**政府绩效产品设计**

**- 一次以RUP和软件产品线方法构建的初始产品框架**

**医疗IT事业部**

**作者：王德刚**

目 录

[摘要 5](#_Toc456272687)

[1 工作思路和过程 5](#_Toc456272688)

[1.1. 过程成果物概览 5](#_Toc456272689)

[1.2. 建模工具及多人合作说明 6](#_Toc456272690)

[1.3. 模型和代码下载地址 6](#_Toc456272691)

[2 业务分析 6](#_Toc456272692)

[2.1. 业务参与者 6](#_Toc456272693)

[2.2. 业务范围 7](#_Toc456272694)

[2.3. 业务目标 7](#_Toc456272695)

[2.4. 业务流程 7](#_Toc456272696)

[2.3.1. 制定考核方案 9](#_Toc456272697)

[2.3.2. 指标绩效考核 10](#_Toc456272698)

[2.3.3. 年底绩效汇总 11](#_Toc456272699)

[2.5. 部门及岗位 12](#_Toc456272700)

[2.6. 领域模型 12](#_Toc456272701)

[2.5.1. 指标考核 14](#_Toc456272702)

[2.5.2. 单元年度成绩 14](#_Toc456272703)

[3 产品定义 15](#_Toc456272704)

[3.1. 愿景 15](#_Toc456272705)

[3.2. 产品分类 15](#_Toc456272706)

[3.3. 产品特性（全集） 15](#_Toc456272707)

[3.4. 系统用例 16](#_Toc456272708)

[3.4.1. 参与者 16](#_Toc456272709)

[3.4.2. 用例 16](#_Toc456272710)

[组织绩效-指标考核 17](#_Toc456272711)

[3.5. 词汇表 22](#_Toc456272712)

[4 系统分析 22](#_Toc456272713)

[4.1. 概念结构 22](#_Toc456272714)

[领域层主要概念 23](#_Toc456272715)

[指标考核 23](#_Toc456272716)

[4.2. 用例分析 26](#_Toc456272717)

[关联被考核单元 26](#_Toc456272718)

[考核打分 27](#_Toc456272719)

[查看考核方案执行情况 27](#_Toc456272720)

[查看被考评单元实时分数 28](#_Toc456272721)

[5 系统设计 28](#_Toc456272722)

[5.1. 逻辑结构 29](#_Toc456272723)

[类图 30](#_Toc456272724)

[指标考核 30](#_Toc456272725)

[控制器 30](#_Toc456272726)

[5.2. 用例实现 31](#_Toc456272727)

[关联被考核单元 31](#_Toc456272728)

[考核打分 31](#_Toc456272729)

[查看考核方案执行情况 32](#_Toc456272730)

[启动方案 32](#_Toc456272731)

[5.3. 设计机制 33](#_Toc456272732)

[5.4. 可变性设计 34](#_Toc456272733)

[Domain的可移植性 35](#_Toc456272734)

[5.5. 测试设计 36](#_Toc456272735)

[5.6. 部署模型 36](#_Toc456272736)

[6 系统实现 37](#_Toc456272737)

[6.1. 主要决策 37](#_Toc456272738)

[6.2. 已模拟的场景 37](#_Toc456272739)

[6.2.1. 创建方案 37](#_Toc456272740)

[6.2.2. 创建被考核对象 37](#_Toc456272741)

[6.2.3. 创建指标类型 38](#_Toc456272742)

[6.2.4. 创建指标 38](#_Toc456272743)

[6.2.5. 关联指标和被考核对象 38](#_Toc456272744)

[6.2.6. 启动方案 38](#_Toc456272745)

[6.2.7. 填报 38](#_Toc456272746)

[6.2.8. 打分 38](#_Toc456272747)

[6.2.9. 查看自己的操作记录 38](#_Toc456272748)

[6.2.10. 删除指定考核执行 38](#_Toc456272749)

[6.2.11. 监控考核状态 38](#_Toc456272750)

[6.2.12. 关闭方案 39](#_Toc456272751)

[6.2.13. 登陆 39](#_Toc456272752)

[6.2.14. 生成部门分数 39](#_Toc456272753)

[6.2.15. 获取部门分数 39](#_Toc456272754)

[6.3. 代码风格举例 39](#_Toc456272755)

[启动方案 39](#_Toc456272756)

[Controller 39](#_Toc456272757)

[AppService 40](#_Toc456272758)

[DomainService 40](#_Toc456272759)

[RootEntity 40](#_Toc456272760)

[ChildEntity 41](#_Toc456272761)

[RootEntity与ChildEntity关系 42](#_Toc456272762)

[获得待填报列表 43](#_Toc456272763)

[Controller 43](#_Toc456272764)

[AppService 43](#_Toc456272765)

[DomainService 44](#_Toc456272766)

[Repo 44](#_Toc456272767)

[6.4. 可变性举例 45](#_Toc456272768)

[通过扩展添加审核环节 45](#_Toc456272769)

