检验和有效性验证

- 确保一个软件系统满足用户的需求

目标

- 软件检验和软件有效性验证之间的区别
- 在程序中发现缺陷的程序检查过程及其在检验和有效性验证中的作用
- 静态分析方法作为一种检验技术
- 净室软件开发方法

内容

- 检验和有效性验证规划
- 软件检查
- 自动静态分析
- 净室软件开发方法

检验与有效性验证

- 检验:
  “我们是否在正确地建立一个产品"
- 软件应该符合它的描述
- 有效性验证:
  “我们是否在建立一个正确的产品"
- 软件应该满足用户真正的需要

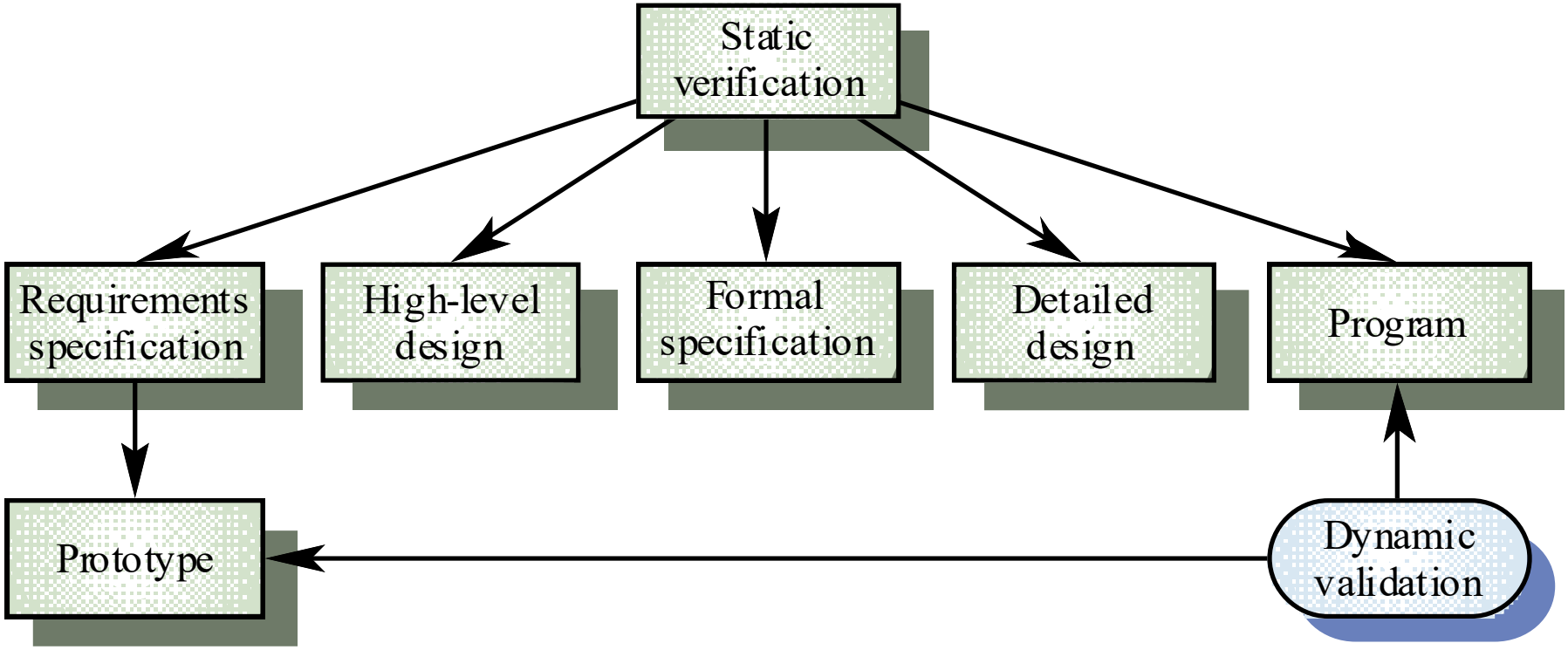
检验和有效性验证过程

- 是个全生命周期的过程——检验和有效性验证应该应用到软件过程的每个阶段
- 有两个主要目的
  - 发现系统中的缺陷
  - 评估系统在实际操作中是否可用

静态和动态检验

- *软件检查* 分析系统的静态表述以发现问题（静态检验）
  - 可能增加基于工具的文档和代码分析
- *软件测试* 实际运行和观察软件的行为 (动态检验)
  - 系统实用测试数据执行，然后观察其运行的行为

