Ch 1 **概述**(专业化软件开发、软件工程职业道德、案例\*4)

1. **软件失效**的因素：

①不断增长的系统复杂性②未有效采用软件工程方法

1. **软件**不仅包括程序本身，还包括所有是程序能够正常使用的相关文档、库、支持网站、配置数据等。
2. **好的软件的基本属性**：①向用户提供所需的功能与性能(可接受性)②可维护性、可依赖性、可用性、效率
3. **通用软件产品**和**定制化产品**的关键区别在于软件规格说明和软件变更决策由开发者自己确定还是客户给出。
4. **软件工程**是一个工程学科，涵盖了软件生产的各个方面，从最初的系统规格说明到运行和维护。**面临的挑战**：不断增长的多样性、缩短交付时间、开发可信软件要求。**重要性**：个人和社会more依赖先进的软件系统，就要求以经济且快速的方式开发出可靠可信的系统；长远看运用软件工程方法和技术开发专业化的软件系统比单纯作为个人编程项目编写程序cheaper，无法有效应用软件工程方法将导致更高的测试、质量保障和长期维护的成本。
5. **影响软件的一般问题**：

①异构性②企业和社会的变革③信息安全与信任④规模

1. **应用的类型**：独立的应用；基于事务的交互式应用；嵌入式控制系统；批处理~；娱乐~；建模和仿真~；建模和仿真~；数据收集和分析~；系统之系统
2. **软件工程基础适用于所有类型的软件系统**：

软件系统开发过程应当是受管理的并被开发人员所理解；可靠性和性能对于所有类型的系统来说都很重要；理解和管理系统规格说明和需求是很重要的；应该尽可能高效地使用已有地资源，在一些适当的地方复用软件。

1. 软件复用已成为构建基于Web的系统的主流方法；~总是增量开发和交付的；~可使用面向服务的软件工程来实现，软件构件是独立的Web服务；创建富客户端界面
2. 保密；诚实陈述工作能力；知识产权；计算机滥用

Ch 2 **软件过程**(Software Processes)

1. Process descriptions:产品交付物；角色；前置后置条件
2. **瀑布模型**：包含规格说明、设计、实现、测试及维护这几个离散的开发阶段。原则上一个阶段必须在进入下一个阶段之前完成。在实践中，阶段之间存在显著迭代。

**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;运行和维护. **缺点**:很难容纳变化

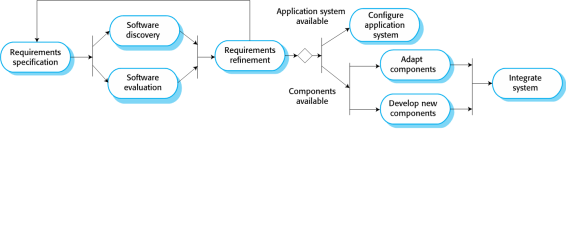
**适用**:嵌入式系统(硬件)\关键性系统(安全)\大型软件系统

1. **增量式开发**使得规格说明、开发和确认活动交错进行，系统开发体现为一系列版本，每一版本在前一版基础上加一些功能，软件以增量的方式进行交付和部署.

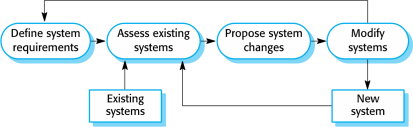
**优势**:降低了实现需求变更的成本,重新分析和修改文档的工作量要少很多;在开发过程中更容易得到客户对于已完成的开发工作的反馈意见;及时并未将所有的功能包含其中,也使得在早期向客户交付和部署有用的软件成为可能,客户可以更早地使用软件并从中获得价值.

**问题**:过程不可见,管理人员需要常规的交付物来掌握进度,系统快速开发,产生系统每个版本的文档就很不划算;伴随着新的增量的增加,系统结构会逐渐退化,代码也凌乱,向系统中添加新特性变得越来越困难且昂贵.

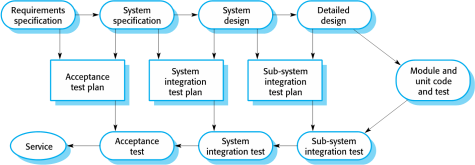
1. **集成和配置**依赖于可复用的构件或系统.关注在新的使用环境中配置这些构件并将它们集成为一个系统.
2. 3类软件构件常被复用:经过配置后可以在特定环境中使用的独立应用系统;作为一个构件或一个包开发的并且将与一个构建框架相集成的一组对象;按照服务标准并且可以通过互联网进行远程调用的Web服务
3. **面向复用的软件工程**:



