软件安全开发服务规范

目录

一、总体情况

软件开发项目的开发过程通常有以下几个阶段：准备阶段、需求阶段、设计阶段、编码阶段、测试阶段、验收阶段、维保阶段。具体参见《软件安全开发服务流程 》。

软件开发从阶段特性、项目组织和管理上的特性，通常可分为几种生命周期模型 。

软件生命周期模型，是指软件开发全部过程、活动和任务的结构框架。软件开发包括需求、设计、编码和测试等阶段，有时也包括维护阶段。 软件生命周期模型能清晰、直观地表达软件开发全过程，明确规定了要完成的主要活动和任务，是软件开发工作的基础。

典型的生命周期模型有：瀑布模型(Waterfall Model)、快速原型模型(RAPid Prototype model)和螺旋模型(SPIral model)。

通过比较在软件工程领域常用的几种软件生命周期模型，不难发现，它们都是以“瀑布模型”为基础，在其上进行了一些增加、迭代和删减，使其能适应某一类软件的开发活动。由于目前还没有一种软件生命周期模型能够适用所有的软件开发活动，所以根据软件的特性，设计适合的软件生命周期模型就显得尤为必要。

生命周期法对应用系统全面完整地按软件开发阶段顺序（直线型）进行。每阶段工作完成后，将分析和设计结果用标准、规范的文档记录下来，由相关业务人员、技术人员、用户和管理人员评审确认。评审通过后才进入下一个阶段直至项目完成。

原型法不过分强调系统的完整性，而是在获得一组基本需求后，借助一些便利的软件开发工具，沿系统的关键部分，尽可能快的构造一个实际系统的简化模型，供开发人员和用户进行交流，以便更准确、更充分地获得用户的需求和关键技术解决方案 。然后随着用户和开发人员对系统理解的加深而逐步求精，使系统不断完善。

从分析问题和解决问题的行为特性可把软件开发方法分为以下三种：结构化方法、信息建模方法和面向对象方法。

结构化方法是传统的软件开发方法。它对系统功能不断分解、细化，再根据功能的需要分析、设计数据结构。它用数据字典、数据流图、程序模块结构图、处理逻辑，说明描述系统的逻辑模型。

信息建模的方法认为数据是数据处理的中心，数据是稳定的，而功能是多变的。它首先针对业务系统 5-10 年的策略设计出各个业务领域稳定的主题数据库。然后，将系

统功能分解为对主题数据进行基本操作的业务过程加以实现 。其结果用主题数据库的数据结构图、系统功能到业务过程的分解、业务过程的动作图和数据流图表示。

面向对象方法是当前比较流行的软件开发方法，用对象来描述系统中的事物。对象有四个要素，即数据抽象、封装、消息传递和继承。对象包括属性（即数据）和方法（一组操作），它们被封装在对象中，别的对象只能通过该对象的方法存取该对象的数据。一个系统由若干相互关联的对象组成，对象间通过消息进行通信。对象类之间存在继承关系。

面向对象开发方法首先从系统的静态特性、动态特性和处理特性三种不同的角度，建立业务领域的对象模型（对象类层次图）、动态模型（对象状态变化图）和功能模型（数据流图），然后，将其平滑映射到机器世界中，设计出数据结构和程序模块。

二、软件安全开发需求分析规范

（一）开发项目的过程管理

开发项目的过程管理按《产品开发+上线项目过程指导手册》的要求执行。

（二）项目策划

任务交接后，任命项目经理并成立开发项目组。项目经理按公司项目管理办法，组织项目策划，明确项目的各详细子任务，划分开发阶段，并对各个阶段的进度、工作产品、质量、人员、设备、开发过程的数据收集进行控制和安排。《软件开发计划》需通过评审。

