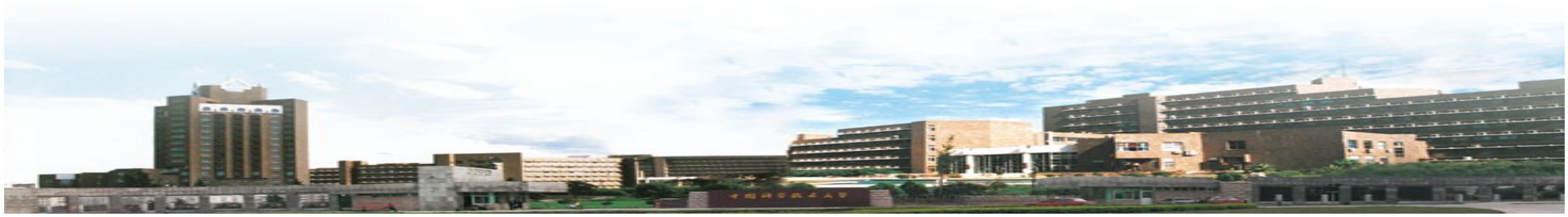
安全的软件开发生命周期

程绍银

sycheng@ustc.edu.cn



大纲

- 软件开发生命周期概述
- 传统的软件开发生命周期
- 安全的软件开发生命周期
- 其他安全软件开发生命周期模型
- 软件保护技术

大纲

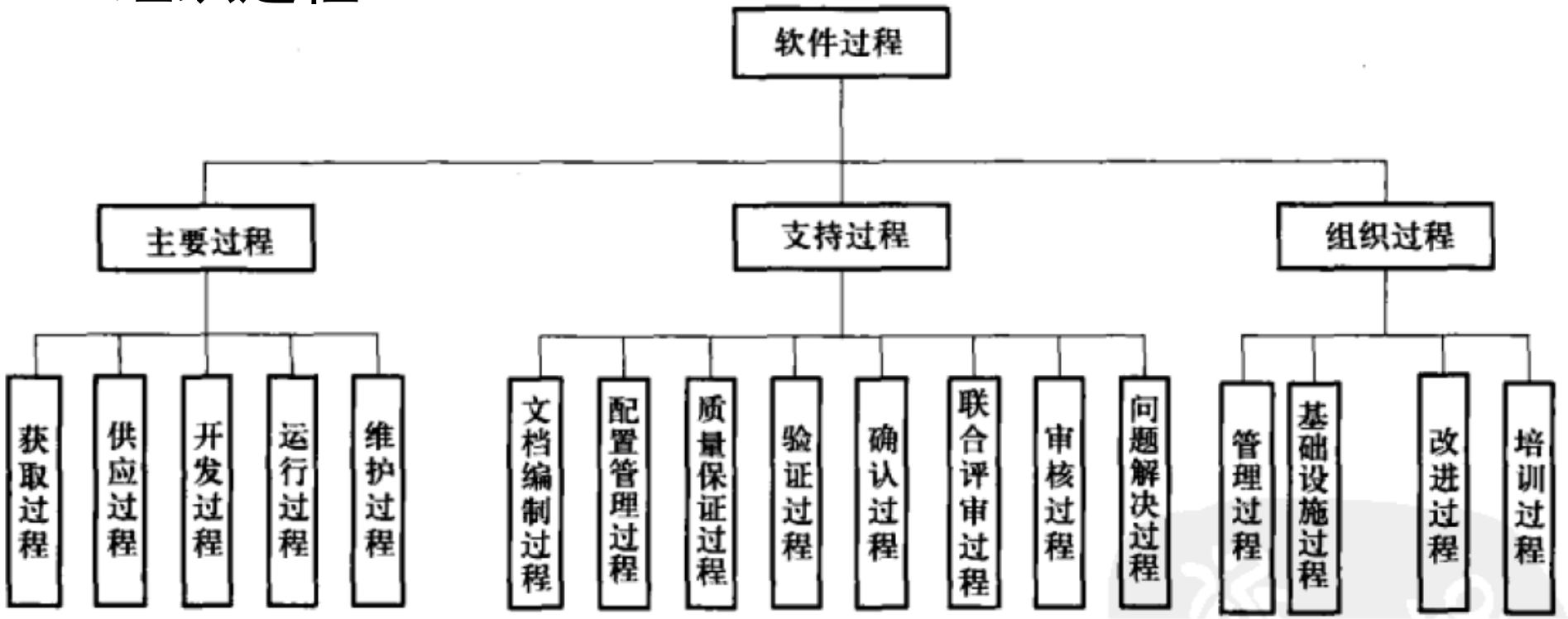
- ✓ 软件开发生命周期概述
- 传统的软件开发生命周期
- 安全的软件开发生命周期
- 其他安全软件开发生命周期模型
- 软件保护技术

软件过程

- 软件过程是为了获得高质量软件产品，在软件工具支持下由软件人员完成的一系列软件工程活动
- 软件过程规定了开发软件所需完成的各项任务和步骤，它与软件生成期、生产期模型、软件开发工具和参与开发的人员等多方面因素有关
- 1995年，《ISO/IEC 12207 信息技术——软件生成期过程》定义了17个软件过程

软件过程（ISO/IEC 12207）

- 主要过程
- 支持过程
- 组织过程



主要过程

- 获取：需方获取系统、软件产品或软件服务的活动。由需方定义需求，然后委托供方或双方一起进行需求分析，需求分析的结果要由需方确认。需方应该准备招标证书、合同及验收条款
- 供应：由供方向需方提供系统、软件产品或软件服务的活动。供方要对需求进行评审，然后准备投标，中标后签订合同。制定项目计划、实施计划、开展评审、交付产品
- 开发：开发者定义并且开发软件产品。由开发者参与进行系统分析和系统结果设计，然后进行软件分析、软件结构设计和软件详细设计。在设计的基础上着手编码和测试，将测试的软件集成在一起，进行系统集成和系统测试，最后进行安装，并且提供验收支持

主要过程

- 运行：供方协助需方制定运行计划并进行运行测试，最终由运行支持者运行系统
- 维护：维护者提供系统维护服务的活动。用户对系统运行中出现的问题进行记录；维护者分析维护记录，实施变更；对维护结果进行评审。维护还包括为满足用户新的要求，对软件进行修改或二次开发

支持过程

- 文档编制
- 配置管理
- 质量保证
- 验证
- 确认
- 联合评审
- 审核
- 问题解决

组织过程

- 管理：制定计划、监控计划的实施，评价计划实施情况；包括产品管理、项目管理和任务管理等内容
- 基础设施：开发、运行和维护等过程所需要的硬件、软件、工具、技术、标准，建议基础设施，并能提供维修服务
- 改进：对整个软件生成期过程进行评估、度量、控制和改进
- 培训：对人员进行相关培训所需的活动，包括制定培训计划、编写培训资料、培训计划的实施

软件生存周期

- 可行性分析和项目计划
- 需求分析
- 软件设计
- 程序编写
- 软件测试
- 运行/维护

软件生存周期

- 可行性分析和项目计划
  - 确定要开发系统的总目标，给出它的功能、性能、可靠性以及接口等方面的要求
  - 研究完成该项软件任务的可行性，探讨解决问题的可能性
  - 制定完成开发任务的实施计划，连同可行性研究报告，提交管理部门审查