[通过接口支持考核流程节点处理人指定到部门、单位、角色、以及具体的人 48](#_Toc456272770)

[通过接口汇总多种分数来源的组织单元的年度成绩 49](#_Toc456272771)

[7 目前的问题 50](#_Toc456272772)

[8 附录 50](#_Toc456272773)

# 摘要

本文以政府绩效业务为背景讲解采用RUP和软件产品线方法的执行过程和重要的成果物，旨在说明RUP过程和软件产品线方法用于构建业务线核心软件产品的有效性。目标读者为正在寻找过程和方法构建业务线产品的团队负责人。

# 工作思路和过程

1. 过程成果物概览



过程由业务建模产生系统用例，同时确立产品愿景及产品特性，通过分析设计产生设计模型及其他指导软件开发的重要模型。其中开发模型即为软件产品的初始框架，其中包含了通过上述过程得到的主要业务组件、接口及重要的软件业务实体。以单元测试作为手段构建起多个业务场景的测试用例来测试和描述业务系统的核心行为。

该过程描述的工作发生于先启和精化阶段的早期迭代，由业务架构师和软件架构师共同完成。

1. 建模工具及多人合作说明

模型以Rose作为建模工具，可以通过创建cat文件形成多人共建一个模型的工作模式。合作的形式以场景为分工、以场景中共同使用的实体、岗位、组件等为合作内容。在模型中一张图有一张图的价值，一张图不能呈现所有关注，一件复杂的事儿是需要多个步骤，产生多张图才可描述清楚的。

1. 模型和代码下载地址

<https://github.com/jdepend/gp>

其中模型在doc文件夹下。

工程需要maven编译和运行，需要JDK1.8+。

# 业务分析

**业务建模的职责是确定待建软件系统的职责（做什么）**，这种职责是通过业务流程分析得到的，可以通过在描述业务用例实现时引入待建软件系统来建模其职责，以及交互的岗位和其他系统（业务工作者）。

## 业务参与者

业务参与者就是业务的（主要或最终）受益人，它往往是业务流程的发起者。它对确定业务的边界，识别业务内的流程，思考改进流程的方向都极有帮助。

本业务没有分析出明确的业务参与者，使得很多对业务流程的分析都缺乏改进的动力。

## 业务范围



政府绩效中组织绩效一般都是以年为周期的。

## 业务目标

业务目标是改进业务流程、探索待建软件系统职责的重要线索。



## 业务流程

为了做业务流程分析，首先需要识别业务流程。业务流程的识别线索来自于业务流程的服务对象（业务参与者）。



1. 制定考核方案



1. 指标绩效考核



1. 年底绩效汇总



## 部门及岗位



## 领域模型

业务建模过程中需总结业务内相关的业务概念，以及它们之间的关系，这些可以输入到软件分析。一般来说，一个业务流程会对应一个核心的业务实体，描述出该业务实体的状态图是重要的。另外，通过实体关系及其核心业务实体的属性变化应该可以解释业务流程执行的效果。在识别业务实体时，在单一业务步骤中或两个业务步骤间交互的信息应该在后续的分析模型中进行建模。



1. 指标考核



1. 单元年度成绩



# 产品定义

1. 愿景

解决政府单位在实施绩效工作中缺少XXX、YYY的问题；

相对于a（类似产品），我们的优势是：

1. 有系统化分解指标的业务能力；
2. ……
3. 产品分类
4. 针对不同类型的客户问题归集产品特性，形成不同的产品，形成不同的价格；
5. 对客户问题的分类可以来源于客户绩效业务发展的阶段，不同的阶段有不同的问题；

|  |  |  |
| --- | --- | --- |
| 绩效业务发展的阶段 | 遇到的主要问题 | 需要的产品特性 |
| 初始阶段（半手工） | 指标不体系化、缺少周期性考核指标的设计 |  |
| X阶段 | 却少具体指标的考核办法或没有合适的软件系统处理 |  |
| Y阶段 | 不能够利用既往考核周期的数据、不能够对下一考核周期的考核工作作出量化指导 |  |
| Z阶段 | 却少绩效工作与日常工作的有效衔接 |  |

