系统分析与设计概述

1. 系统的概念与特性

1.1 系统的概念

> 系统是一组为实现某些结果相互联系、相互作用的部件的集合体。

1.2 系统特性

1.2.1 整体性

1. 系统的最基本特性，观察和分析系统最基本的思想和方法。
2. 系统的整体功能应该大于各部分功能之和。

1.2.2 目的性

1. 每个系统都要实现一定的功能，这是区分不同系统的标志。
2. 系统的目的一般通过更具体的目标实现，系统多个目标有时不完全一致，甚至互相矛盾，这就需要协调，寻找平衡和折中的方法，从而收到整体最佳的效果。

1.2.3 相关性

系统内各部分存在相互依赖和相互制约关系的特定关系，某一部分的变化会影响其他部分的实现。

1.2.4 环境适应性

任何系统都存在于一定的环境中，必然要与外界进行物质、能量和信息的交换，外部环境的变化会相应的引起系统功能和内部组成的变化。系统具有适应环境变化，保持原有功能的特性。

1.2.5 层次性

系统无论大小，都可以分解为一系列的子系统，并存在一定的层次结构，系统各个层次具有独立的功能，它们通过相互联系，相互作用共同完成系统的功能。

2. 系统分析与设计方法

2.1 系统分析

理解问题域

> 是对一种业务问题域的学习活动，能够在系统解决方案中为提升系统性能和明确业务需求提供良好的建议。

2.2 系统设计

求可行解

> 是对系统分析已确定的业务需求的说明或者构建一种相关技术的解决方案。

2.3 系统分析的步骤

系统分析首先要着眼于系统整体，要先分析整体，再分析部分；先看全局，后看局部；先看全过程，再看某一个阶段；先看长远，再看当前。

2.3.1 明确问题，设立目标

明确要研究问题的性质和范围，提出所要达到的目标，明确约束范围。

2.3.2 收集资料，制定方案

收集备选资料，制定解决问题的各种备选方案，预计可能产生的各种结果。

2.3.3 分析计算，评价比较

对资料和数据做必要的计算，进行各子系统的分析，再进行系统的整体分析，将各种方案进行评价对比，选择最佳方案。

2.3.4 校验核实，做出决策

如果对制定的方案不满意，还可按上述程序反复进行，直到获得满意为止。

3. 软件研发趋势

3.1 信息技术发展历程

1. 语言的产生和使用
2. 文字的创造与应用
3. 造纸术和印刷术的发明与应用
4. 电报、电话、广播和电视的发明和普及应用
5. 计算机和网络的普及应用

3.2 现代软件系统的内在特性

复杂性、一致性、可变性和不可见性。其中，复杂性和可变性是根本问题。

3.2.1 复杂性

> 软件实体可能比任何由人类创造的实体更加复杂。

原因如下：

1. 软件拥有大量的状态。
2. 复杂性不仅使技术实现困难，还会引发管理上的问题。

3.2.2 可变性

体现在：

1. 软件是纯粹思维的产物，可以无限扩展而且修改容易。
2. 客户对软件系统的要求随时可能会变
3. 软件的设计和习惯随着开发语言、技术、方法、人的经验习惯而随时发生变化

3.2.3 一致性

> 软件开发的目标就是兼容性，特别是对于复杂度随心所欲，毫无规则的系统。

一致性需要考虑的实例：

1. 需求分析人员列出的规格是用户想要的吗?
2. 研究人员的实现和系统工程师的想法一致吗?
3. 高层软件组和底层软件组对接口的理解一致吗?

3.2.4 不可见性

> 软件是不可见的，无法可视化。

当前的软件表达方式和建模方法都无法详尽展现软件，限制了设计和实现过程。

3.3 传统软件开发 vs. 现代软件开发

- 传统软件开发把开发类比为传统工业，规范后可“重复生产”。
- 现代软件开发认为软件开发是复杂的有机生态系统，不可重复。

3.4 软件发展趋势

3.4.1 传统IT厂商在向云服务转型

云软件是商业模式的变革，是“供应商+客户/运营商”到“开发+运营”的转变。

3.4.2 云计算时代的软件在发生变化

1. 商业模式在发生变化
2. 系统架构在发生变化
3. 开发模式在发生变化

3.4.3 多语言编程成为常态

在云计算、大数据等开源项目中，多语言混合编程成为常态。

3.4.4 企业自身软件交付需应对来自市场、协作、开放与安全的多重挑战

1. 交付频率高，研发周期短
2. 跨地域协作多，研发平台复杂
3. 开放与安全要求高

3.4.5 软件企业的竞争已经从单一产品竞争演进到生态联盟竞争

1. 软件需求方与协作方的合作
2. 软件园区、孵化器向软服务转型
3. 高校、培训机构等与产业需求密切衔接

4. DevOps研发模式

4.1 DevOps兴起的驱动因素

业务诉求：业务负责人要求加快产品交付的诉求

能力基础：大规模使用敏捷开发过程与方法

技术基础：虚拟化和云计算基础设施日益普遍

工程基础：数据中心自动化技术和配置管理工具的普及

4.2 DevOps定义与关键

> DevOps是一套实践方法，在保证高质量的前提下，缩短系统变更从提交到部署环境的时间。

4.2.1 DevOps关键

1. 部署对环境的变更时，质量很重要
2. 交付机制是高质量的
3. 开发人员提交新开发的代码时间和代码部署到生产环境的时间至关重要
4. 不拘泥于实践的形式或者使用何种工具，目标是减少 3 中的时间
5. 需求阶段要包含运维人员的视角

4.3 DevOps五个要素

1. 文化

   建立一体化的功能团队，打破开发与技术运营隔阂

2. 自动化

   自动化一切可以自动化的

3. 精益

   以精益的方式小步快跑，持续改善

4. 度量

   建立有效的监控与度量手段获得反馈，推动产品和团队的持续改进

5. 分享

   不同职能，不同产品之间共享经验

4.4 DevOps生命周期过程

计划 -> 编码 -> 构建 -> 验证 -> 发布 -> 部署 -> 运维

持续集成：计划、编码、构建、验证

持续交付：计划、编码、构建、验证、发布、部署

自动化和监控：构建、验证、发布、部署、运维

4.5 DevOps关键模式与实践

人、流程、技术、文化

4.5.1 人

1. 自治团队

2. 全栈工程师

3. 融合团队

4. 服务式管理者

4.5.2 流程

1. MVP

2. CI/CD

3. TDD

4.5.3 技术

1. 开发者自服务
2. 基础设施即代码
3. 集成工具链

4.5.4 文化

1. 合作文化
2. 持续体验

4.6 华为云

项目管理 -> 配置管理 -> 代码检查 -> 编译构建 -> 测试 -> 发布 -> 流水线