静态和动态V&V


Static verification
Requirements specification
High-level design
Formal specification
Detailed design
Program
Prototype
Dynamic validation

程序测试

- 可以发现错误的存在，不是错误的不存在
- 一次成功的测试能够发现一个或更多的错误
- 对于非功能需求是唯一的验证手段
- 应该和静态检验联合使用以提供全面的检验和有效性验证

测试的类型

- 缺陷测试
  - 设计测试以发现系统缺陷
  - 成功的测试是发现系统中缺陷的存在
  - 第23章中详述
- 统计性测试
  - 设计测试以反映用户输入的频度。用于可靠性估计。
  - 第24章中详述

V&V 目标

- 检验和有效性验证的目标是要确信软件符合使用目的
- 这并不意味着程序完全没有缺陷
- 而是表明系统足以满足使用要求。使用类型决定了所需的信任程度

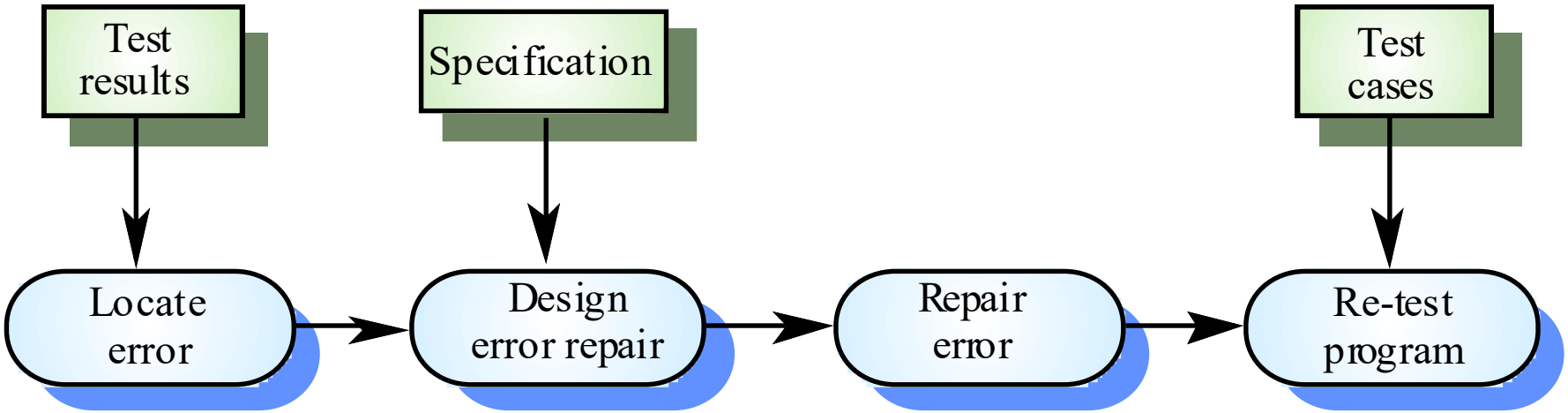
V & V 的信任程度

- 依赖于系统的设计目标、用户的期望和市场环境
  - 软件功能
    - » 系统需要的信任程度取决于该软件在机构中的重要程度
  - 用户期望
    - » 用户对某些软件期望很低
  - 市场环境
    - » 在市场上推出产品可能比在程序中找出缺陷更加重要

测试和调试

- 缺陷测试和调试是截然不同的过程
- 检验和有效性验证是一个证明软件系统中存在缺陷的过程
- 调试是一个对缺陷定位和修改的过程
- 调试需要先对程序行为作出假设，然后对这些假设进行测试以发现系统错误