软件生存周期

- 需求分析
  - 对待开发软件提出的需求进行分析并给出详细的定义，准确地确定系统必须要做什么，应具备哪些功能
  - 编写出软件需求说明书及初步的用户手册，提交管理机构评审
- 软件设计
  - 把已确定的各项需求转换成一个相应的体系结构
  - 该体系结构是由一些意义明确的模块组成，进而对每个模块要完成的工作进行具体的描述，编写设计说明书，提交评审

软件生存周期

- 程序编写
  - 把软件设计转换成计算机可以接受的、结构良好、清晰易懂的程序代码
- 软件测试
  - 设计测试用例，以检验软件的运行情况，确保软件质量
- 运行/维护
  - 已交付的软件投入正式使用，并在运行过程中进行适当的维护，纠正软件的错误，使得软件能够适应环境的变化及扩充软件的功能

软件生存周期

- 软件开发生命周期指软件开发的开始到完成经历的过程，这个过程定义了软件开发经历的不同阶段
- 传统软件开发生命周期，主要从软件功能实现的角度出发，其基本目的是如何合理地组织开发流程以高效地完成软件的各项功能，通常只是注重软件功能的定义，实现与测试，安全策略却没有得到充分地考虑。即使是在测试环节考虑到软件系统的安全性，也往往是在编码完成后进行，而没有将软件安全的思想贯穿到软件开发的整个过程
- 安全软件开发生命周期是将安全原则渗透在整个软件开发的生命周期中，即在每个开发阶段均需要考虑安全性

软件修复代价

- 研究发现：软件的安全缺陷或问题发现得越早，则修复的费用越低。因此，采用安全软件开发生命周期是非常必要的

阶　段	纠正问题的花费	阶　段	纠正问题的花费
定义	1	单元测试	15
全局模块设计	2	集成测试	22
模块详细设计	5	系统测试	50
编码	10	发布后	100

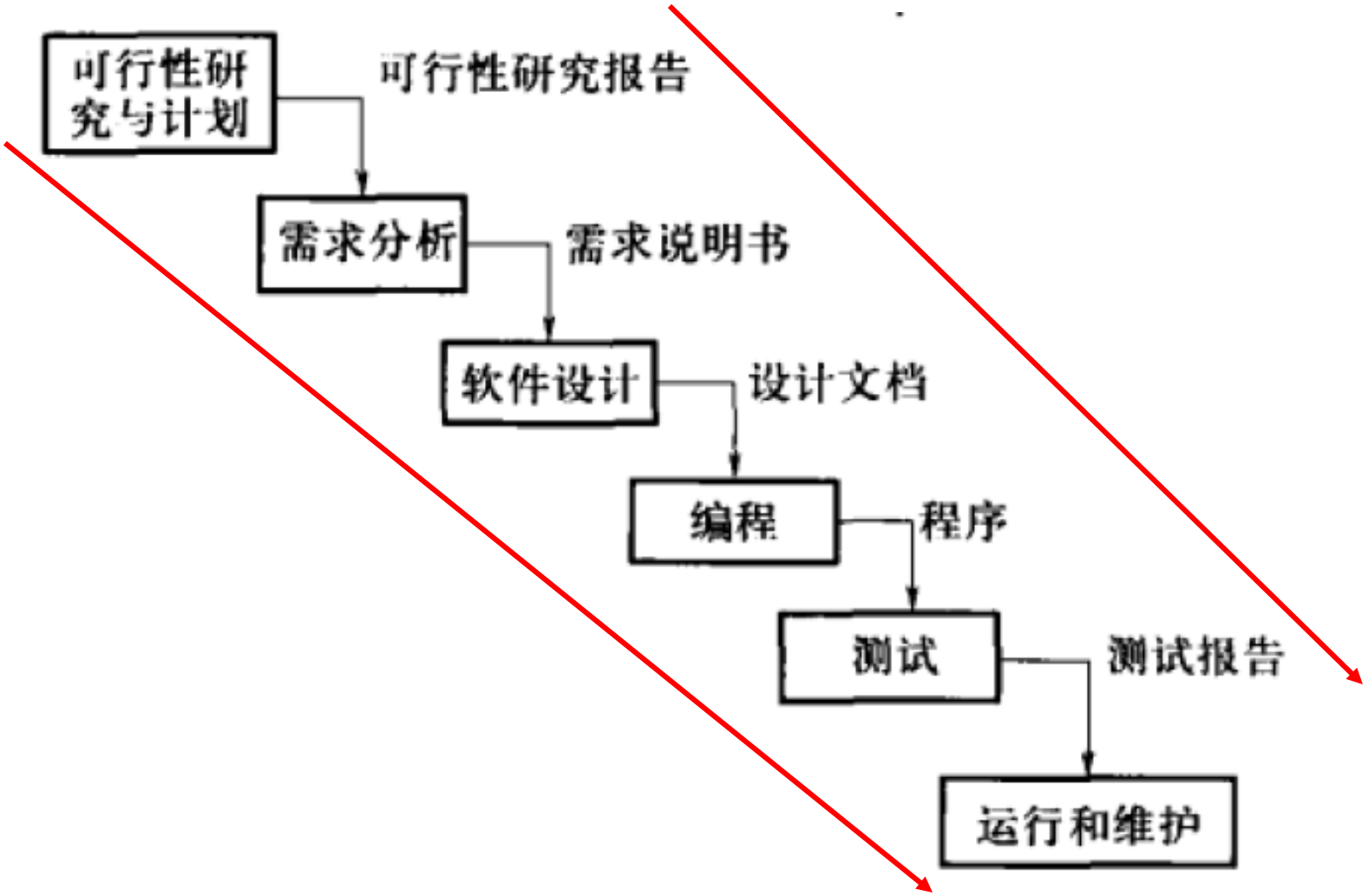
大纲

- 软件开发生命周期概述
- ✓ 传统的软件开发生命周期
- 安全的软件开发生命周期
- 其他安全软件开发生命周期模型
- 软件保护技术

瀑布模型

- 瀑布模型是于1970年由W. Royce最早提出的软件开发模型
- 瀑布模型将软件生命周期的各项活动规定为一定顺序链接的若干阶段工作，这些工作之间的衔接关系是从上到下、不可逆转，如同瀑布一样，因此称为瀑布模型

瀑布模型



需求分析

- 回答“软件需要完成什么功能？”
- 通过研讨或调查研究，对用户的需求进行收集，然后去粗取精、去伪存真、正确理解，最后把他用标准的软件工程开发语言（需求规格说明书）表达出来，供设计人员参考
  - iPhone：创造需求，引领消费
- 在用户中进行调查研究，和用户一起确定软件需要解决的问题
  - 建立软件的逻辑模型
  - 编写需求规格说明书文档，根据用户需求，通过不断沟通，反复修改，并最终得到用户的认可

需求分析

- 是用户需求分析，把所设计实现的系统作为一个整体，是从使用者的角度来看的；而不是从开发者的角度，不要牵涉到内部模块间的需求
  - 系统提供什么样的功能和性能？
  - 系统的输入是什么？包括配置文件。
  - 有哪些输出，输出结果格式和主要内容如何？
- 从文档内容上，可以包括：
  - 功能：对于工具的使用者而言工具提供的功能
  - 性能：描述工具的运行时的时间消耗、内存消耗等
  - 约束：工具的运行环境和一些保护性功能（如容错性等）

软件设计

- 概要设计、详细设计
- 最终任务：将软件分解成一个个模块（可以使一个函数、过程、子程序、一段带有程序说明的独立的程序和数据，也可以是可组合、可分解和可更换的功能单元），并将模块内部的结构设计出来

