一、软件构造的多维视图

	Moment		Period	
	Code-level	Component-level	Code-level	Component-level
Build-time	Source code, AST, Interface-Class-Attribute-Method (Class Diagram)	Package, File, Static Linking, Library, Test Case, Build Script (Component Diagram)	Code Churn	Configuration Item, Version
Run-time	Code Snapshot, Memory dump	Package, Library, Dynamic linking, Configuration, Database, Middleware, Network, Hardware (Deployment Diagram)	Execution stack trace, Concurrent multi-threads	Event log, Multi-processes, Distributed processes
			Procedure Call Graph, Message Graph (Sequence Diagram)	

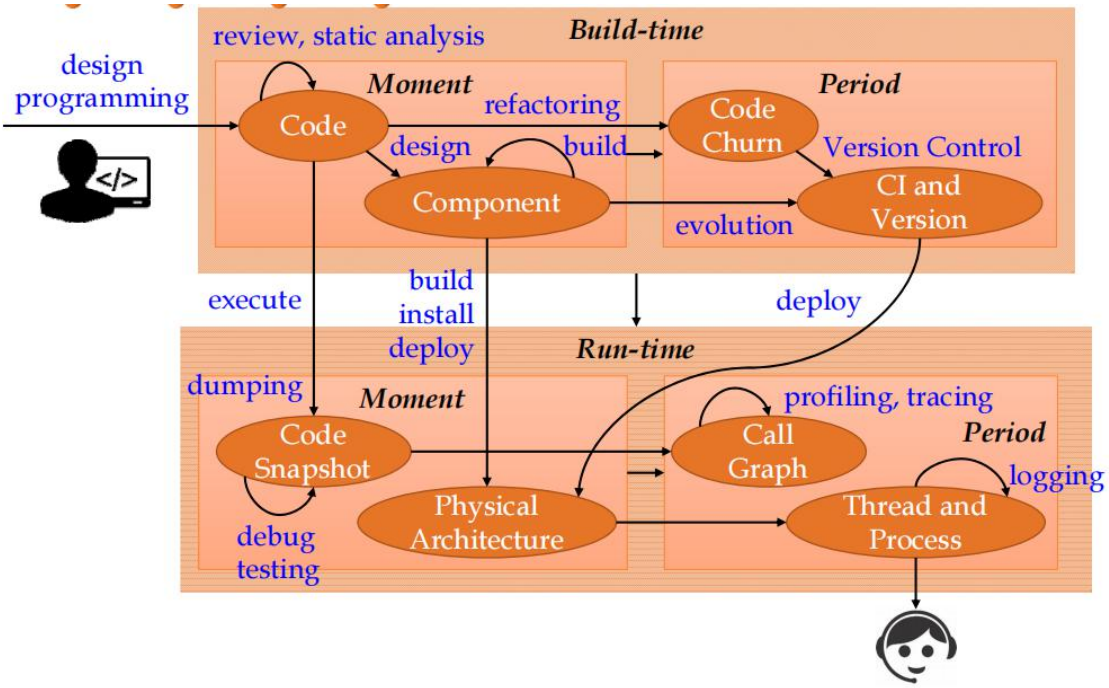
Build-time View:构建阶段
Run-time View:运行阶段
Code-level View:代码的逻辑组织
Component-level View:代码的物理组织
Moment View:特定时刻的软件形态
Period View:软件形态随时间的变化

Build-time View
1. build+moment+code
    Functions,classes,methods,interfaces
    词汇层面：source code 半结构化
    语法层面：Abstract Syntax Tree(AST)
    语义层面：Interface-Class-Attribute-Method(Class Diagram 类图)
        接口-类-选择器-方法 完全结构化
2. Build+period+code
    Chages
    Code Churn:代码变化
3. Build+moment+component
    模块化组织为文件、目录(file、dictionary)
    文件被压缩为 package、library
    静态链接
    Test case
    Build Script

4. build+period+component
    Software Configuration Item(SCI 配置项)
    Version（版本）
    Version Control System（VCS）

Run-time View
    Executable programs：可执行程序
        Native Machine Code 原生机器码
        Full Program Interpretation 程序完全解释执行
        Interpreted byte codes 解释型字节码
        Java Virtual Machine
    Dynamic linking 动态链接
        不会加入可执行文件
        做标记
        运行时根据标记装载库至内存
        发布软件时，记得将程序所有依赖的动态库都复制给用户
        优点：易于升级
    Configuration and Data Files
    Distributed Programs 分布式程序
5. Run+moment+code
    Snapshot diagram 代码快照图
    Memory dump 内存信息转储
6. Run+period+code
    Sequence diagram in UML
    Execution tracing 执行跟踪
        用日志方式记录程序执行的调用次序
    Concurrent multi-threads
7. Run+moment+component
    Deployment diagram in UML
8. Run+period+component
   Event logging 事件日志

二、软件各阶段的构造活动



三、软件的质量指标

外部质量因素影响用户
内部质量因素影响软件本身和它的开发者
外部质量取决于内部质量

1. 外部因素
   (1) 正确性 Correctness
      ① 最重要的质量指标
      ② 按照预先定义的规约执行
      ③ 测试和调试：发现不正确、消除不正确
      ④ 防御式编程：在写程序的时候就确保正确性
      ⑤ 形式化方法：通过形式化验证发现问题
   (2) 健壮性 Robustness
      ① 针对异常情况的处理
      ② 是对正确性的补充
      ③ 出现规约定义之外的情形时，软件要作出恰当反映
      ④ 出现异常时不要崩溃
   (3) 可拓展性 Extendibility
      ① 对软件的规约进行修改，是否足够容易
      ② 为什么拓展：应对变化
      ③ 简约设计主义、分离设计主义
   (4) 可复用性 Reusability

① 一次开发，多次使用
② 发现共性

(5) 兼容性 Compatibility
① 不同软件系统之间互相可容易的集成
② 保持设计的同构性
③ 关键：标准化

(6) 效率 Efficiency
① 性能毫无意义，除非有足够的正确性
② 对性能的关注要与其他质量属性进行折中
③ 过度的优化导致软件不再适应变化和复用

(7) 可移植性 Portability
① 软件可方便的在不同的技术环境之间移植
② 硬件、操作系统

(8) 易用性 Ease of use
① 容易学、安装、操作、监控
② 给用户提供详细的指南

(9) 功能性 Functionality
① 程序设计中一种不适宜的趋势，即软件开发者增加越来越多的功能，企图跟上竞争，其结果是程序极为复杂、不灵活、占用过多的磁盘空间。
② 每增加一小点功能，都确保其他质量属性不受到损失

(10) 及时性 Timeliness

(11) 其他因素

Verifiability 可验证性
Integrity 完整性
Repairability 可修复性
Economy 经济性

2. 内部因素

(1) LOC
lines of code

(2) Cyclomatic Complexity 圈复杂度
衡量一个模块判定结构的复杂程度

(3) Architecture-related factors
Coupling 耦合度 --> 低
Cohesion 内聚度 --> 高
矛盾

(4) 可读性

(5) 易理解性

(6) 清晰 Clearness

(7) 复杂度

(8) 大小 Size

3. 折中

因素之间相互影响、矛盾、相关
经济性 与 功能性/可复用性 矛盾

有效性/可复用性 与 轻便性 矛盾
更高效、对硬件和软件有高要求
时效性 与 可扩展性 矛盾
完整性 与 易用性 矛盾

正确性绝不能与其它质量因素折中