第二部分 建模

- 需求建模
- 体系结构设计
- 构件级设计
- 用户界面设计
- 基于模式的设计

进入软件工程实践的细节，课程的重点

第二部分 建模

- 指导软件工程实践的概念和原则有哪些?
- 什么是需求工程?什么是能导致良好需求分析的因素?
- 如何创建需求模型?它包含哪些元素?
- 好的设计包含哪些元素?
- 体系结构设计的作用，如何为其他的设计活动建立架构?使用什么模型?
- 如何设计高质量软件部件?
- 在设计用户接口时使用什么概念、模型和方法?
- 什么是基于模式的设计?
- 使用什么特别的策略和方法来设计WebApp和移动App?

第7章 指导实践的原则

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014). Slides copyright 2014 by Roger Pressman.

软件工程知识

- 经常听到人们说，软件开发知识的半衰期为3年：现在你需要知道的那些知识，在三年内会有一半将过时。在技术相关的知识领域内，这种说法可能是正确的。但还有另一种软件开发知识— 一种我认为是“软件工程原则”— 并没有3年半衰期的说法。这些软件工程原则可以为专业程序设计人员在其整个职业生涯内提供服务。

Steve McConnell

指导过程的原则

- **原则1：敏捷。**你所选择的过程模型是否是传统的或敏捷的，敏捷开发的基本原则会帮助你判断。
- **原则2：每一步都关注质量。**每个过程活动、动作及任务的出口条件应关注所生产的工作产品质量。
- **原则3：做好适应的准备。**过程不是信奉经验，其中没有信条。当需要的时候，就让你的方法适应于由问题、人员以及项目本身施加的限制。
- **原则4：建立一个有效的团队。**软件工程过程和实践是重要的，但最根本的还是人。必须建立一个彼此信任和尊重的自组织团队。

指导过程的原则

- **原则5：建立沟通和协调机制。**尽可能准确了解项目的有关信息和调动利益相关者的积极性。这些属于管理的问题，必须设法解决。
- **原则6：管理变更。**必须建立一种管理变更的机制，（方法可以是正式的或非正式的），来管理变更要求的提出、变更的评估、变更的批准以及变更实施的方式。
- **原则7：评估风险。**做事出错是必然的，建立应急计划是非常重要的。某些应急计划会成为安全工程任务的基准。（27章）
- **原则8：创造能给别人带来价值的工作产品。**仅仅创建那些能为其他过程活动、动作或任务提供价值的工作产品。

指导实践的原则

- **原则1：分治策略（分割和攻克）。**更具技术性的表达方式是：分析和设计中应经常强调关切点分离(SoC ，Separation of Concerns)。
- **原则2：理解抽象的使用。**在这一核心原则中，抽象就是对一个系统中一些复杂元素的简单化，用以传达词组的含义。（抽象是指从具体事物抽出、概括出它们共同的方面、本质属性与关系等，而将个别的、非本质的方面、属性与关系舍弃的思维过程。）抽象也是分层次的。
- **原则3：力求一致性。**一致性原则建议采用用户熟悉的上下文使软件易于使用。
- **原则4：关注信息传送。**必须特别注意界面的分析、设计、构建以及测试。

指导实践的原则

- **原则5：构建能展示有效模块化的软件。**对重要事务的分割（原则1）建立了软件的哲学。模块化提供了认知这一哲学的机制。模块应该是有效的。
- **原则6：寻找模式。**利用模式帮助软件开发者解决整个软件开发过程中反复出现的问题。
- **原则7：在可能的时候，用大量不同的观点描述问题及其解决方法。**
- **原则8：记住：有人将要对软件进行维护。**所有软件产品都不仅仅是为自己生产的。