1. 产品特性（全集）

1、内置政府单位中组织绩效、个人绩效两块业务，并能够实现其双向互动；

2、支持政府单位内部多级组织结构（如：省、市、县区）下的绩效业务；

3、通过内置应用和外部API的形式实现了年底绩效成绩的多种来源（指标考核、测评、公众评议、领导评价、重点项目执行情况、加减分等）；

4、实现了考核指标的多级灵活设置；

5、支持对考核指标不同频次的考核（年、半年、季、月等）；

6、支持在不同考核频次类型上的进一步定制（删除指定考核频次、设置频次权重及计分规则等）；

7、内置针对不同考核指标的多种计算模型（直接扣分法、累计计分法、数量递减法等）；

8、实现了考核指标向被考核对象上进行计分时的不同规则（加权平均、XXX）；

9、支持不同角色共建指标模型，并提供灵活的审批流程；

10、支持填报和打分过程中灵活的审批流程；

11、支持实时查看不同对象（方案、被考核对象、被考核对象上的指标、指标上的不同频次）上的考核分数情况；

12、能够根据考核结果为被考评对象生成改进建议；

13、支持在创建考核方案时根据之前考核周期生成的数据来提供参考考核指标、目标值等；

14、能够对比多个考核周期不同方案、不同被考核对象的考核结果；

15、提供外部API的方式实现对一个被考核对象上考核指标计分规则中原始分数的采集；

1. 系统用例

系统用例是衔接业务建模和后续分析设计的桥梁，系统用例由于是业务建模的产出物，于是将其分配给业务建模人员来创建更合理。

1. 参与者



1. 用例

每一个系统用例应描述出其核心操作步骤及其软件要实现的重要规则。

#### 组织绩效-指标考核



##### 创建年度考核方案



##### 制定共性指标



##### 关联被考核单元



##### 查看考核方案执行情况



1. 词汇表

当没有业务建模或在业务建模过程中没有识别业务实体的情况下，可以通过创建【词汇表】的形式来描述其软件需求中涉及的核心业务概念，这些信息会在接下来的分析中使用到。

# 系统分析

分析模型是以重要的系统用例作为分析对象，通过描述时序图的手段来做用例分析构建起来的。分析模型需对系统中1/4的有代表性的系统用例进行分析才可以得出一个能够满足多个用例场景中的分析模型。

1. 概念结构

分析包的划分除了要考虑分析类的耦合内聚关系外，分离易变和稳定的分析类也是重要的划分思路。分析包的划分是组件模型的概念性输入，也是其最核心的输入，它从业务层面描述出了系统的概念性组成。



### 领域层主要概念

需求本身提出的可变性要求，以及需求中不明确的地方都是需要抽象思考，通过创建抽象的分析类来容纳变化。这时候这些抽象类的稳定性变得更加重要，需要着重思考和交流。

#### 指标考核



##### 考核方案状态图



##### 计划考核执行记录状态图



1. 用例分析

### 关联被考核单元



### 考核打分



### 查看考核方案执行情况



### 查看被考评单元实时分数



# 系统设计

设计模型要影响代码，通过工具导出成最早的代码框架来影响之。设计模型的稳定性比分析模型弱，其组件关系、接口，以及组件内部重要的设计类值得更新，其他细节的设计类不必保持与代码的一致性。

设计模型构建应遵守的原则、模式和经验：

* 数据库表应私有化，并视其为不稳定的设计元素；
* 组件间通过接口访问，强调其接口的稳定性；
* 通过参数的接口化可增强接口的稳定性；
* 对于操作频繁的Entity，应通过冗余来减少创建实体时表连接的开销；
* Entity间的关系采用manytoone和onetomany来构建更合适；

对于打算在其他环境下还要使用的组件，其接口、移植性、扩展性、错误处理等方面要强化其质量；

1. 逻辑结构



### 类图

#### 指标考核



#### 控制器



1. 用例实现

### 关联被考核单元



### 考核打分



### 查看考核方案执行情况



### 启动方案



1. 设计机制

在构建设计模型时须确定核心的架构风格（如EDA）、组件内结构范型、错误处理的模式等等，这些工作需要在设计机制章节描述清楚。

1. 可变性设计



### Domain的可移植性



1. 测试设计



1. 部署模型

云计算的风靡使得单一计算模型（单应用、单数据库）已经过时，而分布式计算、微服务等计算方式越来越流行。构建一个由单应用向分布式计算过度的方案成为一件大事，挑战不小。最大的挑战不是数据一致性的保证，而是演变过程中设计和编程习惯的改变。

# 系统实现

实现模型就是代码框架，包括了开发期的工程结构、依赖的第三方框架版本、运行的具体方式等等。对于java路线，推荐使用SpringBoot+Maven来构建。对于在设计上强调的组件间的单项关系可以通过在maven环境下创建多个module来约束。