软件设计－主要工作

- 利用结构化分析方法、数据流程图和数据字典等方法，根据需求说明书的要求，设计建立相应的软件系统的体系结构
- 进行模块设计，给出软件的模块结构，用软件结构图表示，将整个系统分解成若干个子系统或模块，定义子系统或模块间的接口关系
- 设计模块的程序流程、算法和数据结构，设计数据库
- 编写软件概要设计和详细设计说明书，数据库或数据结构设计说明书，编制测试计划

概要设计的五大视图

- 逻辑视图、运行视图、开发视图、数据视图和部署视图
- 逻辑视图：整个系统的抽象结构
  - 系统分成几个功能模块，各模块的功能及模块之间的关系
  - 不涉及具体的输入输出和部署
- 运行视图：系统运行时输入和输出之间的关系及处理流程
  - 对整个运行流程的描述：系统如何得到输入，中间经过哪些步骤的处理（一般和逻辑视图中的模块关联上），最后得到输出

概要设计的五大视图

- 开发视图：描述本系统与研发过程中使用到的其他工具之间的关系
  - 使用了哪些第三方软件或第三方库/SDK？
  - 实际上还可以更具体，例如还包含哪些程序包（SDK、解析器、中间件）、文件组织结构、编译依赖关系、目标单元（jar、exe、dll等）
  - 开发视图和逻辑视图的静态元素通常有映射关系
- 数据视图：通常指数据库中的实体关系图
  - 有多少表？各个表的主要字段有哪些？具体含义是什么？表之间有什么关系？具体点，还可以包括
  - 如果系统中没有使用数据库，这部分可以省略

概要设计的五大视图

- 部署视图：描述系统与部署环境的关系
  - 系统实际运行时，如何部署？是否分布式？等等
  - “目标程序及其依赖的运行库和系统软件”最终如何安装/部署到物理机器，以及如何部署机器和网络来配合软件系统的可靠性、可伸缩性等
  - 部署视图和运行视图的关系：运行视图特别关注目标程序的动态执行情况，而部署视图重视目标程序的静态位置问题；部署视图还要考虑软件系统和包括硬件在内的整个IT系统之间是如何相互影响的
  - 如果系统只是单机软件，这部分可以省略

编程

- 把软件设计转换成计算机可以接受的程序，选择某一种程序设计语言，编写出源程序清单
  - 基于软件产品的开发质量的要求，充分了解软件开发语言、工具的特性和编程风格
  - 进行编码
  - 提供源程序清单
- 用工具自动生成代码框架（如Rational Rose）

测试

- 目的：以较小的代价发现尽可能多的错误
- 关键：设计一套出色的测试用例
- 白盒测试：测试对象是源程序，依据的是程序内部的逻辑结构来发现软件的编程错误、结构错误和数据错误（如逻辑、数据流、初始化等错误）。其用例设计的关键，是以较少的用例覆盖尽可能多的内部程序逻辑结构
- 黑盒测试：依据的是软件的功能或软件行为描述，发现软件的接口、功能和结构错误。其用例设计的关键，是以较少的用例覆盖模块输出和输入接口

运维（运行和维护）

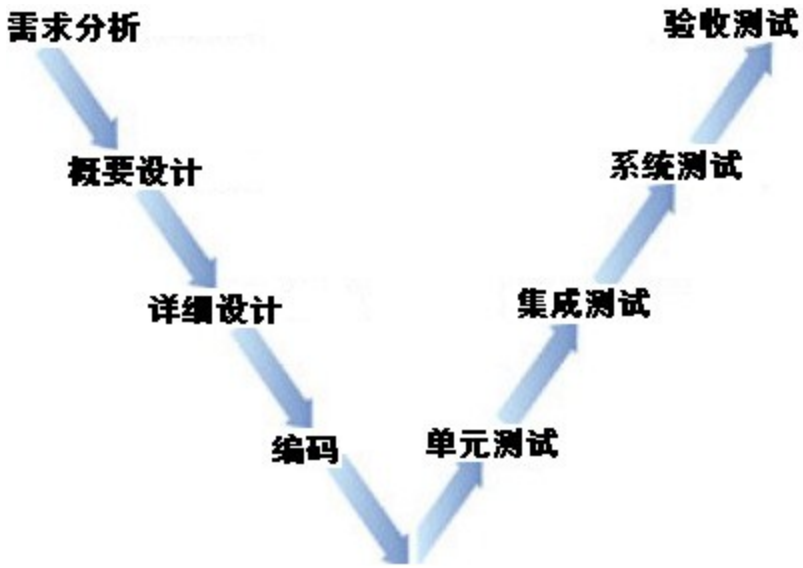
- 根据软件运行的情况，对软件进行适当修改，以适应新的要求；以及纠正运行中发现的错误
- 运维阶段的成本是比较高的，设计不到位或者编码测试考虑不周全，可能会造成软件维护成本的大幅度提高

瀑布模型的缺点

- 在项目开始阶段，开发人员和用户对需求的描述常常是不全面的。比如一些日常行为活动，开发人员通常对项目所涉及的领域不了解，所以理解上难免会出现遗漏或者偏差。需求阶段如果没有发现这些问题，就会影响到后面的工作
- 瀑布模型是由文档驱动的。瀑布模型中的各个阶段所做的工作都是文档说明。当用户在使用软件时往往会产生一些新的想法，或许会对软件的使用方面提出一些建议，而此时想对系统修改难度会很大
- 开发过程中，事先选择的技术或需求迅速发生变化，需要返回到前面某个阶段，对前面的一系列内容进行修改，这样势必会影响整个软件开发角度

