第一章 软件的本质

1.1 软件的本质

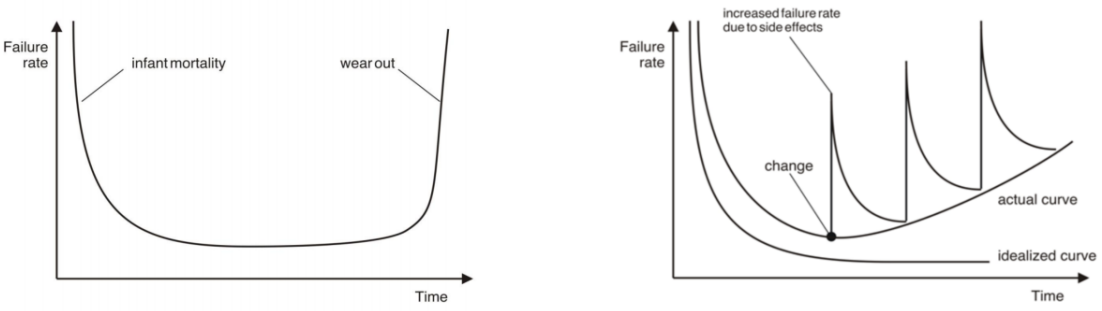
1.1.1 定义软件

1. 指令的集合(计算机程序)，通过执行这些指令可以满足预期的特性、功能和性能需求。(programs)
2. 数据结构，使得程序可以合理利用信息（data）
3. 软件描述信息（document）

软件硬件区别：

1. 软件是被开发的或者设计的，它不是经典意义上被制造出来的
2. 随着时间的推移，因为尘土、振动、使用不当、温度超限制、以及其他一些环境问题，硬件效率会逐渐降低，即硬件会**磨损**

   而软件不会磨损，但会退化，由于软件生命周期，会面临许多变更，每次变更都可能引入新的错误，使得是失效率像实际曲线那样陡然上升。可以说，**不断地变更时软件退化的根本原因。**
3. 尽管该行业正在转向基于组件的构建，但大多数软件仍然是**定制的（custom-built）**。



Wear Out vs. Deterioration (退化)

1.1.2 软件应用

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- Web/Mobile applications
- AI software (robotics, neural nets, game playing)

1.1.3 遗留软件(legacy software)

各种久远的旧的软件系统称为**遗留软件。**

问题：质量差

遗留系统进化的原因(Why must software change ?):

- Software must be **adapted to** meet the needs of new computing environments or technology.

- Software must be **enhanced** to implement new business requirements.
- Software must be **extended** to make it interoperable with other more modern systems

or databases.（软件必须扩展使之具有与更多新的系统和数据库的互操作能力）

- Software must be **re-architected** to make it viable within a network environment**.**(软件架构必须进行改建使之能适应不断演化的计算环境)

第二章 软件工程

2.1 软件工程的定义

1. 将**系统化的(systematic)、规范的(disciplined)、可量化(quantifiable)**的方法应用于软件的**开发(development)、运行（operation）和维(maintenance)**，即将工程化的方法应用于软件。
2. 对1中所述方法的研究。

软件工程是一种层次化的技术，从上到下依次是

- **tools(工具)**：为Process过程和Method方法提供支持
- **methods(方法)**：软件工程方法为构建软件提供技术上的解决办法
- **process(过程)**：
  - Is the foundation of SW engineering to hold technology together to build software timely and rationally （及时地、合理地）
  - Defines a framework as a road map for software development
  - Forms a basis for software management control
- **a quality focus(质量关注点)**：
  - 支持软件工程的根基(bedrock)在于质量关注点。
  - Is a organizational commitment to software
  - Need a continuous process improvement

2.2 软件过程

定义：`软件过程`是工作产品构建是所执行的一系列**活动(activity)**、**动作(action)和任务(task)**的集合。

- 活动主要实现宽泛的目标，(如与利益相关者进行沟通)，与应用领域、项目大小、结果复杂性没有直接关系
- 动作(如体系结构设计)包含了主要工作产品生产过程的一系列任务
- 任务关注小而明确的目标，能够生产实际产品(如构建一个单元测试)