1. 主要决策

1、采用eclipse\jdk8\maven\springboot构建；

2、将应用划分为platform、domain、application、listener、portal、mobile、rest、test八个工程；

3、包结构以com.rofine.gp开始；

4、view采用springboot推荐的thymeleaf构建；

5、持久化采用springdata；

6、主键生成采用uuid；

7、核心对象关系采用manytoone和onetomany构建（采用lazyload策略）；

8、数据库采用mysql；

9、采用Repository模式，通过在IdEntity中调用平台中的ApplicationContextUtil获取Repo；

10、采用可捕获异常来定义业务异常，并建立异常树来分类业务异常；

11、通过实体关系获取实体将不再通过Repo来获取，只通过实体上的引用获取；

12、当通过定制条件获取实体时使用Repo；

13、除了创建实体外，其他操作尽量在DomainService后通过Repo.fin实例化实体；

14、领域逻辑尽量附着在实体上，其次是DomainService；

15、非领域逻辑不允许被DomainService和Entity依赖；

16、领域事件由DomainService抛出；

17、通过@Async可实现异步监听；

18、领域模型中错误处理代码应为正常代码量的n倍；

19、领域模型需配备n倍场景的单元测试代码；

1. 已模拟的场景
2. 创建方案

/scheme/create GET/POST

1. 创建被考核对象

/scheme/{schemeId}/object/create GET/POST

1. 创建指标类型

/scheme/{schemeId}/target/type/create GET/POST

/scheme/ {schemeId}/target/type/{parentId}/target/type/create GET/POST

1. 创建指标

/scheme/{schemeId}/target/create GET/POST

1. 关联指标和被考核对象

/scheme/{schemeId}/target/{targetId}/2object GET/POST

1. 启动方案

/scheme/{schemeId}/start POST

1. 填报

/ scheme/{schemeId}/fill GET/POST

1. 填报审核（可选操作）

/ scheme/{schemeId}/fill/audit GET/POST

1. 打分

/ scheme/{schemeId}/evaluate GET/POST