V模型

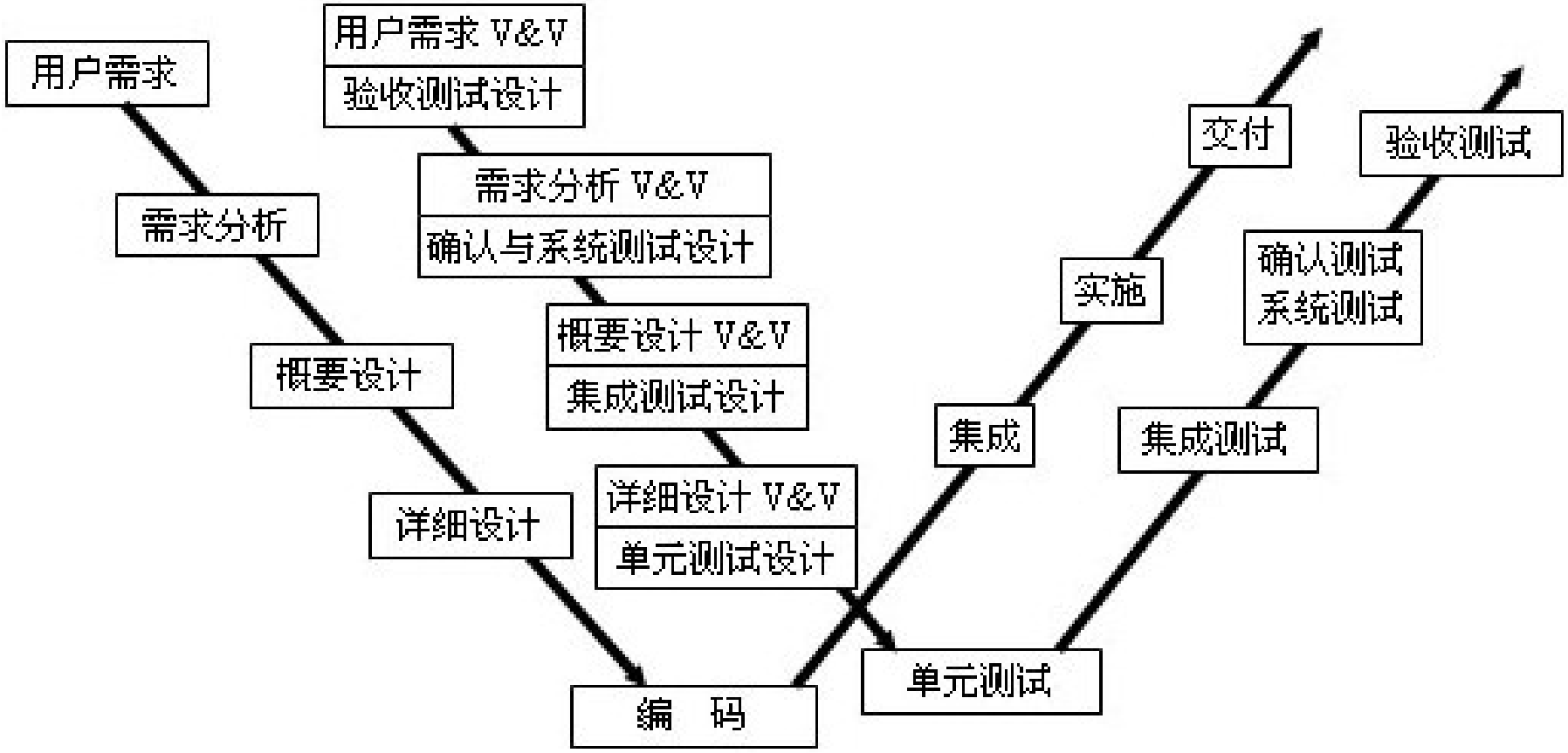
- 单元测试：检测代码的开发是否符合详细设计的要求
- 集成测试：检测此前测试过的各组成部分是否能完好地结合到一起
- 系统测试：检测已集成在一起的产品是否符合系统规格说明书的要求
- 验收测试：检测产品是否符合最终用户的需求



W模型

- W模型由Evolutif公司提出，相对于V模型，W模型增加了软件各开发阶段中应同步进行的验证和确认活动
- W模型由两个V字型模型组成，分别代表测试与开发过程，两者并行

W模型


用户需求
用户需求 V&V
验收测试设计
需求分析
需求分析 V&V
确认与系统测试设计
概要设计
概要设计 V&V
集成测试设计
详细设计
详细设计 V&V
单元测试设计
编　码
单元测试
集成
集成测试
实施
确认测试
系统测试
交付
验收测试

W模型优点

- 测试与开发是同步进行的，测试伴随着整个软件开发周期
- 测试的对象不仅仅是程序，还包括需求、设计等，
- 有利于尽早地全面的发现问题，降低开发成本
  - 例如，需求分析完成后，测试人员就应该参与到对需求的验证和确认活动中，以尽早地找出缺陷所在
  - 对需求的测试也有利于及时了解项目难度和测试风险，及早制定应对措施，这将显著减少总体测试时间，加快项目进度

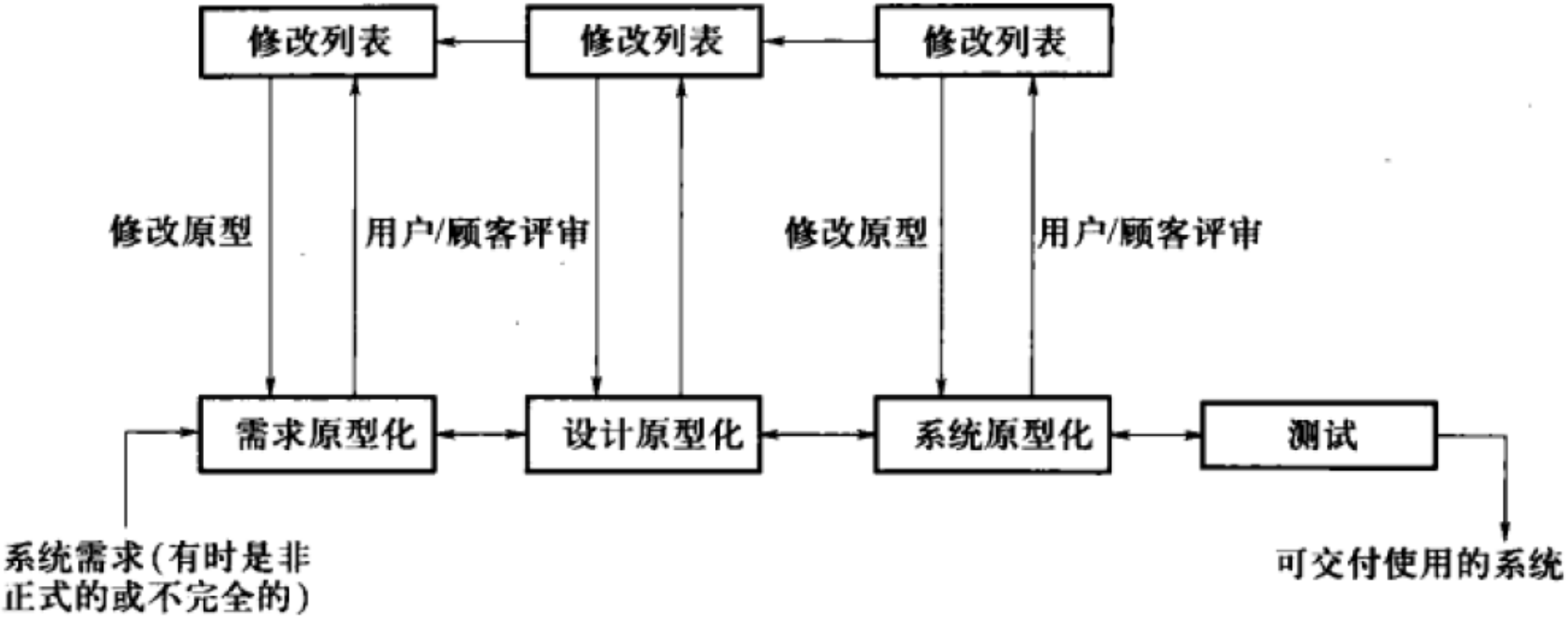
W模型不足

- 但W模型也存在局限性。在W模型中，需求、设计、编码等活动被视为串行的，同时，测试和开发活动也保持着一种线性的前后关系，上一阶段完全结束，才可正式开始下一个阶段工作。这样就无法支持迭代的开发模型。对于当前软件开发复杂多变的情况，W模型并不能解除测试管理面临的困惑。

原型模型

- 原型模型的基本思想
  - 软件开发人员在与用户进行需求分析时，以比较小的代价快速建立一个能够反映用户主要需求的原型模型，让用户在计算机上进行操作，然后提出改进意见
  - 开发人员根据用户的建议，对原型进行补充和完善，然后再由用户试用、评价、提出意见，重复这一过程，直至用户满意为止

原型模型


修改列表
修改列表
修改列表
修改原型
用户/顾客评审
修改原型
用户/顾客评审
需求原型化
设计原型化
系统原型化
测试
系统需求(有时是非正式的或不完全的)
可交付使用的系统

