程



软件有效性验证

V&V：verify & validate
另一种翻译：验证和确认

检验：强调过程（如上课点名）
验证：强调结果（如期末考试）

- 检验和有效性验证是为了说明系统符合它的描述并且满足系统用户的需求
- 包括检查、评审和系统测试

检查：静态，不需要运行程序
测试：动态，需要运行程序

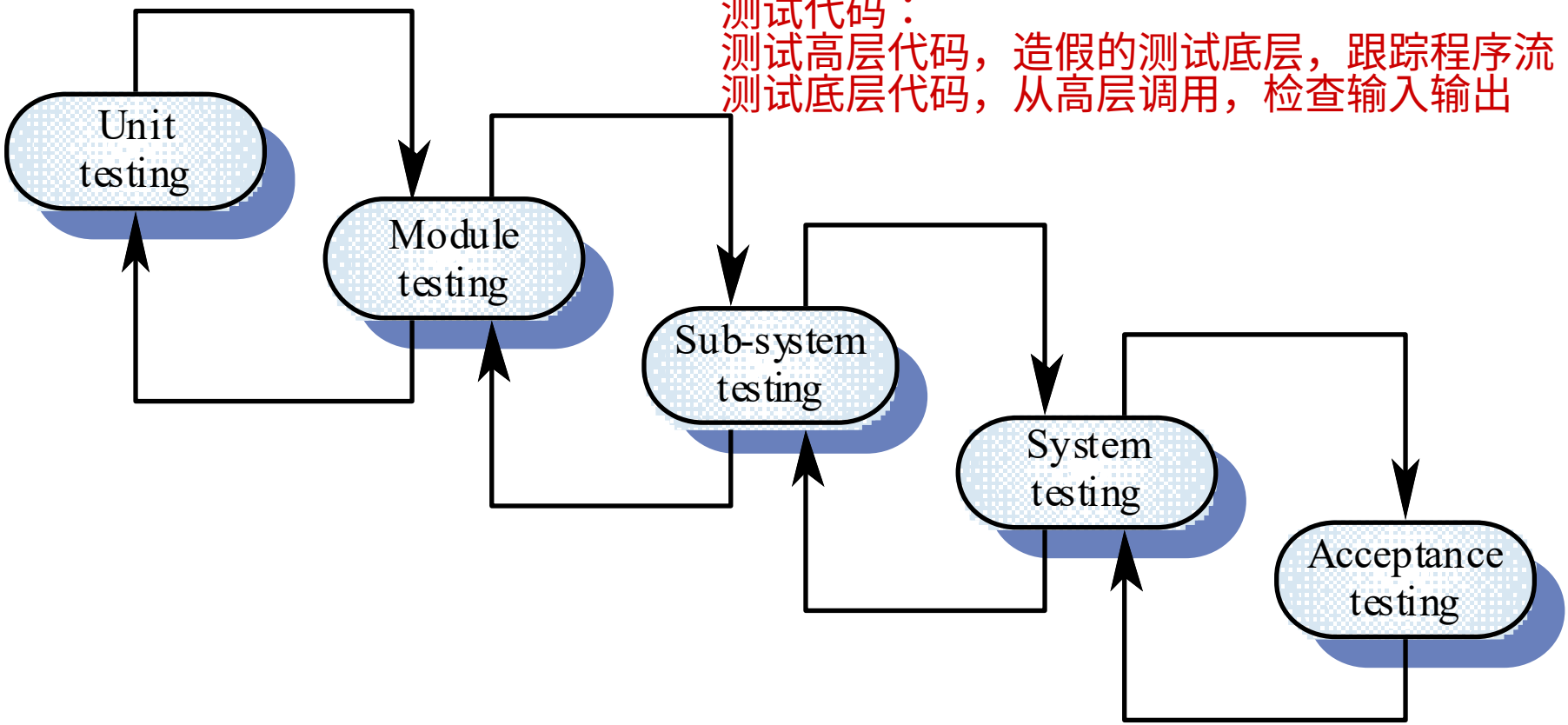
检验　　验证

- 系统测试是用测试用例来执行系统，测试用例来自系统描述，使用真实的系统数据

测试用例：测试数据和期望的输出