1. 查看自己的操作记录

/ scheme/{schemeId}/operated/list GET

1. 删除指定考核执行

/ scheme/{schemeId}/execute/{ executeId}/delete GET/POST

1. 监控考核状态

/ scheme/{schemeId}/monitor GET

1. 关闭方案

/ scheme/{schemeId}/close GET

1. 登陆

/login/user/{userId}/org/{orgId}/dept/{deptId}/role/{roleIds} GET/POST

1. 生成部门分数

/ unit/score/year/{year}/create POST

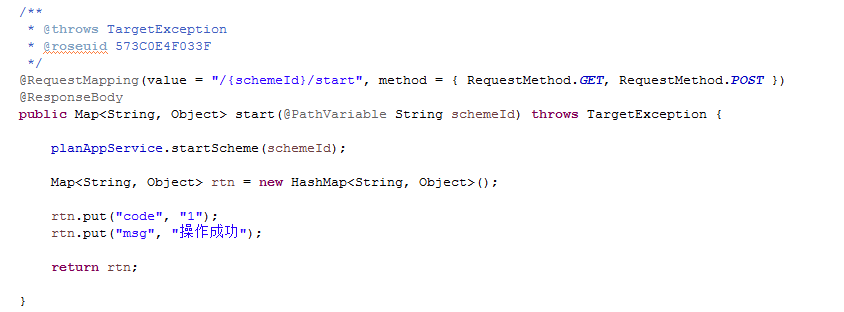
1. 获取部门分数

/ unit/score/year/{year}/get GET

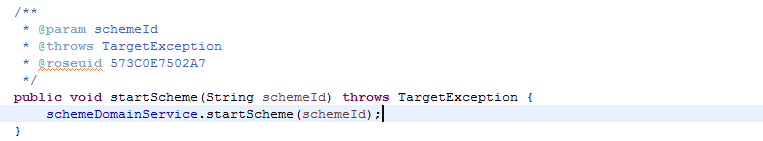
1. 代码风格举例

### 启动方案

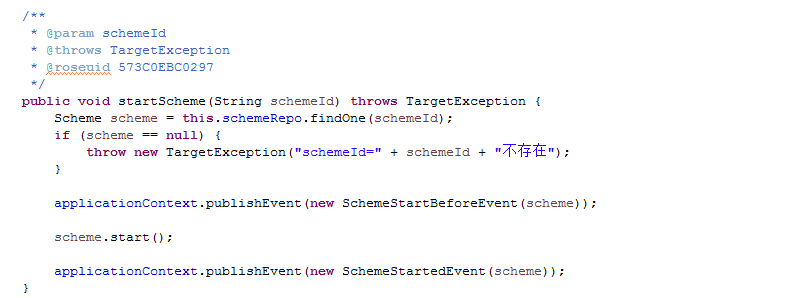
#### Controller



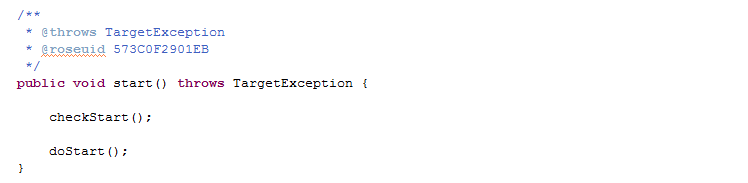