（三）需求分析

开发项目组根据《软件开发计划》等对开发产品的功能与非功能性需求进行详细分析，形成《需求分析报告》。《需求分析报告》需通过评审。

2.1.任务要求

2.1.1 领域需求认定

从业务领域的角度（用户的角度）细化、确定目标系统需具备的业务处理功能。要求完整、具体、准确。

2.1.2 领域活动模式

从业务领域的角度确定目标系统各部分的任务、内容、运作模式。并对落后的、制约系统运行效率的领域活动进行改造。

2.1.3 领域逻辑分析

从业务领域的角度分析目标系统各部件间的关系。并对不规范，不合理的业务领域运作流程进行改造。

2.2.操作方法

根据采用的不同软件开发方法（结构化方法、信息建模方法、面向对象方法），可采用不同的方法步骤（以下仅以结构化方法为例 ）。

2.2.1 通过现场调查，并与用户进行交流、座谈，明确业务领域的具体需求

如果是针对已有软件产品的升级维护，还应注意从以往的《软件运行支撑记录 》中和工程推广服务记录中收集具体需求。

2.2.2 根据业务领域需求，确定系统有关的业务领域活动范围

通过业务调查，确定与系统目标有关的所有业务活动。深入各业务部门，了解各部门职能、人员构成、主要活动、内部信息交流以及与其他部门的信息往来。

对各个业务活动，明确其信息流向、加工逻辑。并对不足之处，在不违反业务领域规定的前提下进行改造，设计出更高效的运作模式。

2.2.3 收集、归纳、分析领域数据

收集业务领域中的所有原始报表、单据，调查它们的来源、用途、使用人员、使用频率、数据量，明确其中包含的各数据项的含义、取值限制、计算方法。为了使收集到的数据完整、准确，可由用户协助收集。

2.2.4 领域模式分析

按业务领域情况对各种领域活动进行分析。

分析内容：领域活动功能，领域活动流程。

提取领域活动中的原子行为：

从规范化的数据流程图中提取数据处理，分析每一数据处理的行为，给出详细的其处理逻辑。

提取业务领域活动中的原子信息：

分析收集到的报表、单据中的数据项，对照规范化后的数据流程图中相应的信息流，确定每个信息流的数据结构，编写数据字典 DD。

2.2.5 领域联系分析

用分层数据流程图客观描述各子系统内部及其之间流动的信息。

修正数据流程图，使得数据规范，流程符合业务标准，系统能够计算机化。

2.3. 系统目标分析

对系统建设目标进行分析，描述本次项目的目标。

- **功能实现相关的问题**

  描述项目系统功能实现会遇到的相关问题。

- **问题症结分析**

  对功能实现会遇到的问题，进行分析并描述处理办法。

- **业务目标分析**

  描述本次项目系统面对的业务目标。

➢ **信息化目标分析**

对本次建设的系统进行信息化目标分析，如：

(1) 建设统一集中的数据库，统一存储和管理信息资源。

(2) 按安全等级保护和分级保护的要求，建设基础安全防护体系，建设安全风险应对机制，提高网上业务经办的基础安全防护能力，应对网络接入威胁和人为威胁等。

➢ **业务范围和业务流程分析**

- **业务范围分析**

  描述本次项目的业务范围，并对业务范围进行分析。

- **业务流程分析**

  描述本次项目的业务流程，并进行流程分析。

➢ **系统需求分析**

- **性能需求**

  描述系统建设的性需求。如：

  应用响应时间 < 3 秒；

  日常查询时间不超过 10 秒；

  一般情况下单个批处理应用不超过 5 分钟。

- **功能需求**

  描述系统建设的功能需求。

- **安全性要求**

  描述系统建设的安全性需求。

- **业务完整性要求**

  描述对项目业务完整性的要求，如提供健全的业务日志管理平台，业务信息记录等信息，保证业务的完整性。

- **业务要求**

  描述本次系统建设对业务的要求，分别叙述业务的服务模式、服务体系和服务目标的等。

2.4.遵循规范

按公司现行有效的需求管理技术要求执行。

三、软件安全开发设计规范

开发项目组根据《需求分析报告》按照公司技术标准及技术规范进行数据库设计 、业务流程设计、系统功能概要设计、系统功能详细设计、系统技术框架设计和系统部署设计等，形成《软件设计报告》以及 CDM/PDM/UML 等设计模型。《软件设计报告》需通过评审。

- **安全设计规范**

安全设计应遵循：

（一）保护最薄弱的环节原则：保护最易受攻击影响的部分；

（二）众深防御原则：不同层面、不同角度之间需要的最小权限；

（三）最小权限原则：只授予执行操作所需的最小权限；

（四）最小共享原则：使共享文件资源尽可能少；

（五）权限分离原则：授予不同用户所需的最小权限，并在它们之间形成相互制约的关系。

安全设计应包括：

（一）确定安全体系架构，设计安全协议和安全接口；

（二）确定访问控制与身份鉴别机制，定义主体角色和权限；

（三）数据结构安全设计，选择加密方法和算法；

（四）确定敏感数据保护方法；

（五）内部处理逻辑安全设计；

（六）评估内部通信机制，确定完整机制。

3.1.任务要求

3.1.1 系统需求认定

从软件实现的角度，细化、确定目标（计算机）系统应具备的软件功能。

3.1.2 系统运行模式设计

从软件实现的角度，对目标系统的基本模式进行组织、划分，并设计其系统行为和系统数据。

3.1.3 系统结构设计

从软件实现的角度，分析目标系统各模式间的关系。

3.2.操作方法（以结构化方法为例）

3.2.1 确定系统边界、整理运行模式

确定系统模型的人机分界线，划定数据处理自动化的范围。

3.2.2 系统模式设计

元操作分析 / 提取：

根据分析阶段得到的业务领域原子行为，提取最底层的原子数据操作（元操作），明确其输入数据项、输出数据项和加工逻辑。

数据库设计：

系统结构分析阶段提取的原子信息，重新组合同类数据要素为关系。确定关系之间和关系内部的属性之间的数据依赖。用规范化方法规范这些关系。最后，建立系统全部数据的 E-R 模型。