测试过程

测试代码：
测试高层代码，造假的测试底层，跟踪程序流；
测试底层代码，从高层调用，检查输入输出



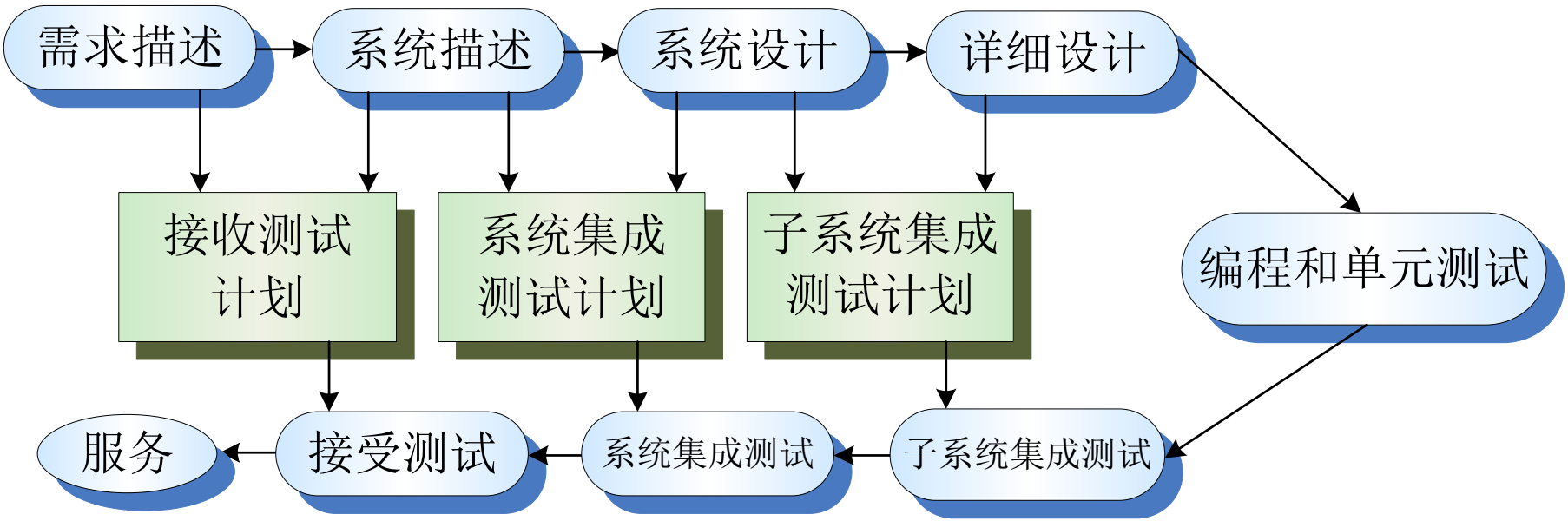
组件测试　　集成测试　　用户测试

测试的阶段

- 单元测试
  - 测试独立的组件
- 模块测试
  - 测试一组互相关联的组件的集合
- 子系统测试
  - 模块集成到子系统并测试。关键是接口测试。
- 系统测试
  - 测试整个系统。测试总体特性。
- 接收测试
  - 用客户的数据测试以确定系统能被接收。