#### AppService



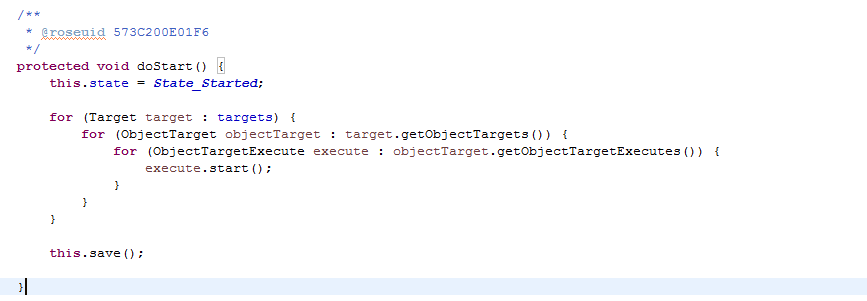
#### DomainService



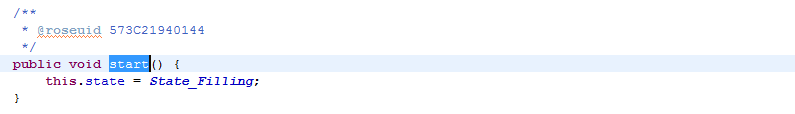
#### RootEntity





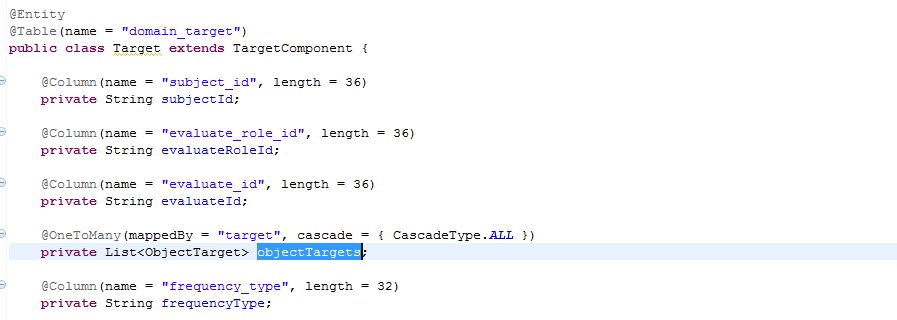


#### ChildEntity



#### RootEntity与ChildEntity关系

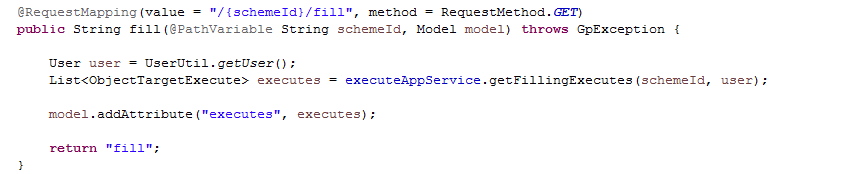






### 获得待填报列表

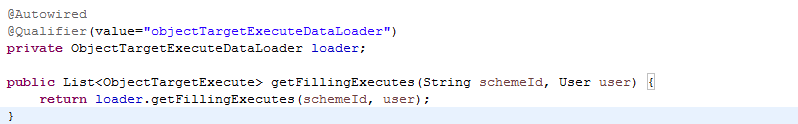
#### Controller

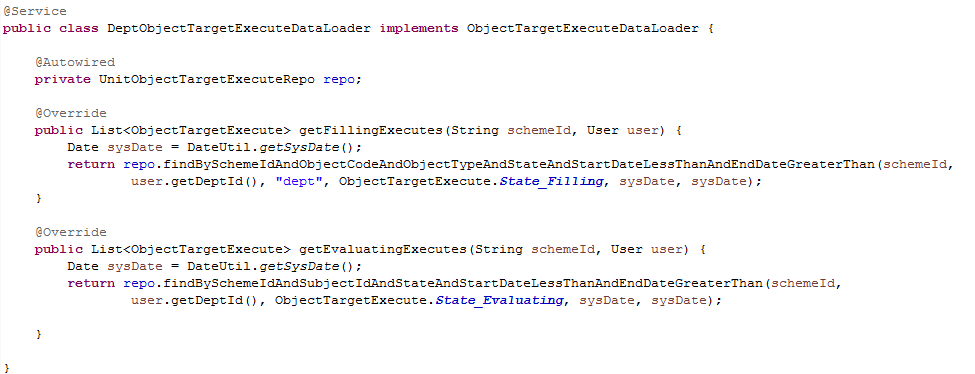


#### AppService

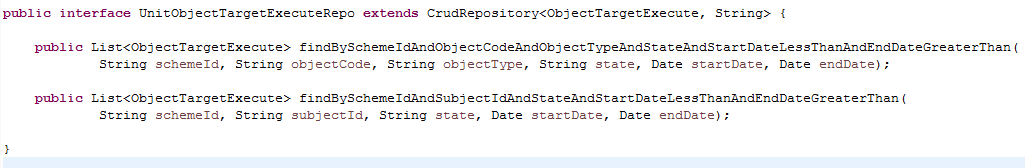


#### DomainService



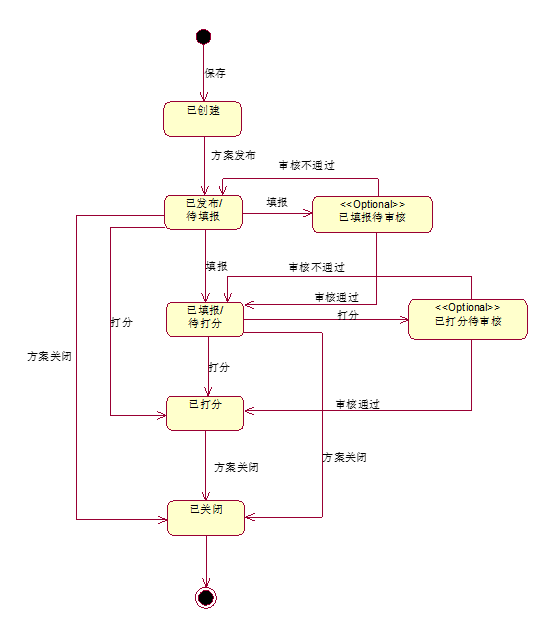


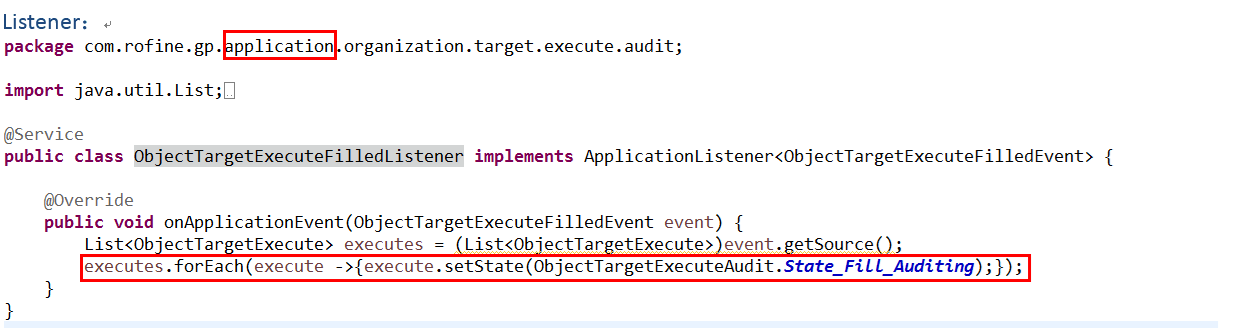