2.2.1 过程框架

过程框架定义了若干**框架活动**，为实现完整的软件工程过程建立了基础。这些活动广泛地适用于所有软件项目开发，无论项目的规模和复杂度如何。此外，过程框架还包含了整个软件过程的**普适性活动(umbrella activity)。**一个通用的软件工程过程框架通常包含一下5个活动：

- **沟通(communication)**
- **策划(planning)**
- **建模(modeling)**
- **构建(construction)**
- **部署(deployment)**

2.2.2 普适性活动

贯穿项目始终，以帮助软件团队管理和控制项目进度、质量、变更和风险。

- 软件项目跟踪和控制
- 风险管理
- 软件质量保障
- 技术评审
- 测量
- 软件配置管理
- 可复用管理
- 工作产品的准备和生产

2.3 软件工程实践

2.3.1 实践的精髓

- 理解问题(沟通和分析)
    - W**ho has a stake in the solution to the problem?**： That is, who are the stakeholders(利益相关者)?
    - **What are the unknowns? (哪些是未知的)**：What data, functions, and features are required to properly solve the problem?
    - **Can the problem be compartmentalized?(问题可以划分吗)**： Is it possible to represent smaller problems that may be easier to understand?（问题细化，区分）
    - **Can the problem be represented graphically? (问题可以图形化描述吗)** ： Can an analysis model be created?(可视化、图形化)
- 策划解决方案(建模和软件设计)、
    - **Have you seen similar problems before?**

      Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
    - **Has a similar problem been solved?**
      If so, are elements of the solution reusable?
    - **Can subproblems be defined(可以定义子问题吗)?**
      If so, are solutions readily apparent for the subproblems?
    - **Can you represent a solution in a manner that leads to effective implementation? (能用一种很快实现的方式来描述解决方案吗？)**
      Can a design model be created?

- 实施计划(代码生成)
    - **Does the solution conform to the plan?(解决方案和计划一致吗)** Is source code traceable to the design model?
    - **Is each component part of the solution provably correct?(解决方案的每个组成部分是否可以证明正确？)** Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?
- 检查结果正确性(测试和质量保障)
    - **Is it possible to test each component part of the solution?(能否测试解决方案的每一个部分？)** Has a reasonable testing strategy been implemented?

- **Does the solution produce results that conform to the data, functions, and features that are required?(解决方案是否产生了与所要求的数据、功能、和特性一致性的结果？)**Has the software been validated against all stakeholder requirements?（满足需求）

2.3.2 通用原则(Hooker’s General Principles)

1. The Reason It All Exists(软件是否存在价值)
2. KISS (Keep It Simple, Stupid!)（简单）
3. Maintain the Vision（保持愿景，目标明确）
4. What You Produce, Others Will Consume （关注使用者）
5. Be Open to the Future（面向未来）
6. Plan Ahead for Reuse（重用）
7. Think!（认真思考）

2.4 软件开发神话

- 管理神话(management myths)
- 客户神话(customer myths)
- 从业者神话(practitioner's myths)

第三章 软件过程结构

3.1 通用过程模型

过程流：过程流描述了在执行顺序和执行时间上，如何组织框架中的活动、动作和任务。

- 线性过程流：从沟通到部署顺序执行五个框架活动
- 迭代过程流：在执行下一个活动前**重复执行**之前的一个或多个活动
- 演化过程流：采用循环的方式执行各个活动，每次循环都能产生更为完善的软件版本
- 并行过程流：将一个或者多个活动与其他活动并行执行。