测试的阶段

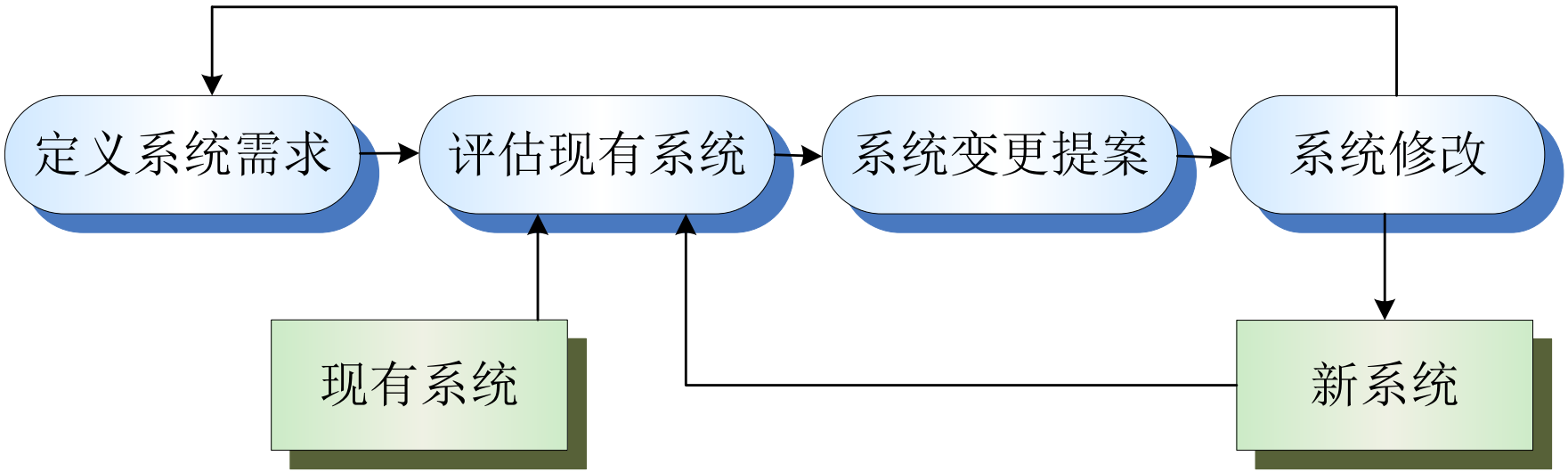
V模型（瀑布模型）



软件进化

- 软件具有灵活、可以改变的固有特性
- 需求会随着业务情况的改变而改变，所以支持业务的软件也应该变更和进化
- 虽然在开发和进化（维护）之间有一个划分，但这种划分已经越来越不合适了

系统进化



自动化过程支持 (CASE)

- 计算机辅助软件工程(CASE)支持软件开发和进化过程的软件
- 过程活动自动化
  - 系统模型开发的图形化编辑器