1. **优劣势**:降低软件开发量,降低成本和风险,实现更快软件交付,但需求权衡是不可避免的,这导致系统不完全满足用户的真实需求,还可能失去一些对系统演化的控制.
2. **软件规格说明**:目的是理解和定义系统需要提供那些服务以及识别对于系统开发和运行的约束.**需求工程活动**需求抽取与分析(可能会开发系统模型和原型),需求规格说明(用户与系统需求),需求确认(纠错修改需求文档)
3. **软件设计和实现**:对将要实现的软件的结构\系统所使用的数据模型和结构\算法\系统构件间的接口的描述;将需求规格说明开发为一个可执行的系统以交付给客户.
4. **设计输入**:平台(OS\DB\中间件\其他应用系统:软件将会在其中运行的环境)信息,软件需求,数据描述;**设计活动**:体系结构设计,数据库~,接口~,构件选取和~;**设计输出**:系统体系结构,数据库设计,接口规格说明,构件描述
5. **软件确认**:验证和确认目的是确定系统是否符合它的规格说明,同时是否符合系统客户的期望.**程序测试**用模拟的测试数据运行系统,是最基本的确认技术.
6. Testing stages:构件测试(函数/对象类/实体构成的内聚的分组单独测试),系统~(构件间交互/接口/需求),客户~
7. **软件演化(维护)**:修改已有的软件系统满足新的需求

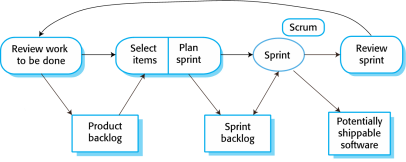


1. **计划驱动的软件过程中的测试阶段**(V-model):



1. 过程应包含应对变化的活动.系统需求随着业务应对外部牙利\竞争和管理优先级的需要等变化而发生改变;新技术出现使新的设计和实现方法成为可能;平台改变APP改变
2. 变化意味着已经完成的工作必须重做(**返工**),需求重新分析,实现新功能,都是成本.**降低成本**:变化预测;变化容忍
3. 原型prototyping:系统或系统的一部分的早期版本被快速开发以检验客户需求以及设计决策的可行性,用于演示概念\尝试候选设计方案\更好地理解问题以及可能的解决方案. 在需求工程过程中,帮助对系统需求进行抽取和确认;在系统设计过程中,探索软件解决方案和系统用户界面的开发;测试过程中,run back-to-back tests. **优点**:改进系统的可用性;更贴合用户真实需求;改善设计质量;提高可维护性;减少开发工作.**原型开发的过程**:定义原型目标(原型计划),定义原型功能(概要设计),开发原型(可运行原型),评估原型(评估报告)
4. **增量式交付**:根据服务优先级将服务分配到各个增量,优先级最高的首先被实现和交付并部署到客户的工作环境供试验,还可以改进后续增量的需求. 开发过程需求不能改. **优点**:客户可以将早期的增量作为原型使用,从中获得关于后续系统增量需求的经验;客户不用等到整个系统交付就能从系统中获得价值;变更容易加进来;整个项目失败风险降低;最重要的系统服务测试最充分,这部分不太可能失效. **问题**:当**新系统**准备**替换**一个已有系统时,迭代化交付会有问题;大多数系统需要一组由系统的不同部分使用的基础设施;迭代化过程的本质是规格说明与软件一起开发,这与组织的采购方式冲突,系统开发**合约**会要求完整的系统规格说明.
5. **过程改进**:意味着理解当前的过程,并对其进行改进以提升软件质量、降低成本或加快开发过程的一种方式.2种方法
6. **过程成熟度方法**:关注改进过程和项目管理,并将好的软件工程实践引入组织中. **过程度量**(对软件过程或产品的属性);**过程分析**(识别过程中的弱点和瓶颈),**过程改变**(提出改变方法,引入变更,改进后再度量看是否改善,由此循环过程)
7. **敏捷方法**:软件按照增量进行开发和交付,过程文档和管理开销被最小化,开发关注代码本身,而不是支持文档. **主要特点**是快速交付功能以及对客户需求变更的快速响应.
8. **The SEI capability maturity model**:初始级(uncontroll),受管理级,已定义级(关注组织的标准化),量化管理级,优化级