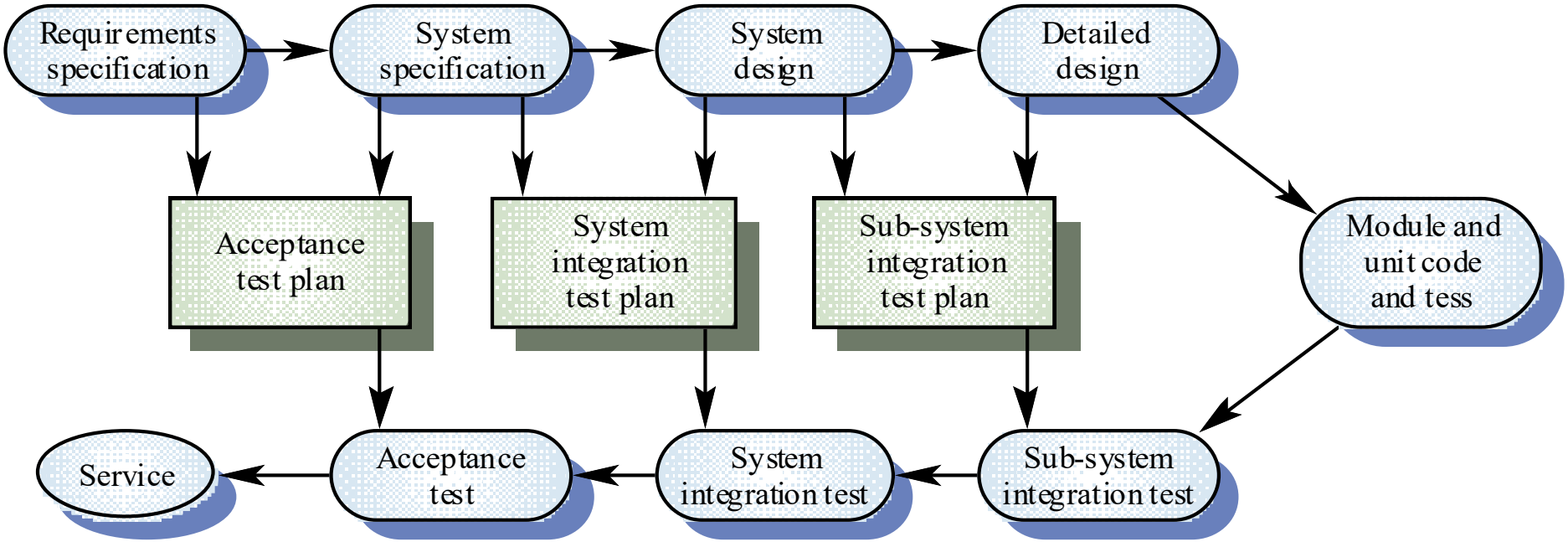
调试过程


Test results
Specification
Test cases
Locate error
Design error repair
Repair error
Re-test program

V & V 规划

- 仔细规划才能从测试和检查过程中收获更多
- 规划应该从开发的较早阶段开始
- 规划应该在静态检验和动态测试之间均衡考虑
- 测试规划是关于测试过程的标准定义，而不是描述产品测试内容

开发的V字形模型


Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and tess
Service
Acceptance test
System integration test
Sub-system integration test

软件测试计划的结构

- 测试过程
- 需求跟踪
- 测试项目
- 测试进度安排
- 测试记录过程
- 硬件和软件需求
- 约束

软件检查

- 相关人员检查代码和文档，以发现不规范和缺陷
- 不要求执行系统，所以可在系统实现前使用
- 可应用于系统的任何表示形式(需求, 设计, 测试数据等)
- 发现错误非常有效的方法

软件检查

- 一次检查中能发现许多不同的缺陷。在测试中，一个缺陷可能屏蔽其它缺陷，所以需要多次执行
- 由于复用了领域和程序语言知识，所以评审人员可能见过经常出现的错误

检查和测试

- 检查和测试是互相补充的，而不是对立的检验技术
- 两者都应该在检验和有效性验证过程中使用
- 软件检查可以检验是否和描述一致，但无法检验是否和用户的真正需求一致
- 软件检查无法检验非功能特征，如性能，可用性等。