原型模型的优点

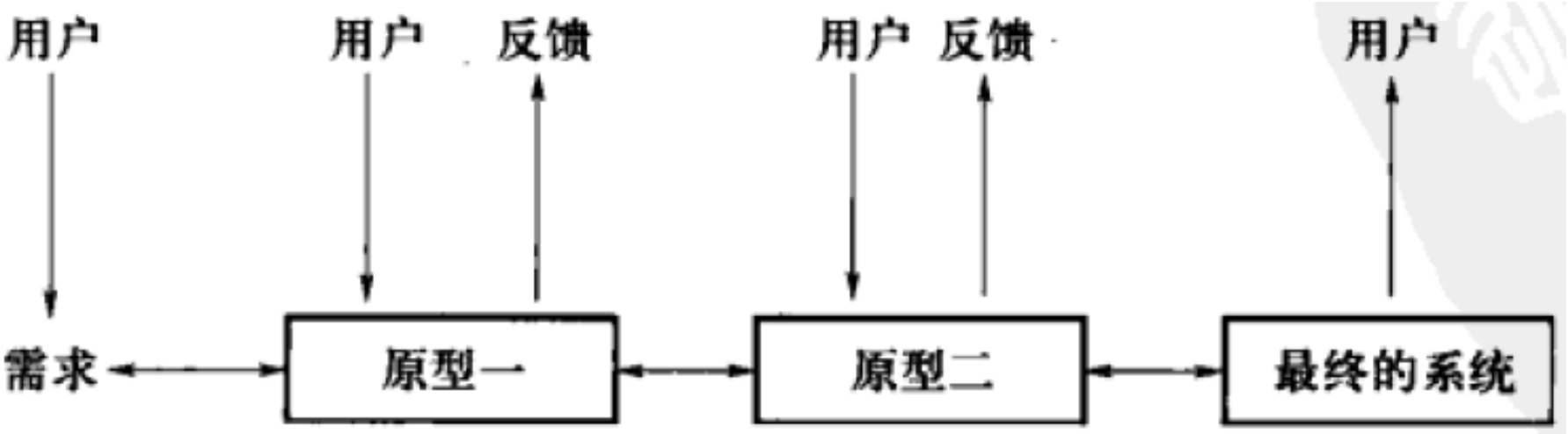
- 原型模型让用户有机会实践系统的基本功能，因而可以对不尽合理的内容提出修改意见和建议
- 原型模型法可以使开发者和用户充分交流，对一些模糊需求也能够处理
- 开发人员通过建立原型模型对系统了更深层次的理解，在设计和编码时可以尽量减少出错，有助于软件的开发工作顺利进行
- 当快速模型的某个部分是利用软件工具自动生成的时候，可以把这部分内容用到最终的软件系统中，比如一些界面或生成报表等
- 原型模型使总的开发费用降低，时间缩短
- 原型模型可以使用户对系统更为满意，也有利于维护
- 用户在使用原型模型时已经对系统有了初步了解，因此，建立模型的过程也相当于是用户的一个学习软件

原型模型的缺点

- 文档
- 版本管理

渐进模型

- 渐进模型的目的是和客户一起工作，从最初的大概的需求说明演化出最终的系统
- 渐进原型的目的是逐渐理解需求，没必要一次性完全理解需求



大纲

- 软件开发生命周期概述
- 传统的软件开发生命周期
- ✓ 安全的软件开发生命周期
- 其他安全软件开发生命周期模型
- 软件保护技术

安全的软件开发生命周期

- 安全软件的开发生命周期(SSDL)旨在通过软件开发的各个步骤来确保软件的安全性，其目标是确保安全的软件得以成功实现
- 通常由5个主要部分组成
  - 安全原则、规则及规章
  - 安全需求工程
  - 架构和设计评审、威胁建模
  - 软件安全编码
  - 软件安全测试

安全原则、规则和规章

- 安全原则、规则和规章通常被视为保护性需求

安全需求工程

- 这里通常是特定功能需求所需的特有安全需求，这些安全需求有别于系统范围的安全策略和安全规范

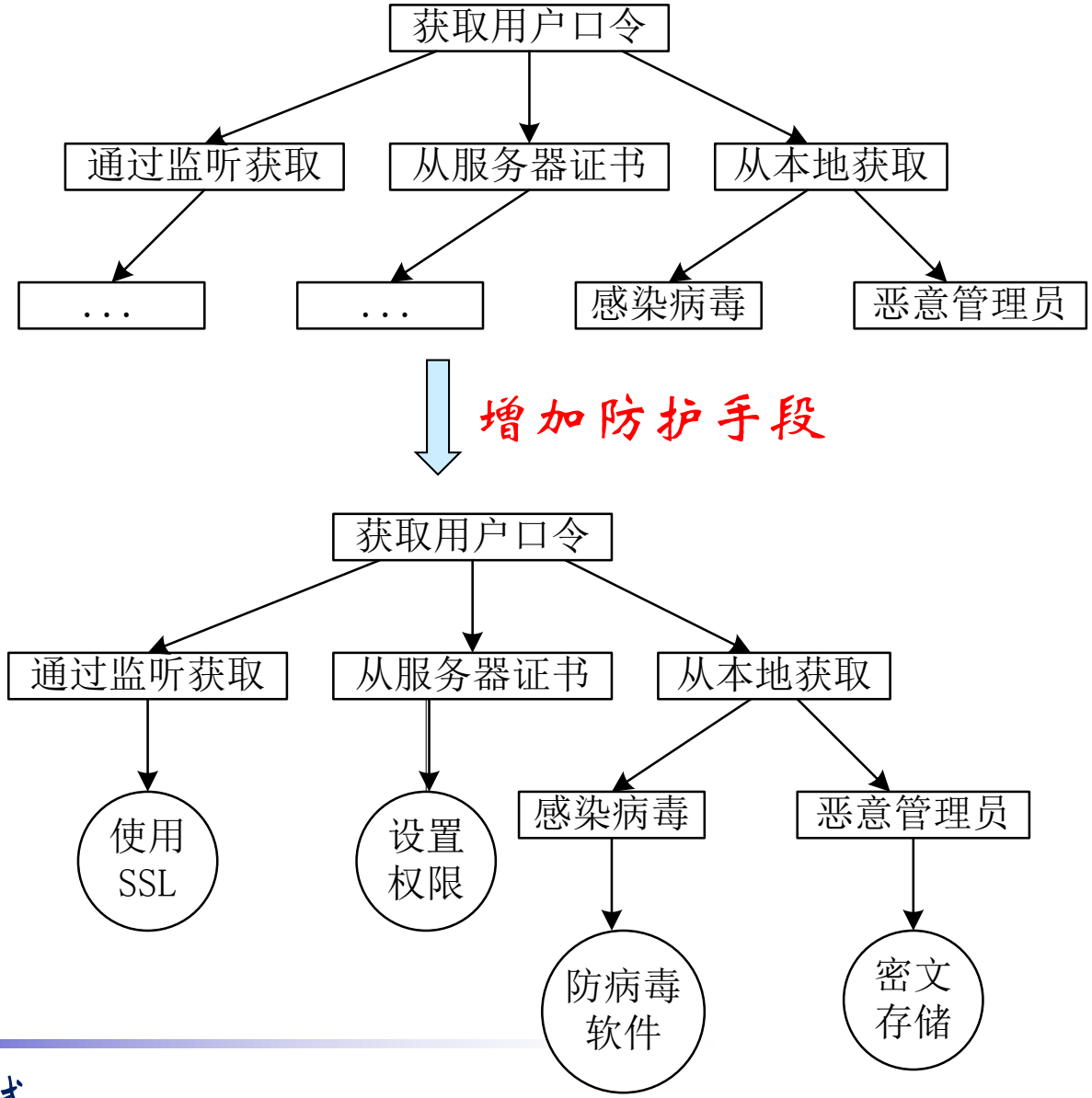
架构和设计评审、威胁建模

- 软件的架构和设计应该被安全分析人员尽早评审，避免形成有安全缺陷的体系结构和设计
- 为了避免设计漏洞，即在软件系统分析和设计阶段就遗漏对安全威胁的考虑，需要进行威胁建模，例如系统是否需要实体认证，是否需要保护信息的私密性。威胁建模有利于及早发现安全问题