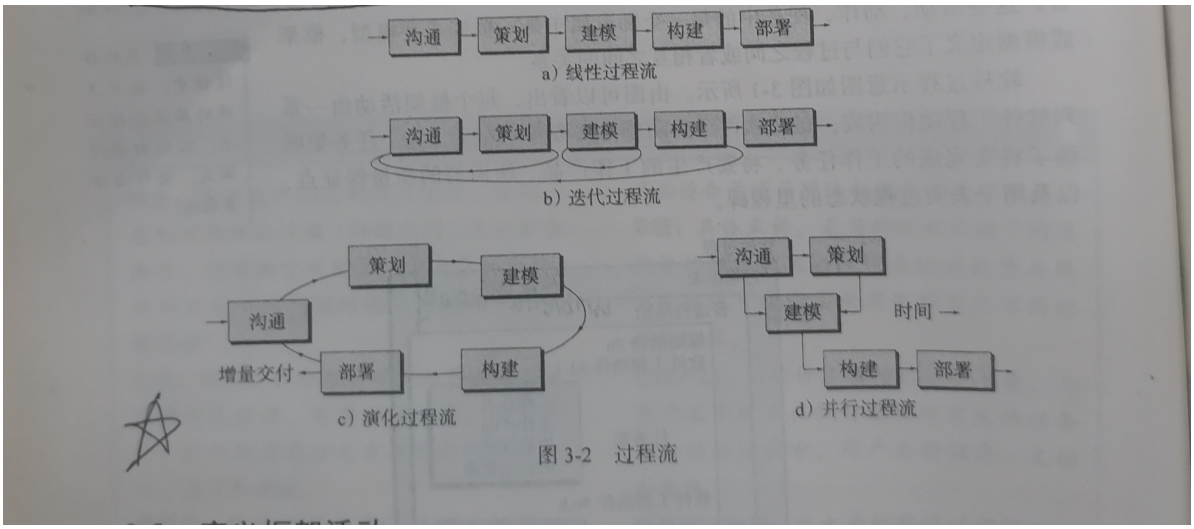


图 3-2 过程流

3.3 明确任务集

任务集定义了为完成软件工程操作的目标而要完成的实际工作。

每个任务集有以下构成:

- 软件工程工作任务
- 相关工作产品
- 质量保证点和项目里程碑

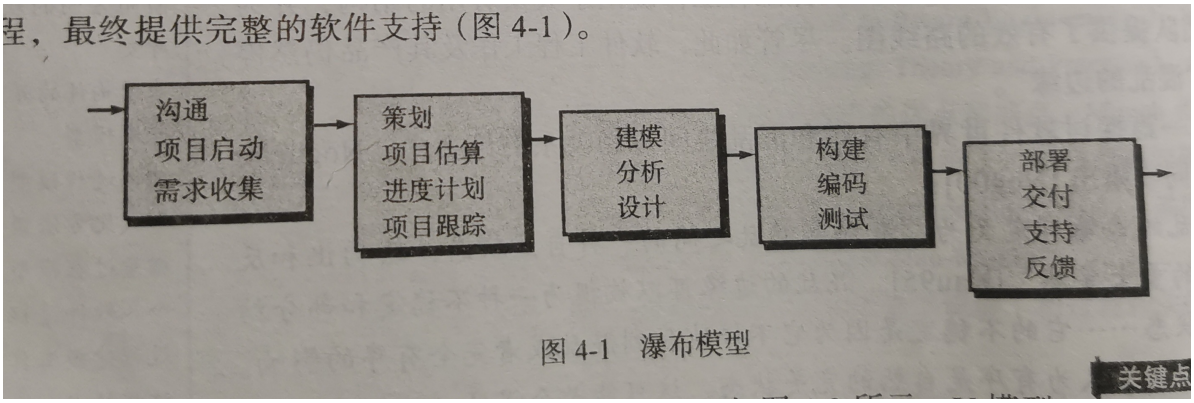
第四章 过程模型

4.1惯用过程模型(Prescriptive process models)

惯用过程模型力求达到软件开发的结构和秩序，其活动和任务都是按照过程的特定指引顺序进行的。是所有的软件过程模型都支持通用活动框架，只是每个模型对框架活动有不同侧重，并且定义不同的过程流以不同的方式执行每一个框架活动(包括动作和任务)

4.1.1 瀑布模型(Waterfall Model)

采用线性过程流，要求**需求必须是准确定义的和相对稳定的。**



瀑布模型又被称为**经典生命周期模型(classic life cycle)**,它提出了一个系统的、顺序的软件开发方法，从用户需求规格说明开始，通过策划、建模、构建和部署的过程，最终提供完整的软件支持。

优点:

1. 在用户需求被很好定义的时候是一个有用并且相当稳定的过程模型。
2. 很好理解和策划
3. 它适用于易于理解的小型项目。
4. 分析和测试非常简单。

缺点:

1. 实际项目很少遵从顺序流，并且变更很容易造成混乱
2. 客户通常很难清楚地描述其需求
3. 客户必须要很有耐心，因为只有在项目接近尾声的时候，他们才可能得到可执行的程序。
4. 在审查工作计划之前，可能无法发现重大错误。
5. 由于任务之间的依赖性，导致花费在等待上的时间可能超过花费在工作中的时间。