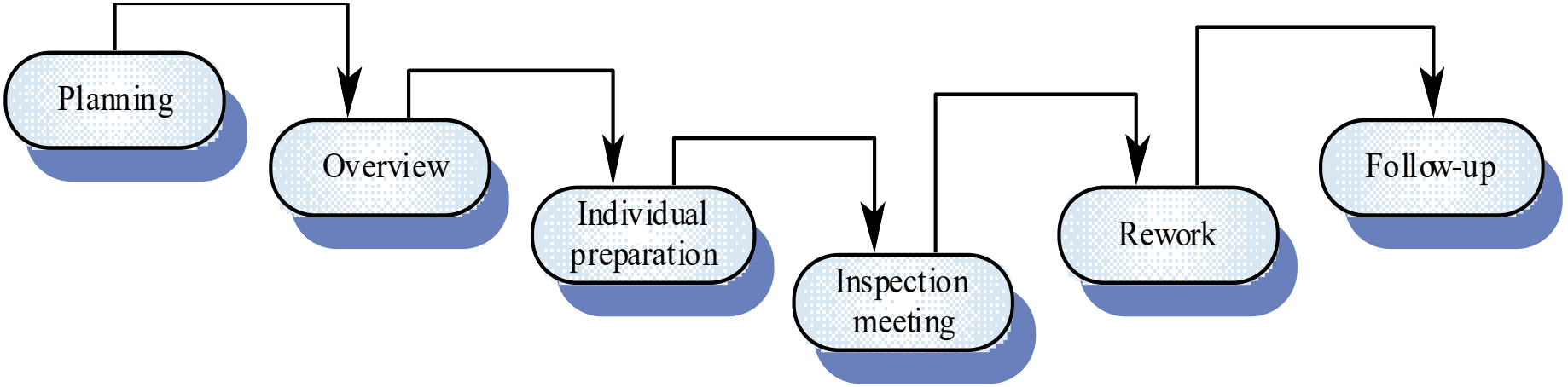
程序检查

- 正式规范的文档评审过程
- 目的是发现缺陷 (而不是改正缺陷)
- 缺陷可能是逻辑错误，可能预示着错误情况的代码中的不规范(e.g.没有初始化的变量)或者不服从标准

程序检查的预处理

- 对被检查的代码有一个精确的描述
- 检查小组的成员应该熟悉机构的标准
- 有一个最新的语法正确的代码版本
- 要准备一个错误检查的核对清单
- 管理层必须接受检查将带来软件过程早期的费用增加
- 管理层不能将软件检查用于员工评价

检查过程


Planning
Overview
Individual preparation
Inspection meeting
Rework
Follow-up

检查的程序

- 检查团体总体观察系统代码
- 代码和相关文档预先分发给检查团队
- 检查并发现错误
- 对发现的错误作出修改
- 可能需要也可能不需要重新检查

检查团队

- 至少由4人组成
- 所要检查的代码的作者
- 检查者，发现错误、多余内容和矛盾的地方
- 讲解者，向团队讲解代码
- 主席/仲裁者，主持会议、记录发现的错误
- 其他角色，如抄写员，首席仲裁者

检查清单

- 常见错误的清单
- 错误清单跟编程语言有关
- 编程语言的类型检查越弱，错误清单越长
- 举例：初始化，变量命名，循环终止，数组边界等。

程序检查内容

缺陷分类	检查内容
数据缺陷	所有的程序变量堵在使用前被初始化了吗 所有的常数都命名了吗 数组的上边界应该等于数组长度还是长度减1 如果使用字符串，定界符应该显式地指定吗 有缓冲区溢出的可能性吗
控制缺陷	对每一个条件语句，条件是正确的吗 每一个循环都能终止吗 复合语句被正确地括起来了吗 对case语句，所有可能的情况都考虑了吗 若每一个case语句都需要跟一个break语句，有遗漏吗
输入/输出缺陷	所有的输入变量都使用了吗 所有的输出变量在输出前都被赋值了吗 有未料到的输入引起系统崩溃吗
接口缺陷	所有的函数和方法调用都使用了正确数量的参数吗 形参和实参类型匹配吗 参数顺序都对吗 如果组件访问共享内存，它们都有相同的共享内存结构模型吗
存储管理缺陷	如果一个链接的结构被修改了，所有的链接都得到重新赋值了吗 如果使用了动态存储，空间分配正确吗 如果空间不再使用，需要显式地对空间释放吗
异常管理缺陷	所有可能的错误状态都已经考虑到了吗

检查的比率

- 总体观察阶段每小时约500行源代码
- 个别准备阶段每小时约125行源代码
- 会议检查每小时约90-125行源代码
- 检查是一个昂贵的过程
- 检查 500 行的费用约40个人时。

自动静态分析

- 静态分析器是源代码文本处理的软件工具
- 分析程序文本以发现潜在的错误情形，提交给V & V团队
- 对帮助检查非常有效。是人工检查的一个补充而不是代替。

自动静态分析内容

缺陷分类	检查内容
数据缺陷	变量在初始化之前被使用 变量被定义但从未使用过 变量被赋值两遍，但中间并未使用过 可能的数组越界 未声明的变量
控制缺陷	不可到达的代码 无条件循环
输入/输出缺陷	变量在两次输出间没有赋值过
接口缺陷	参数类型不匹配 参数个数不匹配 函数的结果没有用处 未调用过的函数和子程序
存储管理缺陷	未赋值的指针 指针参与运算