威胁建模

- 软件设计阶段，考虑安全问题
  - 安全方面有哪些目标需要达到
  - 软件可能遇到的攻击和安全隐患
- 威胁建模过程
  - 在项目组中成立一个安全小组
  - 分解系统需求（从安全角度）
  - 确定系统可能面临哪些威胁（威胁树）
  - 选择应对威胁或者缓和威胁的方法
  - 确定最终技术（威胁树中增加防护手段）

威胁树


获取用户口令
通过监听获取
从服务器证书
从本地获取
. . .
. . .
感染病毒
恶意管理员
增加防护手段
获取用户口令
通过监听获取
从服务器证书
从本地获取
使用SSL
设置权限
感染病毒
恶意管理员
防病毒软件
密文存储

软件安全编码

- 需要代码的实现者对软件漏洞的来源有所了解，软件编码人员应该遵照一些软件安全编码原则
- 编译器做安全检查

安全编码原则

- 内存安全怎样实现？
- 怎样保证线程安全？
- 如何科学地处理异常？
- 输入输出安全怎样保障？
- 怎样做权限控制？
- 怎样保护数据？
- 怎样对付篡改和抵赖？
- 怎样编写优化的代码？

软件安全测试

- 测试决定软件的质量
  - 对软件的可用性进行测评
  - 对软件的安全性进行最大限度的保障
- 普通功能测试的目的
  - 确保软件不会去完成没有预先设计的功能
  - 确保软件能够完成预先设计的功能
- 安全测试：一轮多角度、全方位的攻击和反攻击
  - 非常灵活，测试用例没有太多的预见性
  - 没有固定的步骤可以遵循
  - 工作量大，并且不能保证完全地加以解决

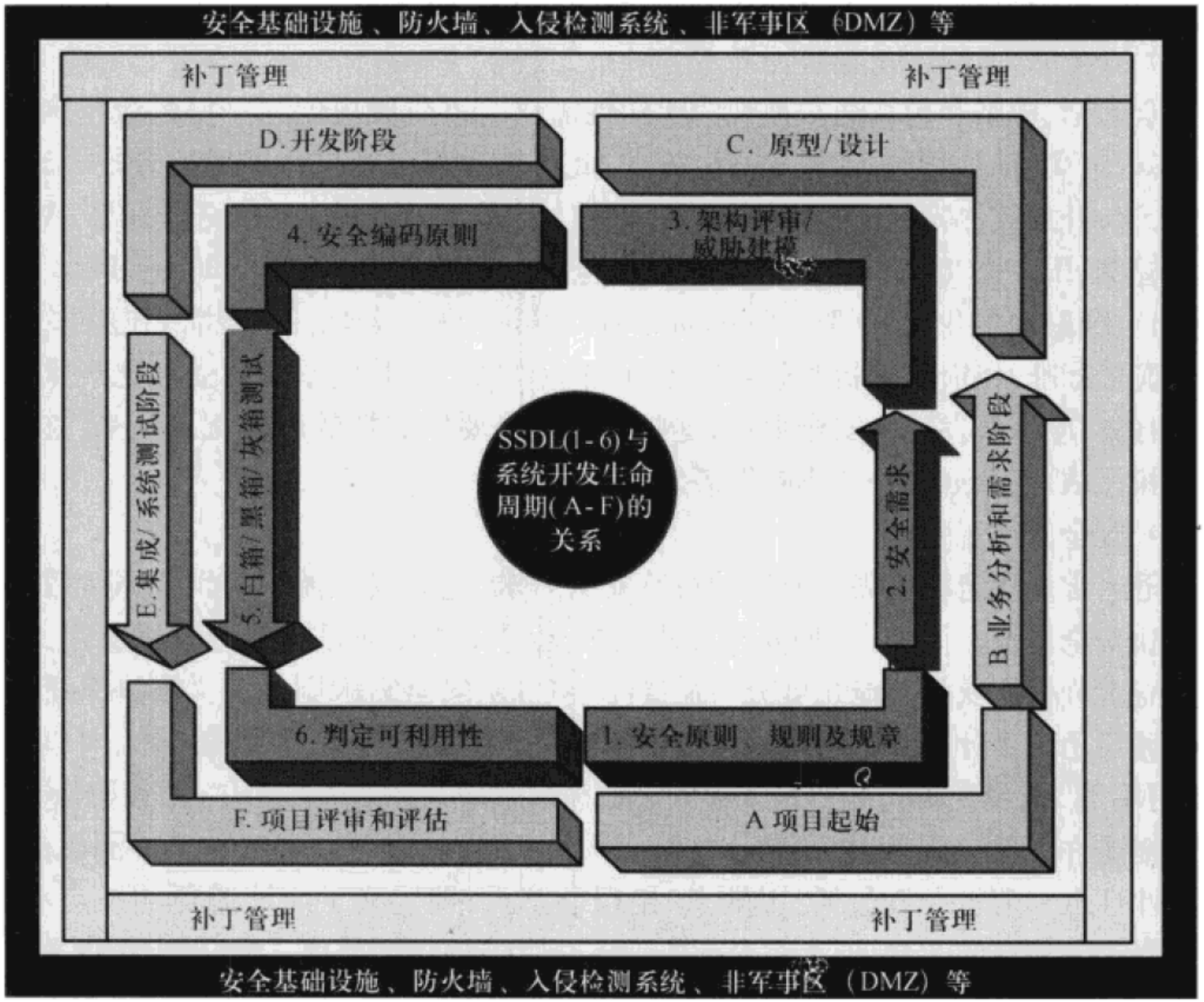
软件安全测试

- 包括白箱/黑箱/灰箱测试，软件渗透测试，基于风险的测试
- 判定漏洞的可利用性，即对测试出的安全漏洞或者在开发结束新公布的软件漏洞进行分析，判定这些漏洞是否可被攻击者利用，构成威胁

漏洞响应和产品的维护

- 漏洞一般在用户使用的过程中被发现，此时，迅速确认、响应、修复漏洞，是非常重要的
- 正常的漏洞响应步骤
  - 发现漏洞通知厂商
  - 确认漏洞和风险评估
  - 修复漏洞
  - 发布补丁及安全简报，对外公布安全补丁

传统软件开发生命周期与安全软件开发生命周期间的关系



大纲

- 软件开发生命周期概述
- 传统的软件开发生命周期
- 安全的软件开发生命周期
- ✓ 其他安全软件开发生命周期模型
- 软件保护技术

其他安全软件开发生命周期模型

- 微软可信计算安全开发生命周期
- 安全软件开发的小组软件过程
- 安全敏捷开发
- 软件可信成熟度模型
- 软件安全框架
- BSI成熟模型

微软可信计算安全开发生命周期

- 为了可以抵抗安全攻击的软件的开发，微软已经采纳了可信计算安全开发生命周期(Trustworthy Computing Security Development Lifecycle, SDL)这一个过程
- 在微软的软件开发的每一个过程的相应阶段中，SDL为其增加了一系列以安全为重点的活动和提交报告
- 微软通过利用安全衡量指标，以及微软核心安全团队的安全专业知识来对软件开发人员进行强制性的安全培训