系统功能设计：

参考设计出的关系，按照一定的原则，聚合操作对象一致或相近的元操作，组成底层软件功能模块，用 IPO 图定义出来。然后，将逻辑上相关的底层模块构架为上层模块，得到系统模块结构图。

3.2.3 系统结构设计

以设计好的表和功能模块作为基本元素，根据元操作的处理流程画出系统的模块结构图，给出模块之间的调用关系。

3.3.遵循规范

按公司现行有效的设计管理技术要求执行。

四、软件安全开发编码规范

开发项目组根据《需求分析报告》《软件设计报告》按照公司技术标准和规范进行编码，单元测试、功能集成，形成系统源代码、软件包、单元测试代码。

- **安全编码规范**

（1）输入验证和数据合法性校验

程序接受数据可能来源于未经验证的用户，网络连接和其他不受信任的来源，如果未对程序接受数据进行校验，则可能会引发安全问题。

（2）避免 SQL 注入

使用 PreparedStatement 预编译 SQL，解决 SQL 注入问题，传递给 PreparedStatement 对象的参数可以被强制进行类型转换，确保在插入或查询数据时与底层的数据库格式匹配。

（3）避免 XML 注入

通过 StringBulider 或 StringBuffer 拼接 XML 文件时，需对输入数据进行合法性校验。对数量 quantity 进行合法性校验，控制只能传入 0-9 的数字。

（4）避免跨站点脚本（XSS）

对产生跨站的参数进行严格过滤，禁止传入<SCRIPT>标签

（5）声明和初始化

避免类初始化的相互依赖，类加载时初始化指向 Cycle 类的静态变量 c，而类 Cycle 的无参构造方法又依赖静态变量 deposit，导致无法预期的结果。

（6）不可忽略方法的返回值

忽略方法的放回值可能会导致无法预料的结果。

（7）不要引用空指针

当一个变量指向一个 NULL 值，使用这个变量的时候又没有检查，这时会导致。

NullPointerException。在使用变量前一定要做是否为 NULL 值的校验。

（8）使用 Arrays.equals（）来比较数组的内容

数组没有覆盖的 Object.equals()方法，调用 Object.equals()方法实际上是比较数组的引用，而不是他们的内容。程序必须使用两个参数 Arrays.equals（）方法来比较两个数组的内容。

（9）数字类型和操作

防止整数溢出，使用 java.lang.Number. BigInteger 类进行整数运算，防止整数溢出。

（10）避免除法和取模运算分母为零：要避免因为分母为零而导致除法和取模运算出现异常。

（11）类和方法操作：数据成员声明为私有，提供可访问的包装方法；攻击者可以用意想不到的方式操纵 public 或 protected 的数据成员，所以需要将数据成员为 private，对外提供可控的包装方法访问数据成员。

（12）敏感类不允许复制：包含私人的，机密或其他敏感数据的类是不允许被复制的，解决的方法有两种：1 、类声明为 final ；2 、Clone 方法抛出 CloneNotSupportedException 异常。

（13）比较类的正确做法：如果由同一个类装载器装载，它们具有相同的完全限定名称，则它们是两个相同的类。

（14）不要硬编码敏感信息：硬编码的敏感信息，如密码，服务器 IP 地址和加密密钥，可能会泄露给攻击者。敏感信息均必须存在在配置文件或数据库中。

（15）验证方法参数：验证方法的参数，可确保操作方法的参数产生有效的结果。不验证方法的参数可能会导致不正确的计算，运行时异常，违反类的不变量，对象的状态不一致。对于跨信任边界接收参数的方法，必须进行参数合法性校验。

（16）不要使用过时、陈旧或低效的方法：在程序代码中使用过时的、陈旧的或低效的类或方法可能会导致错误的行为。

（17）数组引用问题：某个方法返回一个对敏感对象的内部数组的引用，假定该方法的调用程序不改变这些对象。即使数组对象本身是不可改变的，也可以在数组对象以外操作数组的内容，这种操作将反映在返回该数组的对象中 。如果该方法返回可改变的对象，外部实体可以改变在那个类中声明的 public 变量，这种改变将反映在实际对象