  - 管理设计实体的数据字典
  - 用于用户界面构建的图形化用户界面生成器
  - 支持程序缺陷发现的调试器
  - 生成程序新版本的自动转换器

Case技术

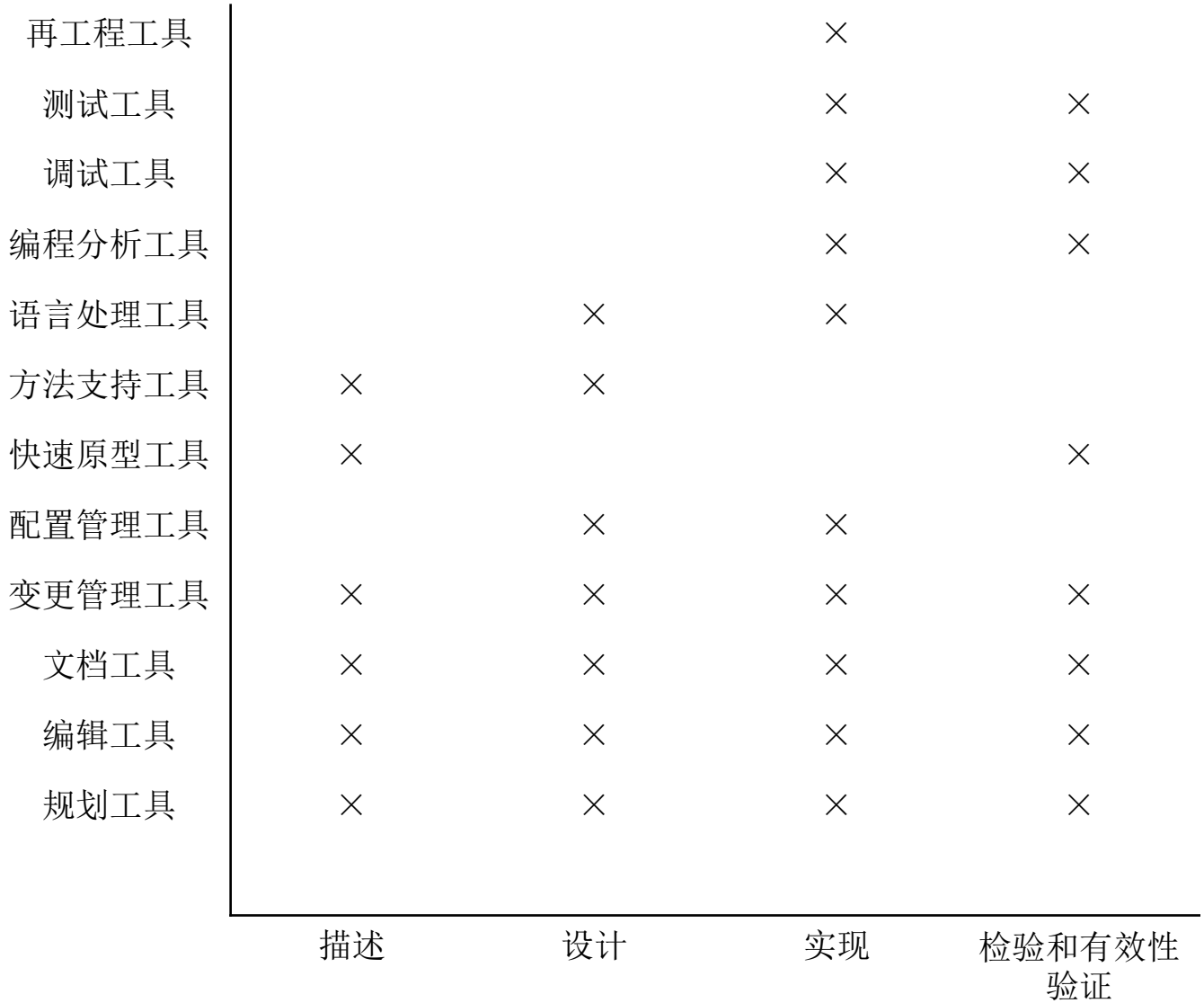
- Case技术带来了软件过程的很大提高，尽管没有人们预期的提高那么大
  - 软件工程要求有创造性的思考 – 这个是无法自动化的
  - 软件工程是一个团队活动，对于大型项目，很多时间花在团队沟通上。CASE技术无法真正地支持这些活动。