沟通原则

- **原则1：倾听。**一定要仔细倾听讲话者的每一句话，而不是急于叙述你对这些话的看法。
- **原则2：有准备的沟通。**在与其他人碰面之前花点时间去理解问题。
- **原则3：沟通活动需要有人推动。**每个沟通会议都应该有一个主持人（推动者），其作用是：(1)保持会议向着有效的方向进行；(2)能调解会议中所发生的冲突；(3)能确保遵循我们所说的沟通原则。
- **原则4：最好当面沟通。**但是，如果能把一些相关信息写出来，通常可以工作得更好。
- **原则5：记笔记并且记录所有决定。**参与交流的记录员应该记录下所有要点和决定。

沟通原则

- **原则6：保持通力协作。**当项目组成员的想法需要汇集在一起用以阐述一个产品或者某个系统的功能或特征的时候，就产生了协作与协调的问题。
- **原则7：把讨论集中在限定的范围内。**在任何交流中，参与的人越多，话题转移到其他地方的可能性就越大。
- **原则8：如果某些东西很难表述清楚，采用图形表示。**
- **原则9：(a) 一旦认可某件事情，转换话题；(b) 如果不认可某件事情，转换话题；(c) 如果某项特性或者功能不清晰，当时无法澄清，转换话题。**
- **原则10：协商不是一场竞赛或者一场游戏，协商双赢时才发挥了协商的最大价值。**

策划原则

- **原则1：理解项目范围。**如果你不知道要去哪里，就不可能使用路线图。范围可以为软件开发团队提供一个目的地。
- **原则2：让利益相关者参与策划。**利益相关者能够限定一些优先次序，确定项目的约束。
- **原则3：要认识到计划的制定应按照迭代方式进行。**项目计划不可能一成不变。在工作开始的时候，有很多事情有可能改变。
- **原则4：基于已知的估算。**估算的目的是基于项目组对将要完成工作的当前理解，提供一种关于工作量、成本和任务工期的指标。

策划原则

- **原则5：计划时考虑风险。**如果团队已经明确了哪些风险最容易发生且影响最大，那么应急计划就是必需的了。
- **原则6：保持可实现性。**人们不能每天百分百地投入工作。
- **原则7：调整计划粒度。**粒度主要指项目计划细节的精细程度。
- **原则8：制定计划确保质量。**计划应该确定软件开发团队如何去确保开发的质量。
- **原则9：描述如何适应变化。**即使最好的策划也有可能被无法控制的变化破坏。
- **原则10：经常跟踪并根据需要调整计划。**项目一次会落后进度一天的时间。

建模原则

- 在软件工程中，要创建两类模型：
  - ***需求模型（也称为分析模型）***通过在以下三个不同域描述软件来表达客户的需求：信息域、功能域和行为域。
  - ***设计模型***表述了可以帮助开发者高效开发软件的特征：架构、用户界面以及构件细节。

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

建模原则（敏捷）

- 原则1：软件团队的主要目标是构建软件而不是创建模型。
- 原则2：轻装前进，不要创建任何你不需要的模型。
- 原则3：尽量创建能描述问题和软件的最简单模型。
- 原则4：用能适应变化的方式构建模型。
- 原则5：明确描述创建每一个模型的目的。

建模原则（敏捷）

- 原则6：调整模型来适应待开发系统。
- 原则7：尽量构建有用的模型而不是完美的模型。
- 原则8：对于模型的构造方法不要过于死板。
- 原则9：如果直觉告诉你模型不太妥当，尽管书面上很正确，那么也要仔细注意。
- 原则10：尽可能快地获得反馈，以消除误解、纠正错误，弥补不足。

建模原则（我的建议）

- 建模阶段尽可能考虑周全，落实时再裁剪
- 建模在领域内或按层次进行
- 模型总是包括静态的结构、功能和动态的行为

需求建模原则

- 原则1：必须描述并理解问题的信息域。
- 原则2：必须确定软件所要实现的功能。
- 原则3：必须描述软件的行为（作为外部事件的结果）。
- 原则4：描述信息、功能和行为的模型必须以一种能揭示分层（或者分级）细节的方式分解开来。
- 原则5：分析任务应该从本质信息转向实现细节。需求建模是在问题领域，从用户的角度来描述问题的，不要考虑解决方案。