中。

（18）不要产生内存泄露：垃圾收集器只收集不可达的对象，因此，存在未使用的可到达的对象，仍然表示内存管理不善。过度的内存泄漏可能会导致内存耗尽，拒绝服务（DoS）。

（19）异常处理：不要忽略捕获的异常；对于捕获的异常要进行相应的处理，不能忽略已捕获的异常。

（20）不允许暴露异常的敏感信息：没有过滤敏感信息的异常堆栈往往会导致信息泄漏。

（21）不允许抛出 RuntimeException, Exception,Throwable。

（22）不要捕获 NullPointerException 或其他父类异常。

（23）不要调用 Thread.run（），不要使用 Thread.stop（）以终止线程。

（24）相互依存的任务不要在一个有限的线程池执行：有限线程池指定可以同时执行在线程池中的线程数量的上限。程序不得使用有限线程池线程执行相互依赖的任务 。可能会导致线程饥饿死锁，所有的线程池执行的任务正在等待一个可用的线程中执行一个内部队列阻塞。

（25）及时释放资源：垃圾收集器无法释放非内存资源，如打开的文件描述符与数据库的连接。因此，不释放资源，可能导致资源耗尽攻击。

（26）不要序列化未加密的敏感数据：序列化允许一个对象的状态被保存为一个字节序列，然后重新在稍后的时间恢复，它没有提供任何机制来保护序列化的数据 。敏感的数据不应该被序列化的例子包括加密密钥，数字证书。

4.1.任务要求

4.1.1 数据组织

结合所选定的系统平台进行详细的、规范的数据结构设计。

4.1.2 程序设计

结合所选用的开发工具，进行应用系统组织，并设计实现各个目标模式所需的处理流程，功能菜单和人机界面。

4.1.3 机器实现

在选定的开发平台上，用平台提供的工具实现目标系统。

4.2.操作方法

4.2.1 创建数据结构

根据应用的需要，修改数据模式。如逆规范化，定义索引、聚簇，决定数据组织（库，文件）与分配方案。

用选定的数据库平台提供的 DDL 语言，建立数据库，包括定义 database、table、index、cluster、database device、segment、view 等等。编写数据库模式定义脚本。

4.2.2 编写联机帮助与使用手册。

五、软件安全开发测试规范

测试内容应包括代码的安全测试和安全功能测试。代码的安全测试是指使用代码测试工具来识别代码的安全脆弱性，并应按照其提供的修复建议进行修复。安全功能测试主要包括身份认证和访问控制的功能测试。

测试系统环境应尽可能模拟生产环境，并与生产环境进行安全隔离。

真实数据不得直接在测试环境中使用，须进行适当修改或屏蔽。在测试完成之后，须立即从测试应用系统清除运行信息。

测试人员编制安全测试方案，构造安全测试用例。

测试人员不得由开发人员兼任。

信息安全等级保护定级为二级及以下的应用软件，由技术部门组织代码漏洞检测；信息安全等级保护定级为三级及以上的应用软件，技术部门应聘请有相关资质的专业机构进行代码漏洞检测，并提交分析报告。

六、软件安全开发验收规范

软件通过测试，试运行正常，项目组应尽快做好软件产品推广工作，组织软件产品的发布。

测试组对成果验收后发放《软件发行通知单》。

公司知识产权管理部门牵头组织申报相应软件著作权登记和软件产品登记。

- **项目结项**

按《软件验收标准》要求执行。

七、软件安全开发维护规范

在一个软件产品在经过测试验收交付后，当产品进入市场或工程实施时，还会由于种种原因而对软件提出修改的要求，也就是进行软件维护。软件维护是在软件产品交付之后，为纠正故障，改进性能和其它属性，或使产品适应改变了的环境所进行的修改活动。软件维护实质上包含了需求分析、设计、实现、测试等软件开发活动。软件维护一般分为完善性维护、适应性维护和改正性维护三种类型。

完善性维护是为扩充功能和改善性能而进行修改和扩充，以满足用户变化了的需求。

适应性维护是为适应软件运行环境的变化而作的修改。例如因为硬件配置、系统软件的变化而要求进行的修改。

改正性维护是为了维持系统操作运行，针对在开发过程产生但在测试和验收时没有发现的错误而进行的改正。

软件维护对产品提供可持续改进和运行支撑。技术支持与服务部门根据产品使用方提出的技术支持申请及《软件缺陷跟踪报告》进行技术支持与服务，并进行任务跟踪管理，建立产品知识及问题库。形成《软件维护报告》。

开发项目组根据需求变更及积累的《软件缺陷跟踪报告》，对提出的需求变更、产品缺陷及问题，对产品提供可持续改进和升级，按照公司软件开发流程执行。