4.1.2 增量过程模型(The Incremental Model)

许多情况下，软件需求有明确定义，但是整个开发过程不宜单纯采用线性模型，同时，可能迫切需要为用户迅速提供一套功能有限的软件产品，然后在后续版本中不断细化和扩展功能，这就是增量过程模型。

优点:

- Useful for unavailable staffing for complete，implementation by the business deadline.（克服人手不足）
- 节约时间.
- 线性（每个增量按照瀑布模型进行管理）+**并行**
- 每个增量都是可提交运行的版本，（**第一个增量往往是核心产品**core product）

缺点:

- 对于每次提交新的增量，早期的增量往往容易被忽略。

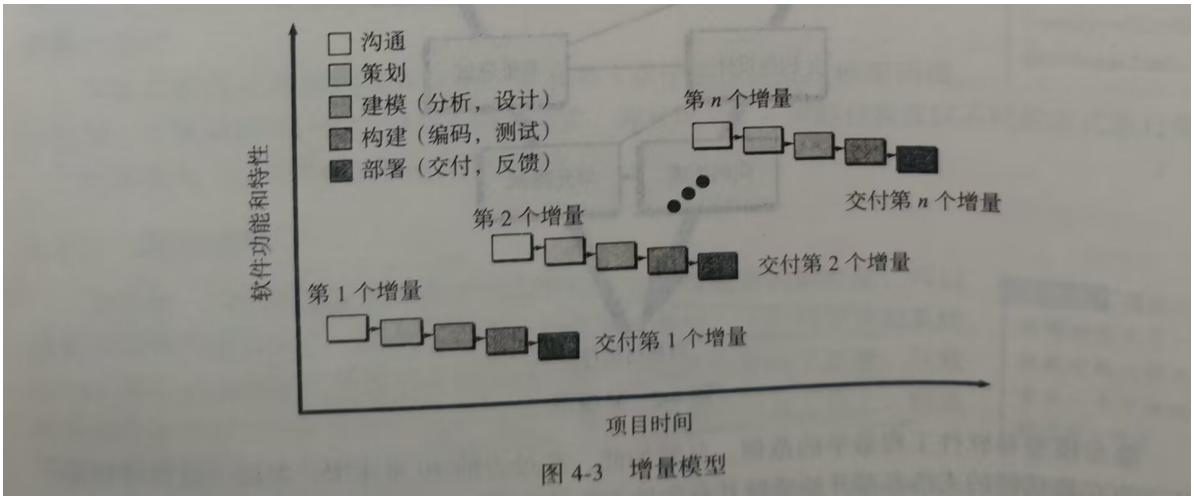


图 4-3 增量模型

4.1.3 演化过程模型（Evolutionary Models）

1.**原型过程模型(prototyping)**

虽然原型模型可以作为单独的过程模型，更多的还是作为一种方法，可以在任何一种过程模型中出现。原型模型开始于**沟通**。不论人们以什么方式运用它，当需求很模糊时，原型开发模型都能帮助开发人员和利益相关者更好地理解究竟需要做什么。（原型过程模型开发是为了尽快完成软件，开发者并没有考虑软件整体软件质量和长期可维护性，而只是将其快速开发可用。）

优点:

- 当**需求很模糊时**，原型开发模型都能帮助开发人员和利益相关者更好地理解究竟需要做什么。

缺点:

- 原型常常会被抛弃，而不是在其上继续开发,增加成本
- 客户通常会把原型当作工作产品，没有意识到**原型缺乏考虑整体质量和长期可维护性**，不愿重构整个系统，从而导致软件开发管理失效。()
- 一些凑合的技术和算法可能会遗留在最终系统中

