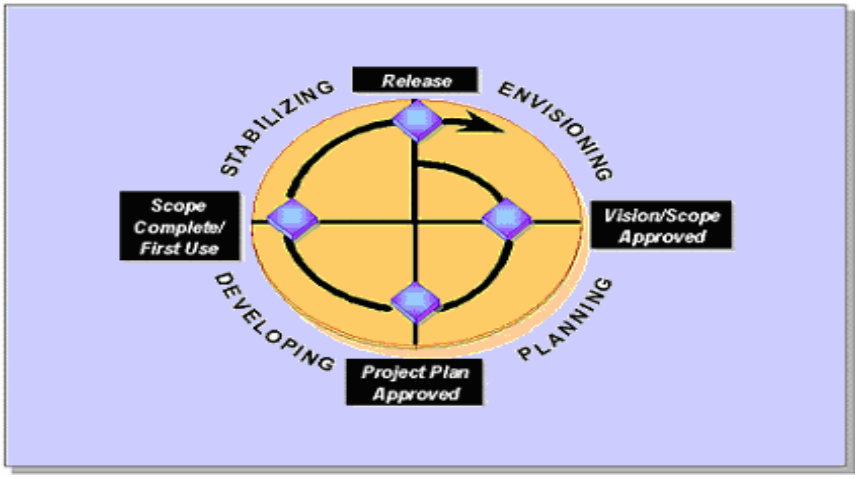
KnowleSys 软件开发过程指南

1. 概述

软件项目的成败并非在于完全的技术层面的考虑上，而是取决于项目本身是否被小心规划，谨慎执行。绝大部分软件项目都可以以一种几乎保证成功的决定性的方式进行。本开发过程就是这样一种决定性的开发方式。我们把软件开发过程看成是一个螺旋上升的有序循环，每一个循环都包括预想，规划，开发与稳定四个过程，每次循环的结束都产生一个新的对外发布的软件版本。

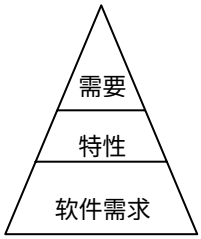


2．各个过程的活动任务

过程	任务		交付内容
预想	组成核心小组		用户原始需求记录 相关软件功能特性收集，相关文档综述 应用模型（业务，用户，逻辑，技术，开发，物理） 项目预想文档（需求规格确认书）
	小组与用户交流		
	项目预想		
	风险估计		
规划	外部设计	结构设计	外部设计规格书 结构，特性，交互，功能 内部设计规格书 技术组件，工具组件，业务组件； 组件协作图与说明； CDM，PDM，StoreProcedure，DataStructure 项目主计划与进度表
		特性设计	
		交互设计	
		功能设计	
	内部设计	组件设计	
		数据设计	
开发	版本 1(主要特性)：编码，测试		源代码，可执行文件，测试报告，用户手册 项目进度追踪表
	版本 2(次要特性)：编码，测试		
	版本 3(高级特性)：编码，测试		
稳定	缺陷修复		软件源代码与文档 软件测试计划与测试用例 软件安装光盘
	零缺陷发布		
	发布候选版本		
	发布最终版本		

3．活动目标与工具箱

活动		目标	方法
需求开发		用户需要软件做什么？ （用例与场景）	用户角色区分 访谈 会议 原型 相似产品研究 问题域研究
外部设计	结构设计	如何搭建系统？ 用户能用软件做什么？ 用户如何使用软件？ 用户使用软件的体验？	原型 用户参与 相关设计研究 用户反馈
	特性设计		
	交互设计		
	功能设计		
内部设计	组件设计	需要哪些组件，如何配合？ 数据结构如何？如何存储？	概念设计（用例与场景） 逻辑设计（组件接口） 物理设计（组件规格）
	数据设计		
开发		推出可执行版本	分层开发 按优先级分段完成 每日构建 反馈 士气激励
测试		保证软件零缺陷	执行测试 功能测试 压力测试 配置测试 可用性测试
项目管理		确定 3W：Who，When，What	公开规划与进度 个人负责 沟通反馈



注：特性是描述功能而不陷入细节的方便方法。

4．各阶段活动向导

需求开发

1．项目背景

客户属于哪个行业？规模？业务？
项目启动原因？必要性？
项目要解决的主要问题？

2．业务模型调查

问题陈述表：对于不同人员，问题，影响，结果，本项目的优点各是什么？
组织业务图：有哪些组织，各种组织开展哪些业务，每种业务有哪些功能
业务数据图：有哪些数据，每种数据的概括属性，各种数据的关系
处理流程图：每种功能的处理流程是怎么样的？
信息流程图：每种处理流程中的有哪些信息，它们是如何流动和改变的？
用户角色图：有哪几种典型参与者？他们的个人信息如何？（年龄，学历，水平）
用户需要表：来自不同角度的不同需要列表
模型总结图：使用用例与场景描述

3．前景讨论

系统框图： 带有确定使用者与内部组成部分的框图
原型列表： 界面原型（总体，输入，输出），交互原型
特性列表： ID，描述，属性（状态，优先级，工作量，风险，兴奋度）

4．拟定软件需求规格说明

使用模版

外部设计

1．结构设计

多层结构中的各种服务如何部署？
如何将系统化整为零？如何让各部分协同工作？

2．交互设计

概念思考： 交互的本质是什么？
行为思考： 怎么样的交互行为才是可理解的？易用的？简单的？
界面思考： 什么样的界面才能满足数据需求？易用的？美观的?
思考原则： 为欢乐而设计，为效能而设计，为人而设计（不是为 CPU 或内存）

3．特征设计

用户想要么样的功能？特性？
它们的优先级如何？开发量如何？兴奋度如何？

内部设计

1．组件设计

在用户服务层中需要哪些组件？它们提供哪些服务接口？如何协作？
在业务服务层中需要哪些组件？它们提供哪些服务接口？如何协作？
在数据服务层中需要哪些组件？它们提供哪些服务接口？如何协作？
从复用的角度出发，在问题域需要哪些组件？它们提供哪些服务接口？如何协作？
从复用的角度出发，在技术域需要哪些组件？它们提供哪些服务接口？如何协作？
从复用的角度出发，如何抽象与划分技术组件，工具组件，业务组件？

2．数据设计

完成每项功能需要哪些数据要素？
如何规范化这些数据要素为一个实体关系图？
如何利用数据库约束来自动实现业务规则？

开发

1． 模块编码

统一的编码风格是什么？
代码的框架确定了吗？
采用基于接口的广度优先编码方法了吗？（先定义多个对象然后逐个实现）
一年以后代码还能被理解吗？

2． 模块测试

开发了测试程序来测试各个模块了吗？

测试

1. 执行测试

所有的界面入口元素都能如期执行吗？

2. 功能测试

每个功能隐藏在界面后的部分实际上都完成了吗？

3. 压力测试

与外部元素的数量有关的模块在该数量很大时还能正常执行吗？

4. 配置测试

在各种正常的可能的软硬件配置条件下，系统可以正常运行吗？

5. 可用性测试

一个不熟悉系统的新用户能够很快把系统用起来吗？
系统是否提供了的用户实际上不需要的功能？
系统是否提供了的用户实际上潜在需要的功能？

6．用户体验测试

实际的用户对于各个视觉界面的感觉（一般，期望，兴奋）？
实际的用户对于各种交互方式的感觉（一般，期望，兴奋）？
实际的用户对于各个功能的感觉（一般，期望，兴奋）？