CASE 分类

- 分类有助于我们理解CASE工具的不同类型以及它们对软件过程活动的支持
- 从功能角度分
  - 依照它们的专门功能分类
- 从过程角度分
  - 根据所支持的过程活动来分类
- 从集成角度
  - 根据集成方式分类

功能工具分类

工具类型	举例
规划工具	PERT tools, estimation tools, spreadsheets
编辑工具	Text editors, diagram editors, word processors
变更管理工具	Requirements traceability tools, change control systems
配置管理工具	Version management systems, system building tools
建立原型工具	Very high-level languages, user interface generators
方法支持工具	Design editors, data dictionaries, code generators
语言处理工具	Compilers, interpreters
程序分析工具	Cross reference generators, static analysers, dynamic analysers
测试工具	Test data generators, file comparators
调试工具	Interactive debugging systems
文档工具	Page layout programs, image editors
再工程工具	Cross-reference systems, program re-structuring systems

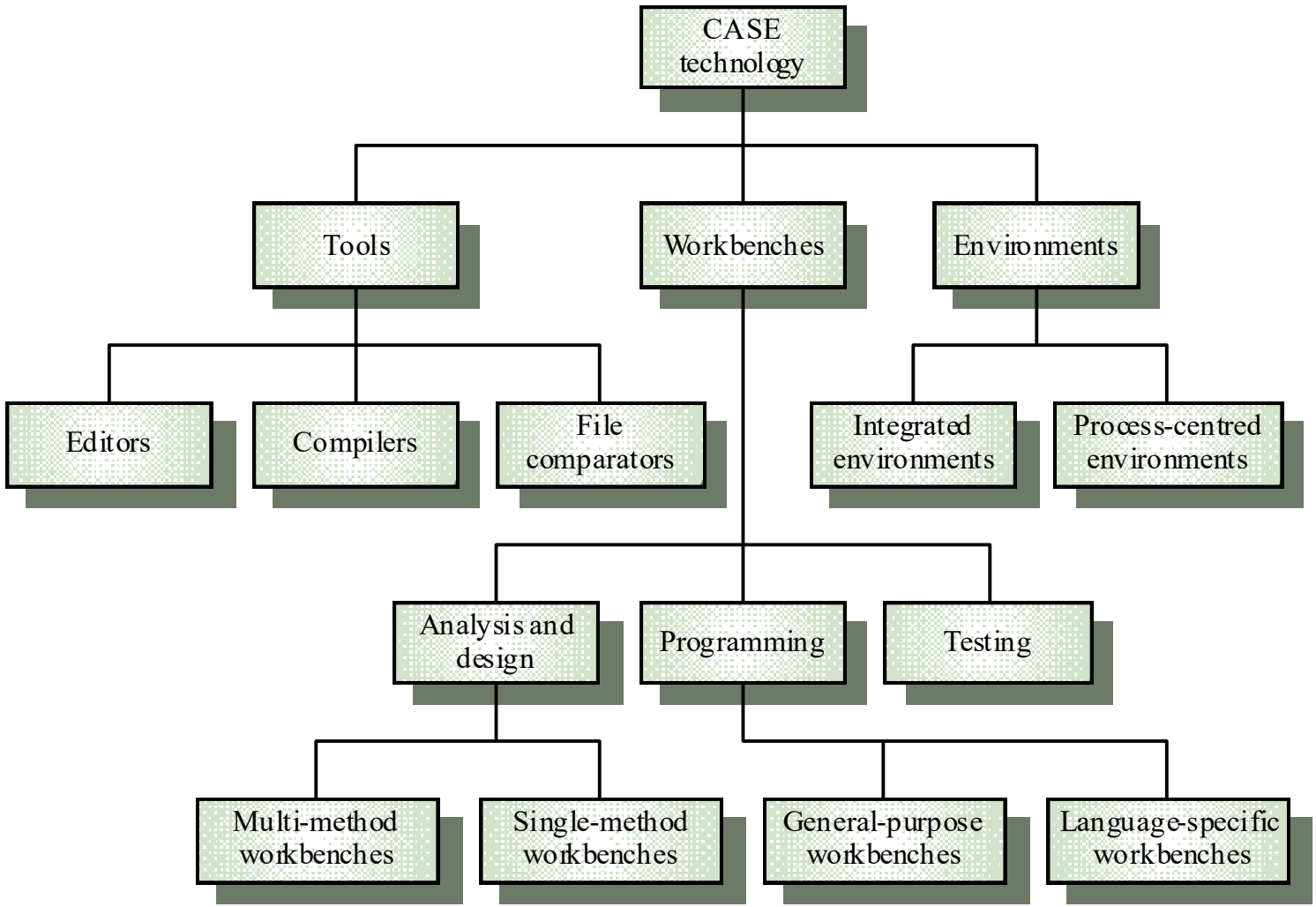

再工程工具
测试工具
调试工具
编程分析工具
语言处理工具
方法支持工具
快速原型工具
配置管理工具
变更管理工具
文档工具
编辑工具
规划工具
描述
设计
实现
检验和有效性验证

基于活动的分类

CASE集成度

- 工具
  - 支持单独的过程任务例如设计一致性检查、文本编辑、等等
- 工作平台
  - 支持过程阶段例如描述或设计。通常包括一组集成的工具
- 环境
  - 支持整个软件过程或者主要部分。通常包括几个工作平台

工具、工作平台、环境


CASE technology
Tools
Workbenches
Environments
Editors
Compilers
File comparators
Integrated environments
Process-centred environments
Analysis and design
Programming
Testing
Multi-method workbenches
Single-method workbenches
General-purpose workbenches
Language-specific workbenches

要点

- 软件过程是生产和进化一个软件系统涉及的活动。通过过程模型来表达。
- 普通的活动包括描述、设计、实现、有效性验证和进化
- 通用过程模型描述软件过程的组织
- 反复过程模型将软件过程描述为周期性活动

要点

- 需求工程是开展软件描述的过程
- 设计和实现过程将描述转化为一个可执行程序
- 有效性验证包括检查系统是否符合描述以及用户的需求
- 进化是关于系统使用后的修改
- CASE技术支持软件过程的活动

课堂练习

- 请画一下软件生命周期的V模型
- 写出每个阶段的主要工作内容和产出物