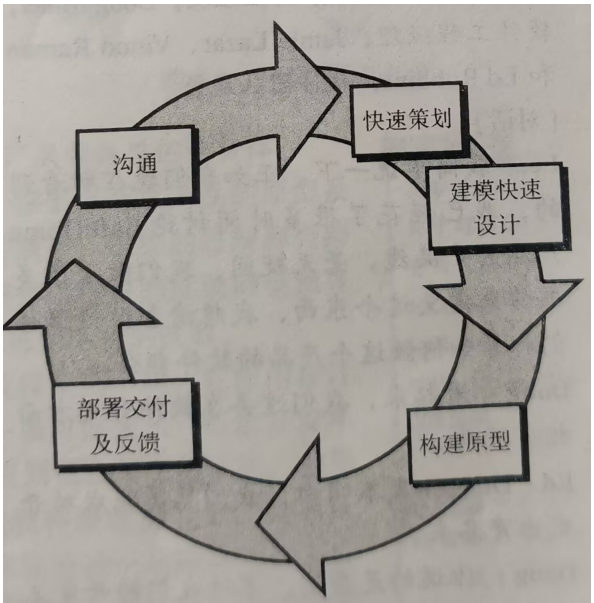


图 4-4　原型开发范型

2.螺旋模型(Spiral Model)

螺旋模型是一种演化过程模型，它结合了**原型的迭代性质**和**瀑布模型的系统性和可控性**，注重**风险控制（里程碑）**，**适合大型项目开发**(这是因为其他过程模型在软件交付后就结束了，螺旋模型则不同，它应用与计算机软甲的整个生命周期)。运用螺旋模型，把软件开发为一系列演进版本，早期的迭代中，软件可能是一个理论模型或原型，后来的迭代，才会产生一系列逐渐完善的系统版本
在每一次迭代的过程中，都要考虑风险(a risk-driven process model)、标记里程碑

螺旋模型沿着螺线旋转，自内向外每旋转一圈便开发出一个更完善的版本。

如果发现风险太大，项目可能终止。多数情况下沿着螺线的活动会继续下去，自内向外，逐步延伸，最终得到所期望的系统。

优点:

- 采用循环的方式逐步加深系统定义和实现的深度,同时降低风险。
- 确定一系列里程碑，确保利益相关者认可是可行的且令各方满意的解决方案。

缺点:

- 依赖大量的风险评估专家来保障其成功。如果存在较大的风险没有被发现和管理，就肯定会发生问题。

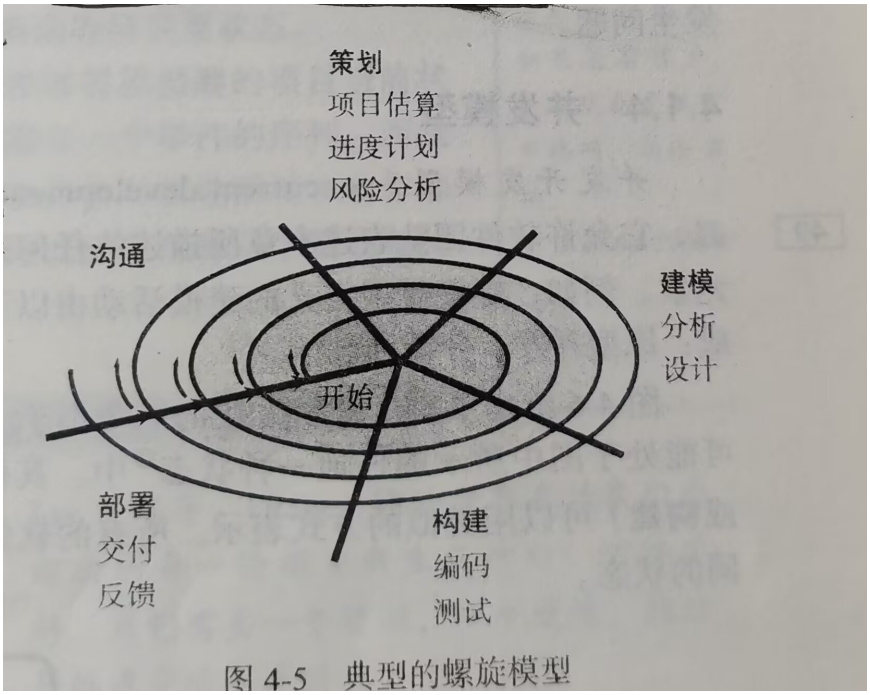


图 4-5 典型的螺旋模型

4.3 统一过程(Unified Process)

特点

- use-case driven （用例驱动）
- architecture-centric （软件架构为核心）
- Risk Focused （关注风险）
- 迭代增量过程(an iterative and incremental development process.)
- 与统一建模语言 （UML） 紧密结合
- Inception(起始)/Elaboration(细化)/Construction(构建)/Transition(转换)（/production(生产)

第五章 敏捷开发

第八章 理解需求