静态分析的阶段

- *控制流分析.* 检查带有多个出口或入口的循环，发现不可到达的代码等。
- *数据使用分析.* 检出未初始化的变量，一个变量赋值两次而中间未使用，变量定义后未使用等。
- *接口分析.* 检查过程声明和使用之间的一致性

静态分析的阶段

- *信息流分析.* 确定输出变量和输入变量的依赖关系。不是去发现异常本身，而是为代码检查和评审列出信息流的关系。
- *路径分析.* 确定程序中的可能路径以及路径中执行的语句。也是为了评审中的潜在使用。
- 这些阶段产生大量的信息。必须小心使用。

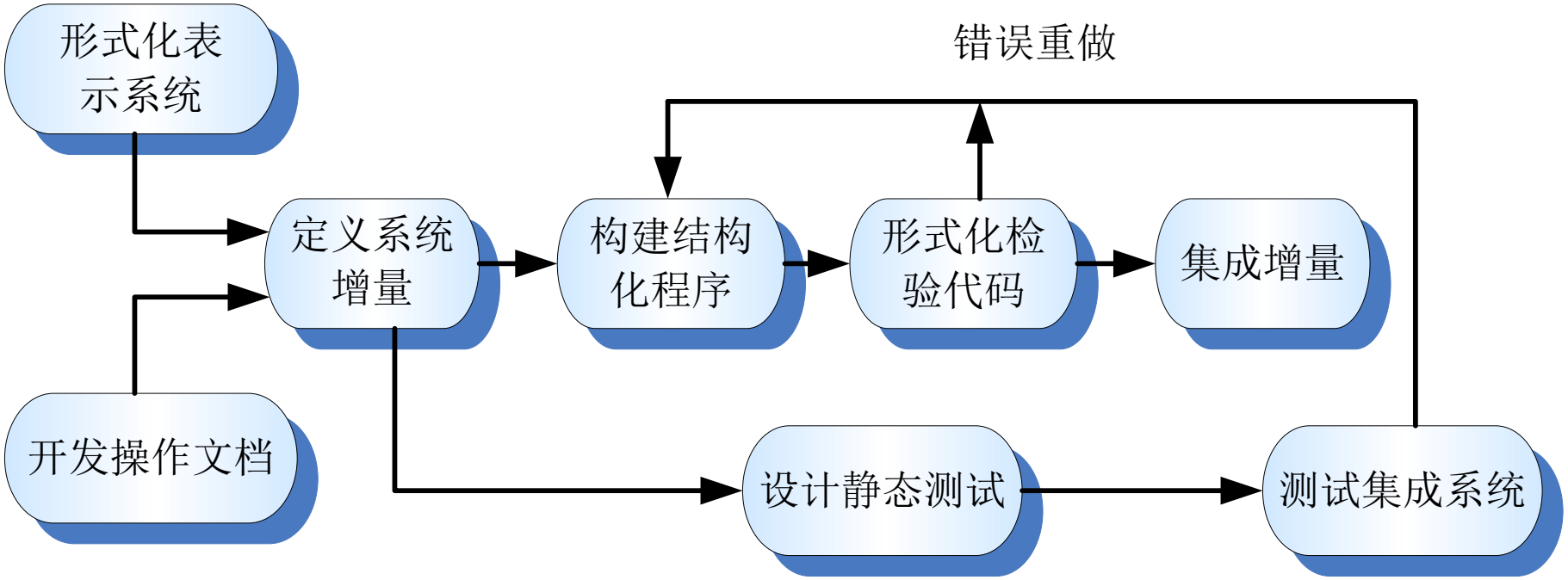
静态分析的使用

- 对于一种弱类型检查的语言特别有价值，如C语言，许多错误无法由编译器检测出来
- 对强类型检查的语言来说性价比不高，如Java，很多错误可以在编译中检测出来

净室软件开发

- 名称是从半导体生产过程中的“净室”得来的。其思想是缺陷避免，而不是缺陷移除。
- 基于下列要点的软件开发:
  - 增量开发
  - 形式化描述
  - 使用正确性论证进行静态检验
  - 用统计性测试以决定程序可靠性