Ch 3 Agile SD**敏捷软件开发**(敏捷方法\技术\项目管理\伸缩)

1. **敏捷方法的共性**:规格说明\设计和实现过程交织在一起;系统按照一些列增量进行开发,利益相关者参与每个增量的规格说明和评估;新版本快速交付以评估;使用广泛的工具来支持开发过程;最小化文档:关注代码
2. **敏捷宣言manifesto**:概括了敏捷软件开发方法背后的基本思想的一组原则. 个体和交互胜过过程和工具;可工作的软件>全面的文档;客户协作>合同谈判;响应变化>遵循计划.
3. **敏捷方法的原则principles**:客户参与\拥抱变化\增量交付\保持简洁\人而不是过程
4. **适用于**软件企业开发的用于市场销售的中小规模产品\组织内的定制化系统开发
5. **极限编程**:**极限**在于多个新版本在一天之中开发;增量每两周交付给客户;每个build后必测试，测试成功接受该生成。
6. **XP发布周期**:环(选择本次发布的**用户故事**,将用户故事分解为任务;规划发布;开发\集成\**测试**;发布软件;评估系统)
7. **XP实践**:共同拥有权\**持续集成**\现场客户\**结对编程**\简单设计\重构\小的发布\可持续的步调\测试先行的开发
8. **用户故事**:解释一个软件或系统如何被使用的情形以及关于系统之间可能发生的交互的一种自然语言描述.
9. **重构**:修改一个程序以改进它的结构和可读性,同时不改变其功能.(去除重复代码\对属性和方法进行整理和重命名\用对定义的方法的调用来代替相似的代码段)
10. 增量开发和计划驱动的一个重要区别是系统测试方式.
11. **XP testing features**:测试先行的开发;基于场景的增量测试开发,用户参与测试开发和确认,使用自动化测试框架
12. **测试先行的开发**:可执行的测试是在程序代码之前编写的,每次对程序进行修改后都会自动执行这一组测试.**存在问题**:程序员更喜欢编程而不是测试,有时会在编写测试走捷径;有些测试很难增量地编写;判断一组测试的完备性很难.
13. **结对编程**:程序员两两配对工作而不是独立工作来开发代码的一种开发情形.**好处**:支持对于系统的共同所有权和共同责任的思想;扮演了非正式的评审过程的角色,∵每一行代码都由至少两个人看着;鼓励通过重构改进软件的结构;知识共享很重要,∵降低由团队成员离开而带来的项目风险.
14. **敏捷项目管理**:Scrum是基于冲刺sprint(短开发迭代周期)的一种敏捷开发方法.可以和其他敏捷方法一起用作敏捷项目管理的基础. **3阶段**:初始阶段:大纲规划,确立总目标并设计软件体系结构;一系列冲刺周期;结束项目,完成所需文档.
15. **Scrum benefits**:产品被分解为一组可管理\可理解\利益相关者可以对应上的条块;不稳定的需求不会影响进度;整个团队都对所有的一切保持可见,提升团队交流沟通和士气;客户按时看到增量的交付并获得产品如何工作的反馈;客户和开发者建立了信任与积极的文化,每个人都期望项目成功
16. Scrum sprint cycle
17. Scaling agile methods**敏捷方法的伸缩**:**规模化scaling up**以处理大系统的开发,应用范围从专业的开发团队**扩展到~out**在一个有着多年软件开发经验的大企业内更加广泛地使用.**问题**:非正式性与大型企业基于法律的合同定义不相符;最适合于新的软件开发而不是软件维护(缺少产品文档\保持客户参与\开发团队的延续性),大型企业软件成本大多在维护;适用于小的\同处一地的团队.
18. 规模化和扩展敏捷方法的根本要求是与计划驱动的方法集成, Agile&plan-based因素:系统:规模\类型(复杂性\实时)\生命周期\外部监管;团队:人\开发技术\组织分布\竞争力;组织:合同\交付\文化
19. 系统之系统;棕地系统(关注与现有系统的交互,很难灵活);集成创建新系统关注配置而不是代码开发;受外部规则和监管制约;采购和开发时间长;各样利益相关者→IBM敏捷规模化模型ASM
20. **规模化的敏捷方法共性**:完全增量的需求工程方法是不可能的;没有哪个人能单独作为产品负责人或客户代表;不能只关注系统代码;必须设计和实施整个团队的沟通机制;集成时持续集成无法实现,保持频繁系统构建和常规的系统发布是重要的.多团队Scrum:角色重复(产品负责人和Scrum主管),产品架构师,发布同步
21. **面向组织扩展**:难:项目经理没有敏捷方法经验有风险,官僚主义有质量规程和标准,团队人员技术参差不齐,文化上的抵制