SDL中新增加的安全活动

- 需求分析阶段：对安全功能的要求和可信行为的确切定义
- 设计阶段：对安全风险识别的威胁建模
- 代码实现阶段：静态分析、代码扫描工具和代码审核工具的使用；以安全为核心的测试，如Fuzzing
- 审查阶段：最终的代码审查和历史代码的审查
- 发布阶段：最后的安全检查有微软核心安全小组来完成。该小组由安全专家组成，在整个软件开发生命周期中都参与产品的开发

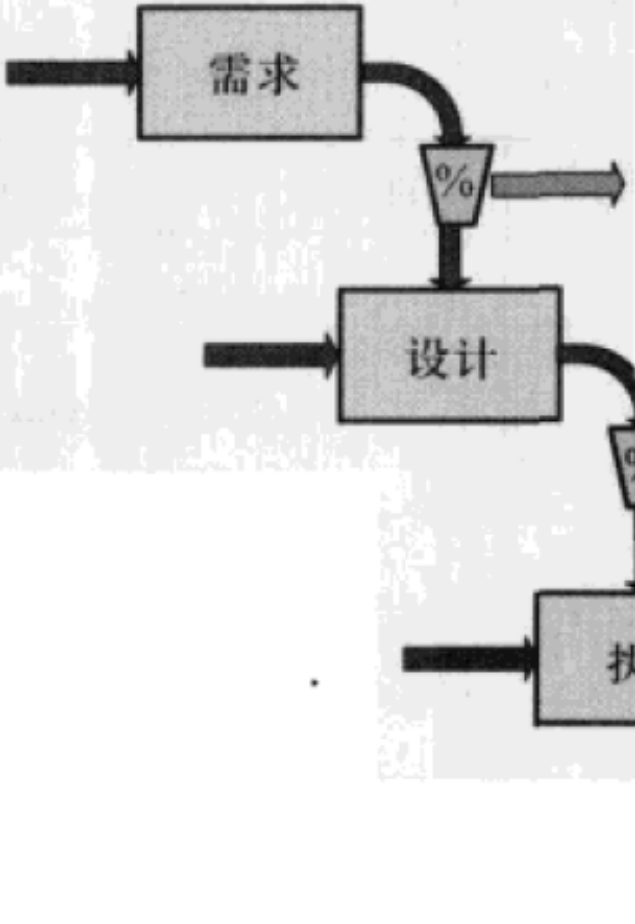
SDL的12个阶段

- Stage 0: 教育和意识
- Stage 1: 项目启动
- Stage 2: 定义和按照设计的最佳实践
- Stage 3: 产品风险评估
- Stage 4: 风险分析
- Stage 5: 创建安全文件，工具，以及为客户提供最佳实践
- Stage 6: 安全编码策略
- Stage 7: 安全测试策略
- Stage 8: 安全推动（Push）
- Stage 9: 最终安全审查
- Stage 10: 安全响应计划
- Stage 11: 产品发布
- Stage 12: 安全响应执行

安全软件开发的小组软件过程

- Team Software Process (TSP)为适用于团体和个人的软件工程提供了一个框架，它是一组进程的集合
- 通过TSP产生的软件比起根据现有方法产生的软件要少一个或者两个数量级的缺陷数目
- 安全软件开发TSP (TSP-Secure) 将TSP进一步扩展，它直接专注于软件应用的安全

去除漏洞过滤器



在需求分析、威胁建模或者开发滥用的过程中，一定百分比的那些在需求阶段注入的漏洞被移除

在设计审查和验证的过程中，一定百分比的在需求和设计阶段注入的漏洞被移除

那些在需求，设计和编码阶段注入的漏洞，以一定百分比的比例在代码审查，动态分析，静态分析和安全测试中，被移除

目的是使得能够导致漏洞的缺陷的剩余量达到最小

安全敏捷开发

- 敏捷宣言认为：通过敏捷开发，发现了更好的开发软件的方法，并已经开始在一些方面达到共识：如个体和交互胜过过程和工具；可以工作的软件胜过面面俱到的文档；客户合作胜过合同谈判；响应变化胜过遵循计划
- 个体的敏捷开发包括最为熟知的极限编程（Extreme Programming），Scrum，精益软件开发（Lean Software Development），水晶（Crystal）方法，特征驱动开发（Feature Driven Development），以及动态系统开发方法（Dynamic Systems Development Methodology）

软件可信成熟度模型

- 软件可信成熟度模型（Software Assurance Maturity Model, SAMM）
- 测试版本于2008年8月发布，官方1.0版本在2009年3月发布。这一模式的开发帮助组织制定和实施软件安全战略。这个模型旨在帮助
  - 一个组织对现有软件安全计划的评估
  - 使用反复的软件安全项目的开发
  - 改进的安全保障
  - 在组织内与安全有关活动的定义和测量

软件可信成熟度模型

- SAMM是一个开放的项目，提供免费可用非供应商特定的内容。该模型集中在四个包含在软件开发中的核心业务功能上
  - 监管：进程和活动与组织管理其软件开发的方法有关
  - 构造：进程和活动与组织定义其目标和在开发项目中软件创造的方法有关
  - 验证：进程和活动与组织验证和测试在软件开发创造的工件的方法有关
  - 部署：进程和活动与组织管理软件业务的发布创建一个运行环境有关

软件可信成熟度模型结构

- 在每个业务功能的具体实践领域列于下表

监管	构造	验证	部署
战略和度量	威胁评估	设计审查	弱点管理
政策和法规遵从	安全需求	代码审查	环境加固
教育与指导	安全构造	安全测试	操作使能

软件可信成熟度模型结构

- 一个在实践中获得认知的成熟层次结构如下
  - 成熟度等级0：在实际中，大部分活动都没有执行的起点
  - 成熟度等级1：实践领域的活动和进程理解为一个初步的程度，但实现是专案
  - 成熟度等级2：实践效率和/或有效性正在增加
  - 成熟度等级3：实践领域的活动和进程是全面的，表明该领域的全面掌握

软件能力成熟度模型（CMM）

- 1984年，美国国防部资助建立了卡内基·梅隆大学软件研究所（SEI）
- 1987年，SEI发布第一份技术报告介绍软件能力成熟度模型（CMM）及作为评价国防合同承包方过程成熟度的方法论
- 1991年，SEI发表1.0版软件CMM(SW-CMM)
- CMM自1987年开始实施认证，现已成为软件业权威的评估认证体系
- CMM包括5个等级，共计18个过程域，52个目标，300多个关键实践

CMM模型

- 能力成熟度模型，Capability Maturity Model for Software，缩写为SW-CMM，简称CMM
- 对于软件组织在定义、实施、度量、控制和改善其软件过程的实践中各个发展阶段的描述
- CMM的核心是把软件开发视为一个过程，并根据这一原则对软件开发和维护进行过程监控和研究，以使其更加科学化、标准化、使企业能够更好地实现商业目标

CMM模型

- 软件过程研究的是如何将人员、技术和工具等组织起来，通过有效的管理手段，提高软件生产的效率，保证软件产品的质量
- 软件过程的三个流派：CMU-SEI的CMM/PSP/TSP；ISO 9000质量标准体系；ISO/IEC 15504（SPICE）
- 学术界和工业界公认美国 Carnegie Mellon 大学软件工程研究所(CMU/SEI) 主持研究与开发的软件能力成熟度模型CMM是当前最好的软件过程，已成为业界事实上的软件过程的工业标准