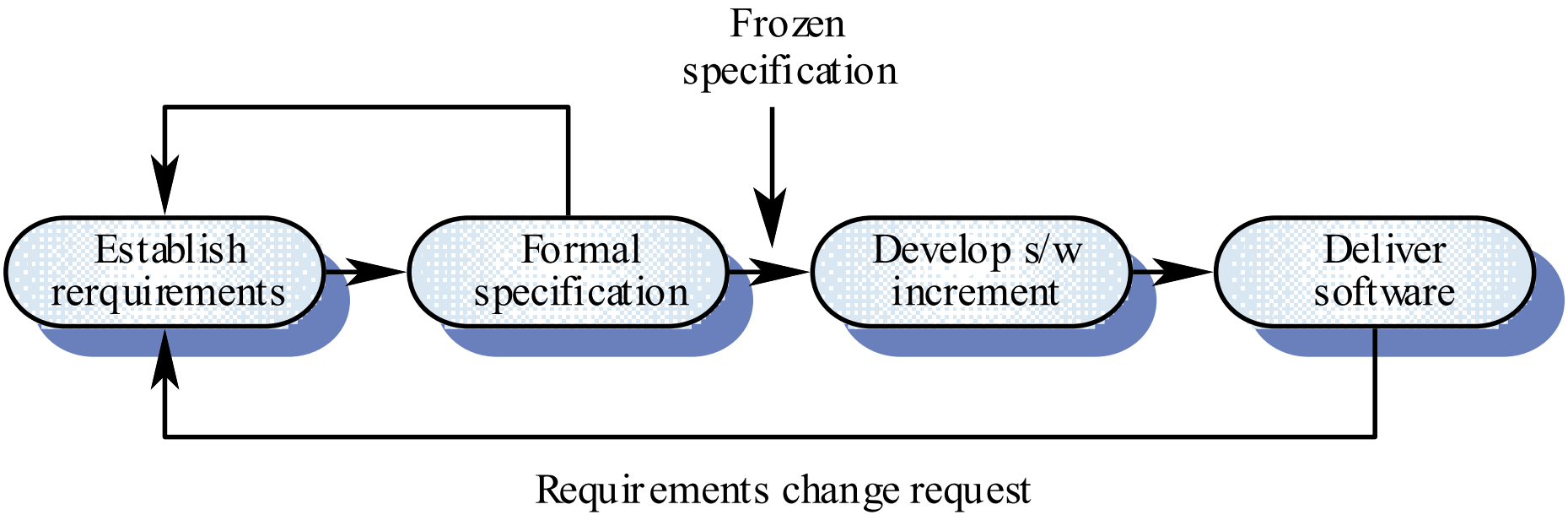
净室过程


形式化表示系统
错误重做
定义系统增量
构建结构化程序
形式化检验代码
集成增量
开发操作文档
设计静态测试
测试集成系统

净室过程的特征

- 形式化描述，使用状态迁移模型
- 增量式开发
- 结构化编程 – 使用有限的控制结构和抽象结构
- 静态检验，使用严格的检查
- 系统统计性测试

增量式开发


Frozen
specification
Establish
rerquirements
Formal
specification
Develop s/w
increment
Deliver
software
Requirements change request

形式化描述和检验

- 基于状态的系统模型，是系统描述和静态检查的基础
- 模型和系统之间的对应关系清晰
- 数学化的论证 (不是证明) 用来提高软件检查的信任程度

净室过程的团队

- *描述团队*. 负责开发和维护系统描述
- *开发团队*. 负责对软件的开发和检验。在这个检验过程中软件是不执行的，甚至不编译。
- *认证团队*. 负责开发统计性测试用例集合，在软件开发出来后对其进行测试. 使用可靠性增长模型来决定何时可靠性是可接受的。

净室过程的评估

- IBM公司用净室开发的系统在交付后故障很少，给人们留下深刻印象
- 独立的评估显示该过程并不比其它方法昂贵
- 比起传统的开发过程，错误更少
- 要由技术熟练、有责任心的软件工程人员执行时才起作用。

要点

- 检验和有效性验证不是一回事。检验的目的是要看软件是否符合其描述，而有效性验证的目的是要看软件是否满足了用户的要求。
- 测试计划应该指导测试过程
- 静态检验技术包括对程序的检查和分析以发现错误。

要点

- 程序检查在发现错误中是非常有效的
- 程序代码由一个小组检查，定位软件缺陷
- 静态分析工具可以发现程序的不规则之处，这些不规则之处可能预示着代码缺陷
- 净室开发过程依赖于增量式开发、静态检验和统计性测试。