Ch 4 需求工程RE

1. 对于一个系统的**需求**是关于该系统应当提供的服务以及对其运行的约束的描述.**用户需求**是高层的抽象陈述(自然语言和图形),而**系统需求**是关于系统功能的详细和正式的定义(结构化文档). 找出\分析\文档化并且检查这些服务和约束的过程被称为RE.
2. **功能性需求**(取决于软件类型\期望用户\书写需求方法):对系统应该提供的服务或功能的陈述(一致\完整).**非功能性需求**:对系统的约束或期望行为的陈述.约束可针对正在开发的软件的涌现特性(可靠性\响应时间\存储使用)或针对开发过程(IDE\编程语言).分类:产品~,组织~,外部~. 实现跨越整个系统∵影响一个系统的体系结构;会产生多个相互联系的功能性需求. 应该’可测试的’
3. **需求抽取**:通过和利益相关者交互发现需求(需求发现和理解,~分类和组织,~优先级排序和协商,~文档化,循环).**问题**:利益相关者不知道想从系统得到什么,用他们工作的知识表达需求,需求冲突,组织和政策因素影响,需求(可变)所处经济和业务环境是动态的.
4. **访谈**:开发者与系统利益者谈论他们做的事(封闭式\开放式),虚心倾听,避免预设,用跳板性的问题\需求建议\原型等提示受访人
5. **ethnography**(**人种学调查**):观察人们做自己的工作来了解他们使用哪些制品&如何使用.从实际工作方式\与他人的合作\对他人活动的了解中得出的需求.不总是助于创新.
6. **故事和场景**:关于系统使用的高层描述;场景是对于系统的典型使用方式或者用户执行某些活动的典型方式一种描述.(结构化故事,最初假设,常规事件流,可能问题,其他活动,结束时的系统状态)
7. **需求规格说明**:在需求文档中正式撰写用户和系统需求的过程.
8. **书写系统需求的方法**:自然语言,结构化自然语言,图形化表示,数学规格说明;可以作为系统实现的合约的一部分
9. **需求和设计无法分离**:可能要设计一个初始的系统体系结构来帮助组织需求规格说明/满足非功能性需求;系统须与现有系统互操作,对设计构成约束并提出新需求.
10. **使用自然语言进行需求规格说明的问题**:表述**不清楚**,功能和非功能**需求混合**在一起,多个需求混合到一条语句中表述
11. Use case用况(UML的一种):是一种使用图形化模型和结构化文本描述用户和系统间交互的方式.
12. 软件需求文档:是关于系统开发者应当实现的所有东西的正式陈述. **用户**:系统客户\管理人员\系统工程师\测试~\维护~
13. **需求确认**:检查需求是否定义了客户真正想要的系统的过程.**检查**:正确性\一致性\完整性\现实性(预算)\可验证性(可测试),**技术**:需求评审(-\来源\变更不影响其他需求),原型化,测试用例生成
14. **需求变更**:由于业务和技术环境发生改变/为系统付钱的人和系统用户不是同一批,利益相关者需求冲突,优先级矛盾,要折中.
15. **需求管理**:管理对需求的变更以保证所做的变更是经过适当的分析并在系统中保持追踪的过程. 维护需求间依赖关系连接,这样就可以评估需求变更的影响. **decisions**:需求标识,变更管理过程(评估变更影响),追踪关系策略,工具支持(需求存储\-\-管理).
16. **需求变更管理**:问题分析和变更规格说明(变更请求合理或撤回),变更分析和成本考虑(评估成本),变更实施(模块化,可修改性).

