【软件构造】第二章第二节 软件构造的过程、系统和工具

第二章第二节 软件构造的过程、系统和工具

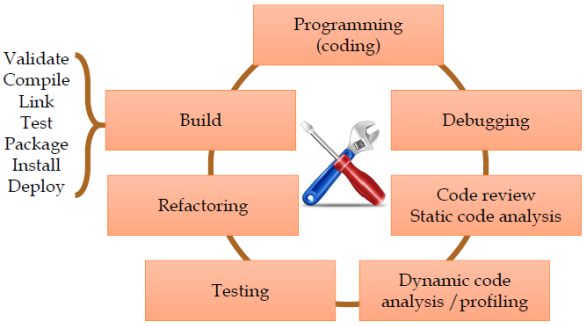
本节主要讲述软件从从零到一遵循的子过程，在考试中不会做重点考察

Outline

- 广义的软件构造过程
  - 编程
  - 静态代码分析
  - 动态代码分析
  - 调试与测试
  - 重构
- 狭义的软件构造过程
  - 构造系统：经典BUILD场景
  - 构造系统的组件
  - 构造过程和构造描述
  - Java编译工具
  - 子目标和结构变体
  - 构造工具

Notes

广义的软件构造过程



【编程(Coding)】

- 开发语言：如Java、C、Python
  - 使用IDE（集成开发工具）的优势（组成）
    - 方便编写代码和管理文件（有代码编辑器，代码重构工具、文件和库（Library）管理工具）
    - 能够编译、构建（有编译器、解释器、自动构建工具）
    - 结构清晰（有面向对象的类层次结构图和类浏览器）

        - 有GUI界面
        - 支持第三方扩展工具
- 建模语言：UML(Unified Modeling Language,统一建模语言）
    - UML是用来对软件系统进行可视化建模的一种语言；
    - UML的结构由一组一致的规则定义；
    - 建模的目的：
        - 有助于按照需求对系统进行可视化分析
        - 能够理解系统的结构或行为
        - 给出了构造系统的模板
        - 对做出的决策进行文档化
    - 更多关于UML，参考 jiuqiyuliang的博客专栏
- 配置语言：键值文件(.ini；.properties；.rc)；XML，YAML，JSON
    - 配置语言用于配置程序的参数和初始设置
    - 目的：
        - 部署环境设置
        - 应用程序功能的变体
        - 组件之间连接的变体

【静态代码分析】

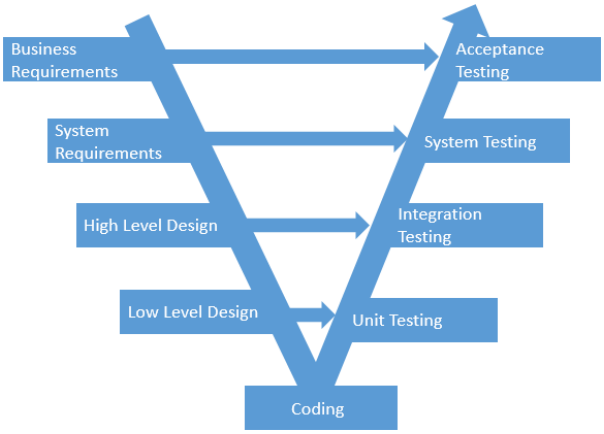
- 定义：静态代码分析是指不运行被测程序本身，仅通过分析或检查源程序的语法、结构、过程、接口等来检查程序的正确性。
- 注：该过程提供了对代码结构的理解，有助于确保代码符合行业标准
- 注：自动化的工具可以帮助程序员和开发人员进行静态代码分析

【动态代码分析】

- 定义：动态测试方法是指通过运行被测程序，检查运行结果与预期结果的差异，并分析运行效率、正确性和健壮性等性能。
- 注：必须执行足够的测试输入，使用诸如代码覆盖率之类的软件测试措施有助于确保已经观察到程序的一组可能行为的足够部分。
- 注：配置文件（“程序配置文件”，“软件配置文件”）是一种动态程序分析形式，用于度量程序的空间（内存）或时间复杂度，特定指令的使用情况，函数调用的频率和持续时间。

【调试与测试】

- 测试（Test）
    - 狭义：程序是否正常运行、能否满足所有需求
    - 广义：



- 调试（Debug）：识别错误的根本原因并对其进行纠正的过程。

【重构】

- 重构它不会改变代码的外部行为，但会改进其内部结构。
- 投入短期时间/工作成本以获得长期收益，并对系统的整体质量进行长期投资。
- 重构需要保持代码正常工作，只需要用一些小步骤保留语义
- 需要进行单元测试来证明代码正常工作

狭义的软件构造过程

【构造系统：经典的**BUILD**场景】

Build场景综述：

- 用传统编译语言(如C、C++、Java)编写软件（compilation）
- 用解释型语言(如Perl、Python)编写软件的打包和测试(packaging and testing)
- 用基于Web的应用程序进行编译和打包
  - 使用静态HTML页面
  - 使用Java或C# 编写的源代码
  - 使用JSP，ASP或PHP语法编写的混合文件以及多种类型的配置文件
- 执行单元测试代码的其余部分对软件进行隔离验证。
- 执行静态分析工具来是被程序源代码中的错误
- 生成PDF或HTML文档