软件过程的三个流派

- CMM/PSP/TSP即软件能力成熟度模型/ 个体软件过程/群组软件过程，是1987年美国 Carnegie Mellon 大学软件工程研究所(CMU/SEI)以W.S.Humphrey为首的研究组发表的研究成果"承制方软件工程能力的评估方法“
- ISO 9000质量标准体系是在70年代由欧洲首先采用的，其后在美国和世界其他地区也迅速地发展起来。目前，欧洲联合会积极促进软件质量的制度化，提出了如下ISO9000软件标准系列：ISO9001、ISO9000-3、ISO9004-2、ISO9004-4、ISO9002
- ISO/IEC 15504（SPICE）是1991年国际标准化组织采纳了一项动议，开展调查研究，按照CMU-SEI的基本思路，产生的技术报告ISO/IEC 15504--信息技术软件过程评估

CMM五个等级

能力等级	特点	关键过程
第一级 基本级	软件过程是混乱无序的,对过程几乎没有定义,成功依靠的是个人的才能和经验,管理方式属于反应式	
第二级 重复级	建立了基本的项目管理来跟踪进度.费用和功能特征,制定了必要的项目管理,能够利用以前类似的项目应用取得成功	需求管理,项目计划,项目跟踪和监控,软件子合同管理,软件配置管理,软件质量保障
第三级 确定级	已经将软件管理和过程文档化,标准化,同时综合成该组织的标准软件过程,所有的软件开发都使用该标准软件过程	组织过程定义,组织过程焦点,培训大纲,软机集成管理,软件产品工程,组织协调,专家审评
第四级 管理级	收集软件过程和产品质量的详细度量,对软件过程和产品质量有定量的理解和控制	定量的软件过程管理和产品质量管理
第五级 优化级	软件过程的量化反馈和新的思想和技术促进过程的不断改进	缺陷预防,过程变更管理和技术变更管理

软件安全框架

- Citigal和Fortify合作开发出的软件安全框架（Software Security Framework,SSF）
- SSF的结构初步建立在SAMM的内容上，然后根据在组织中的解决安全发展中的开发审查进行了调整
- SSF的作者明确提出了一个基于项目分析的BSIMM成熟模型（Building Security In Maturity Model）

软件安全框架SSF

监管	智能	SSDL 接触点	部署
策略和度量	攻击模型	架构分析	渗透测试
履约和策略	安全特征和设计	代码审计	软件环境
培训	标准和需求	安全测试	配置管理和漏洞管理

活动涉及到的所有SSF审核机构

是什么	干什么
建立组织支持	创建安全有关的布道角色/内部营销
建立统一的办法来解决管理客户需求	创建安全有关的政策
促进一个安全的组织文化	提供意识到安全的训练
描述组织的具体安全问题	创建/使用具体到公司历史的内容
通过一个产品的安全特性创造安全向导	建立和发布有关安全功能的详细资料(身份验证,角色管理,密钥管理,审计/日志,加密,协议)
建立安全体系的组织能力	有安全架构专家领导架构和产品功能的评论
站在攻击者的角度评价	将黑箱安全工具纳入到质量审核过程中
确定组织特定问题领域	利用外部专家应用渗透测试
确保软件开发和验证坚实安全基础设施	使用适当的主机/网络安全开发和测试

BSI成熟模型

- BSI成熟模型（BSIMM），是用来帮助理解和设计安全的软件
- BSIMM的目标是一个构建和不断发展软件安全行动的指南

大纲

- 软件开发生命周期概述
- 传统的软件开发生命周期
- 安全的软件开发生命周期
- 其他安全软件开发生命周期模型
- ✓ 软件保护技术

软件保护技术

- 软件保护技术概述
- 静态分析技术
- 动态分析技术
- 常用软件保护技术
- 软件加壳与脱壳
- 设计软件的一般性建议

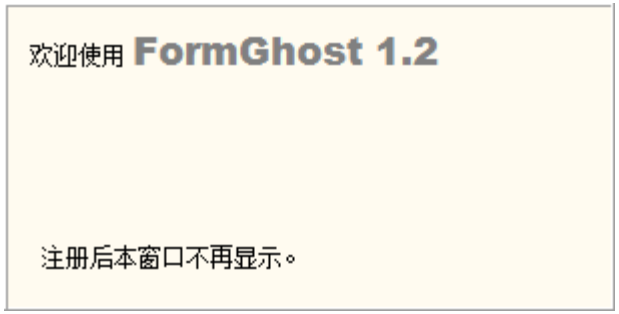
实验

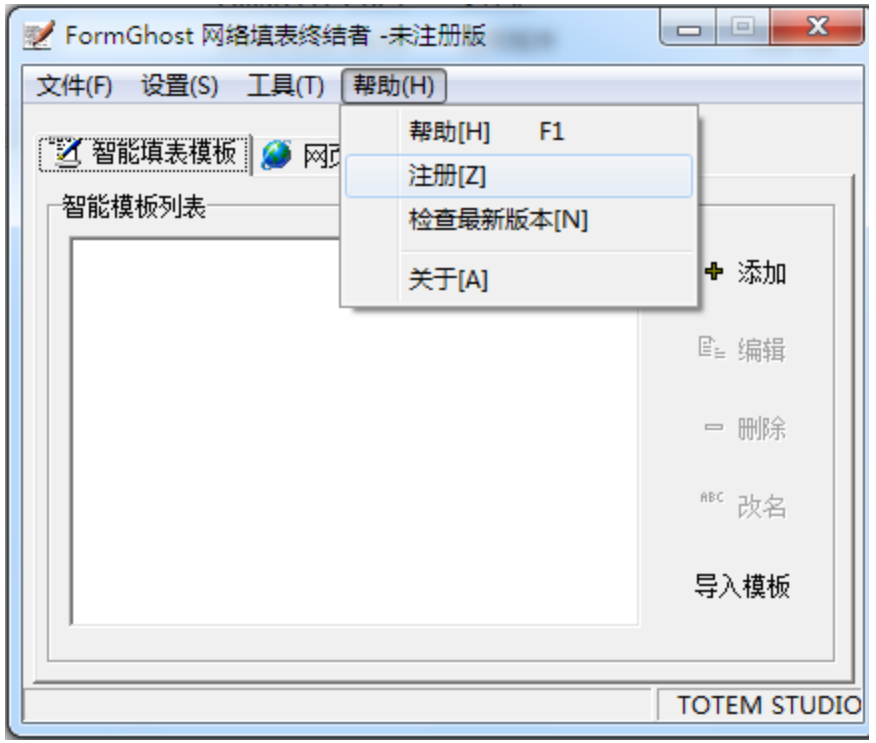
- 破解软件


FormGhost 网络填表终结者
文件(F) 设置(S) 工具(T) 帮助(H)
帮助[H] F1
注册[Z]
检查最新版本[N]
关于[A]
智能填表模板
网页
智能模板列表
添加
编辑
删除
改名
导入模板
图22
TOTEM STUDIO

实验一进一步

- 破解后，再次打开程序





- 如何去掉提示页？
- 如何自动注册？

小结

- 软件开发生命周期概述
- 传统的软件开发生命周期
- 安全的软件开发生命周期
- 其他安全软件开发生命周期模型
- 软件保护技术