Ch 5 系统建模

1. **系统建模**modeling:建立系统**抽象**模型的过程,其中每一个模型表示系统的一个不同的**视角**或观点.基于UML的图形化表示系统
2. **System perspectives**:外部视角(系统的上下文或环境),交互~(系统与环境或系统构件之间的交互),结构化~(组织或处理的数据结构),行为~(动态行为以及如何响应事件)
3. **活动图**描述一个过程或数据处理中所包含的活动,**用况图**描述一个系统与其环境之间的交互,**顺序图**描述参与者与系统之间以及系统构件之间的交互,**类图**描述系统中对象类以及这些类之间的关系,**状态图**描述系统如何对内部和外部的实践做出响应.
4. **图形化模型的使用方式**:推动关于现有或所设想的系统的讨论,使讨论聚焦;文档化现有系统;用于生成系统实现的详细系统描述
5. **上下文模型**:描述一个所建模的系统是如何被置于包含其他系统和过程的环境之中,帮助定义待开发系统的**边界**(哪些属于/不属于系统).可以与**业务过程模型**(UML活动图显示)一起使用．
6. **交互模型**:用户交互建模帮助识别用户需求,建模系统间交互突出可能出现的通信问题,建模构建交互帮助理解提出的系统结构是否能实现所要求的系统性能和可依赖性.
7. **用况建模**:每个用况表示一个包含与系统的外部交互的离线任务.**顺序图**:显示在一个特定的用况或用况实例执行过程中发生的交互序列.
8. **结构模型**:按照构成系统的构件以及它们之间的关系显示系统的组织,是描述系统设计组织的静态模型或系统执行时的组织的动态模型.**类图**,**泛化**(△类继承),**聚集**Aggregation(⋄一个对象(整体)由其他对象(部分)组成)
9. **行为模型**:关于系统在运行时的动态行为的模型.**数据驱动的模型**:业务系统.描述处理输入数据以及生成相关的输出过程中所涉及的动作序列. **数据流图**(用活动图表示),**顺序图**. **事件驱动的模型**:实时系统具有有限的状态.描述系统如何对外部和内部事件做出响应. **状态图**.
10. **模型驱动的体系结构**MDA:在构建一组系统模型的基础上开发软件的方法,这些模型可以被自动或半自动化地处理以生成一个可执行的系统.**计算无关模型**CIM对系统中使用的重要领域抽象进行建模,**平台无关模型**PIM在不涉及实现的情况下对系统的运转进行建模;**平台相关模型**PSM对PIM转换得到,每个应用平台都有一个单独PSM,存在多层次,每个层次增加一些平台相关的细节. **Cons**:对讨论有用的抽象对实现而言并不总是正确的抽象;对于复杂系统,实现不是主要问题,RE\信息安全可依赖性\遗留系统集成\测试更重要;追求平台无关性只对大型\长生命周期的系统有意义;平台的模型转换器成本可能超过自动生成代码的节省成本.

Ch 6 体系结构设计

1. **体系结构设计**是设计和RE之间的关键性衔接环节,∵它会确定组成一个系统的主要结构构件以及它们之间的关系.输出是一个**体系结构模型**,描述系统如何被组织.
2. 两个抽象层次:**小体系结构**关注单个程序的体系结构;**大体系结构**关注包括其他系统\程序和程序构件的复杂企业系统体系结构.
3. explicit architecture**优点**:聚焦利益相关者的讨论,系统分析:影响非功能性特性;大范围复用(相似需求的系统有相似的体系结构).用简单**框图**表示系统结构的高层样貌.
4. Use of体系结构模型:鼓励对系统设计进行讨论的方式;文档化已经设计好的体系结构的方式.
5. **体系结构设计决策**:问题:是否可复用?系统将如何分布到各个硬件核或者处理器上?体系结构模式或风格?什么策略来控制系统中构件的运行?系统的体系结构如何文档化?什么样体系结构组织最佳实现系统非功能性需求?系统结构构件如何分解为子构件?什么方法组织系统?
6. 体系结构风格&结构由非功能性需求来决定:性能(关键操作集中部署到一台PC,用大构件减少通信)\信息安全(层次化结构)\安全性(集中在单个构件)\可用性(包含冗余构件)\可维护性(容易改变的细粒度\自包含的构件)
7. **体系结构视图**Views:4+1软件体系结构视图模型:共同的用况或场景+逻辑\进程\开发\物理视图