传统编译语言：**C**、**C++**、**Java**等：

- 源文件被编译成目标文件，连接到代码库或可执行程序中
- 生成的文件被收集到可安装在目标机器上的发行包中
- 版本控制工具
- 源树和对象树：特定开发人员使用的源文件和编译对象文件集。
- 构建机器：执行编译工具的计算设备。
- 发布打包和目标机器：打包软件，分发给最终用户，然后安装到目标机器上的方法。

解释型语言：**Perl**、**Python**等：

- 解释的源代码不会编译到目标代码中，不需要对象树，解释源文件本身被收集到一个发行包中被安排在目标机器上；
- 编译工具专注于转换源文件并将它们存储在发行包中；
- 不在程序构建时编译成机器码

基于**Web**应用程序的构建系统：编译代码，解释代码和配置或数据文件的混合。

- 静态HTML文件，只包含标记数据显示在Web浏览器中，直接复制到发行包。
- 包含代码的JavaScript文件将由最终用户浏览器解释，直接复制到发行包。
- JSP，ASP或PHP页面，包含HTML和程序代码的混合，由Web应用程序服务器而不是构建系统编译和执行，复制到发布包，准备安装到Web服务器上。
- 构建系统在编译打包Java类文件之前执行转换。Java类在Web应用程序服务器上或浏览器内执行（使用小程序）。

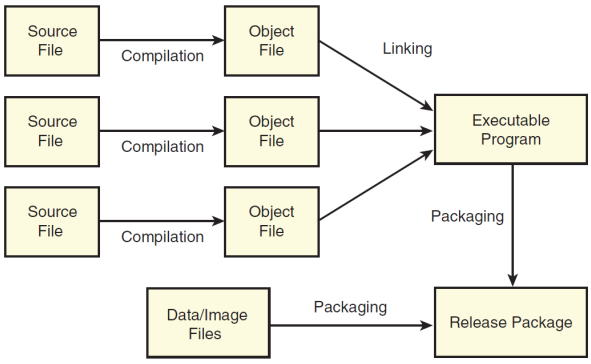
【构建系统的组件】**(**该部分不作为考试内容，内容转自 长安蒹葭的博客 以保持课程内容完整性**)**

- 源树：程序的源代码被存储为多个磁盘文件，将这些文件安排到不同的地方称为源树，其结构通常反映了软件的体系结构。
- 对象树：一个单独的树层次结构，用于存储由构建过程构建的任何对象文件或可执行程序。
- 编译工具：将可读的源文件转换为机器可读的可执行程序文件的程序。
  - 编译器：源文件 **->** 对象文件

链接器：多个相关的目标文件 -> 可执行的程序映像
- 基于UML的代码生成器：模型 -> 源代码文件
- 文件生成器：脚本 -> 文件

- 发布打包和目标机器：生成可以实际安装在用户机器上的东西。
  - 从源和目标树中提取相关文件并将它们存储在发行包中。
  - 发行包应该是单个磁盘文件，并且应该进行压缩以减少下载所需的时间。
  - 任何不重要的调试信息都应该被删除，以免它使软件的安装变得混乱。
- 包装类型：
  - 档案文件：zip和解压缩
  - 软件包管理工具：UNIX风格，例如.rpm和.deb
  - 定制的GUI安装工具：Windows风格

【构造过程与构造描述】



- 如何构建系统
  - 开发人员构建：开发人员已检出VCS的源代码并正在专用工作区中构建软件，结果发布包将用于开发人员的私人开发。
  - 发布版本：为测试组提供一个完整的软件包供验证，软件的质量足够高时为客户提供相同的软件包。用于发布版本的源代码树只编译一次，永不修改。
  - Sanity构建：与发布版本类似，但并非针对客户，可以每天发生多次，并且趋向于完全自动化。

【子目标和构建变体】**(**该部分不作为考试内容，内容转自 长安蒹葭的博客 以保持课程内容完整性**)**

- 三种不同的构建方法
  - 构建子目标：仅重建开发人员正在处理的树的部分。
  - 构建不同版本的软件：定制输出以改变软件的行为。
  - 构建不同的目标体系结构：为各种不同的CPU类型和操作系统编译相同的源文件集，包括x86，MIPS和PowerPC等CPU以及Linux，Windows和Mac OS X等操作系统。
- 构建子目标
  - 任何大型软件都可以分成许多子组件，通常采用静态或动态库的形式。
  - 避免耗费时间，最好限制构建子组件的数量。

【构建工具】

- Java构造工具：Make、Ant、Maven、Gradle、Eclipse
- Maven将项目的生命周期大致分为9个，分别为：clean、validate、compile、test、package、verify、install、site、deploy
  - 使用maven自动构建的方法：如mvn compile
    - validate - 验证项目是否正确，并提供所有必要的信息
    - compile - 编译项目的源代码
    - test - 使用单元测试框架测试已编译的源代码。这些测试不应该要求打包或部署代码

- package - 获取已编译的代码并将其打包为可分发的格式，例如JAR。
- verify - 对集成测试结果进行任何检查，以确保符合质量标准
- install - 将软件包安装到本地存储库中，作为本地其他项目的依赖项
- deploy - 在构建环境中完成，将最终包复制到远程存储库，以便与其他开发人员和项目共享。

- 更多参考 u011991249的专栏