设计建模原则

- 原则1：设计可追溯到需求模型。
- 原则2：要始终关注待建系统的架构。
- 原则3：数据设计与功能设计同等重要，是架构设计的基本元素。
- 原则4：必须精心设计接口（包括内部接口和外部接口）。
- 原则5：用户界面设计必须符合最终用户要求。

设计建模原则

- 原则6：构件级设计应是功能独立的。
- 原则7：构件之间以及构件与外部环境之间松散耦合。
- 原则8：设计表述（模型）应该做到尽可能易于理解。
- 原则9：设计应该迭代式进行。
- 原则10: 创建设计模型不包括在敏捷开发的方法中。

生存（Living）建模原则

- 原则1：利益相关者为中心的模型应以特定的利益相关者和他们的任务为目标。
- 原则2：模块和代码应该更加紧密联系在一起。
- 原则3：在模块和代码间建立双向信息流。
- 原则4：应该生成常规系统视图。

生存（Living）建模原则

- **原则5：模型信息必须持续跟踪系统变更。**
- **原则6：必须验证各个模型的信息一致性。**
- **原则7：每个模型元素分配给利益相关者权利和责任。**
- **原则8：各种模型元素的状态描述。**

构造原则

构造活动包括一系列编码和测试任务，从而为向客户和最终用户交付可运行软件做好准备。

- 编程原则和概念与编程风格、编程语言和编程方法紧密结合。
- 测试原则和概念的指导目标是要设计一些能用最短的时间、最少的工作量来系统地揭示不同类型错误的测试。

编码原则（准备原则）

- ***在写下每行代码之前，要确保：***
  - 理解所要解决的问题。
  - 理解基本的设计原则和概念。
  - 选择一种能够满足构建软件以及运行环境要求的编程语言。
  - 选择一种能提供工具以简化工作的编程环境。
  - 构件级编码完成后进行单元测试。

我觉得重要的要有一个编码规范

编码原则（编程原则）

- ***在开始编码时，要确保：***
  - 遵循结构化编程方法来约束算法[Boh00]。
  - 考虑使用结对编程。
  - 选择能满足设计要求的数据结构。
  - 理解软件架构并开发出与其相符的接口。
  - 尽可能保持条件逻辑简单。
  - 开发的嵌套循环应使其易于测试。
  - 选择有意义的变量名并符合相关编码标准。
  - 编写注释，使代码具有自说明性。
  - 增强代码的可读性（例如：缩进和空行）。有助于理解。

我觉得重要的是要写容易被理解的代码

编码原则（确认原则）

- ***在完成第一阶段的编码之后，要确保：***
  - 适当进行代码走查。
  - 进行单元测试并改正所发现的错误。
  - 重构代码。

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

测试原则

- Davis[Dav95b]提出了一套测试原则：
  - **原则1：所有的测试都应该可以追溯到用户需求。**
  - **原则2：测试计划应该远在测试之前就开始着手。**
  - **原则3：将Pareto原则应用于软件测试。80%的错误在20%的模块中。**
  - **原则4：测试应该从“微观”开始，逐步转向“宏观”。**

测试原则

- 原则5：穷举测试是不可能的。
- 原则6：为系统的每个模块做相应的缺陷密度测试。
- 原则7：静态测试技术能得到很好的结果。
- 原则8：缺陷跟踪和查找测试未覆盖缺陷的模式。
- 原则9：包含在演示软件中的测试用例是正确的行为。

部署原则

- 原则1：客户对于软件的期望必须得到管理。
- 原则2：完整的交付包应该经过安装和测试。
- 原则3：技术支持必须在软件交付之前就确定下来。
- 原则4：必须为最终用户提供适当的说明材料。
- 原则5：有缺陷的软件应该先改正再交付。

小结

- 软件工程实践包括概念、原则、方法和在整个软件过程中所使用的工具。虽然每个软件工程项目是不同的，但却有着通用的普遍原则和一些与项目或产品无关的适用于每个过程框架活动的实践任务。

需要通过实践来感悟来体会来掌握