#### Repo

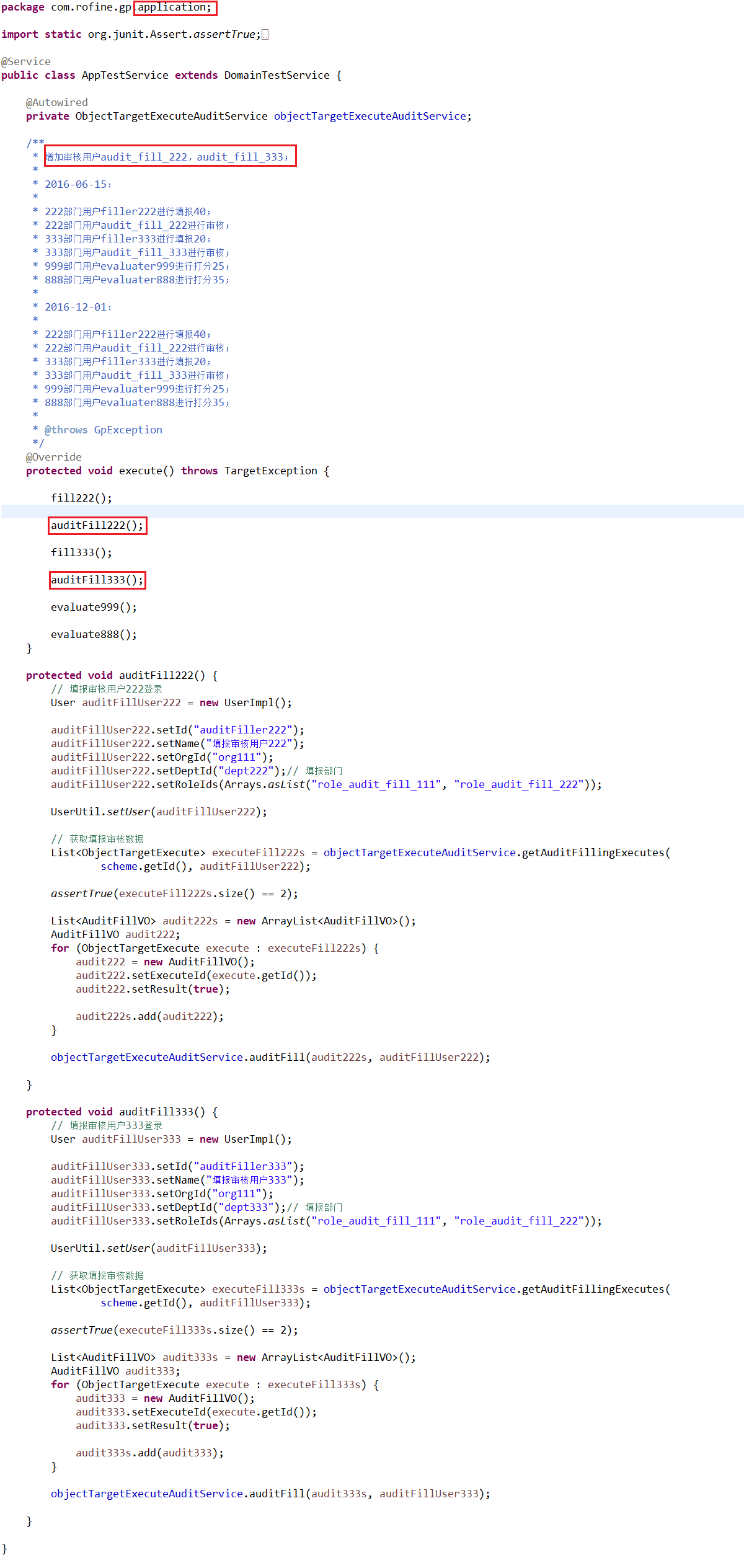


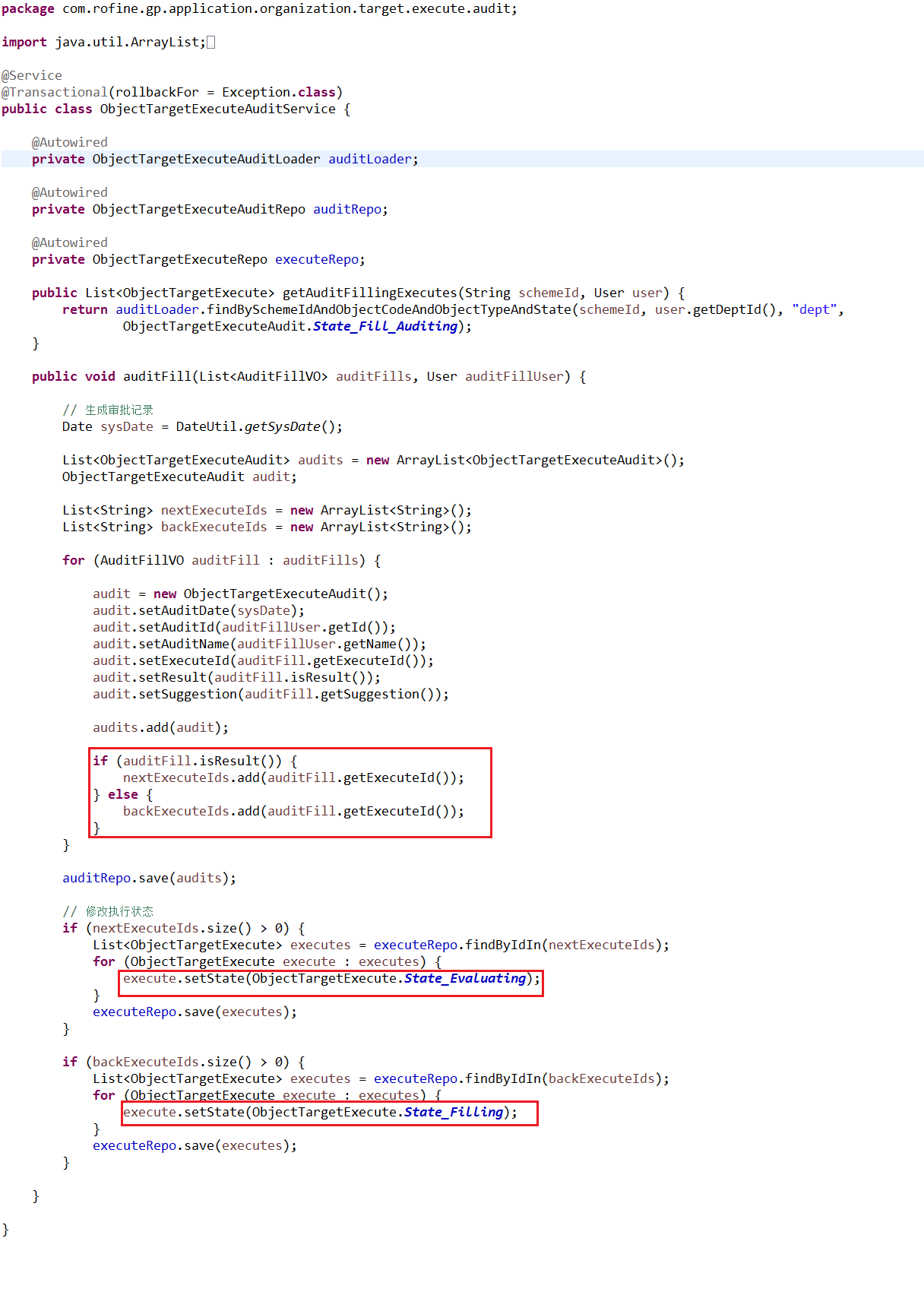
1. 可变性举例

### 通过扩展添加审核环节

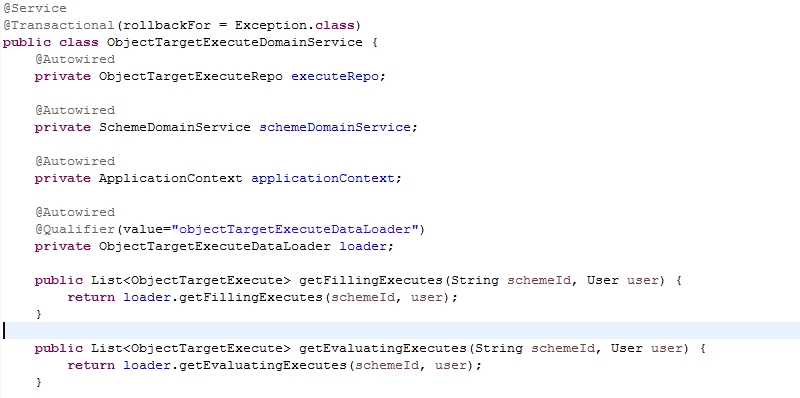


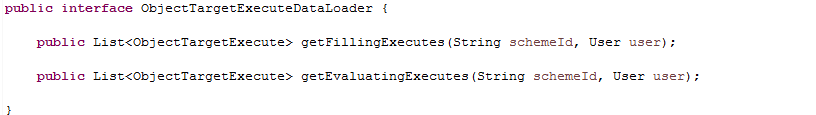


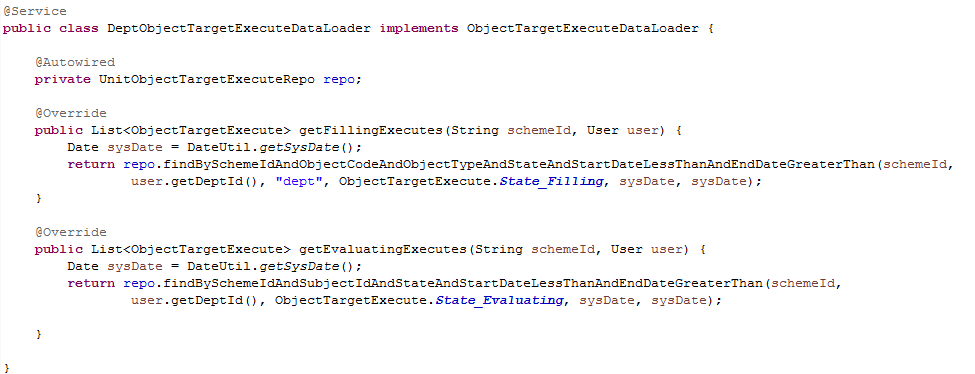


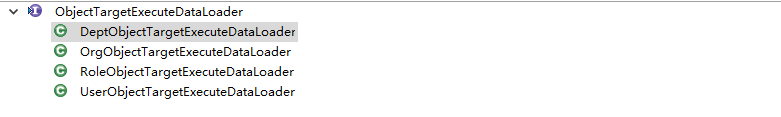


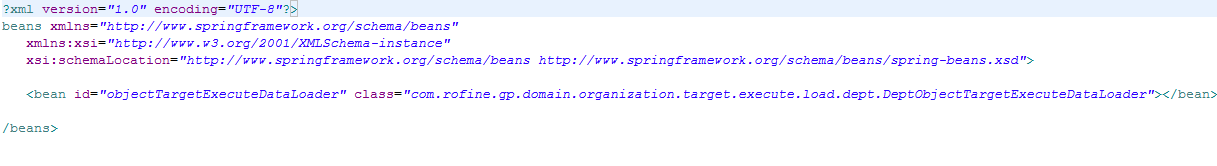
### 通过接口支持考核流程节点处理人指定到部门、单位、角色、以及具体的人





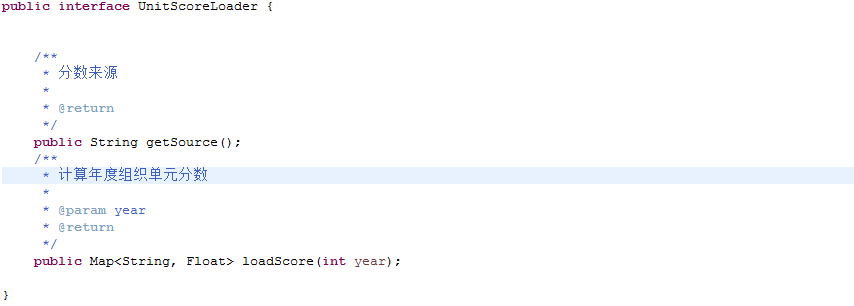


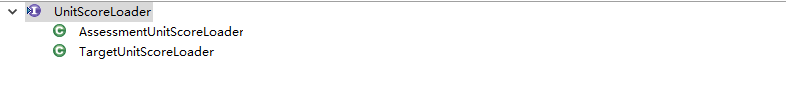




### 通过接口汇总多种分数来源的组织单元的年度成绩









# 目前的问题

缺少对业务系统性的理解，尤其是对客户问题的总结；业务分析部分的内容实际上是基于软件使用手册阅读的，使得多数在业务分析中出现的内容都可以映射为软件职责，实际的业务分析要比这复杂得多。

# 附录

问题备忘录

1. 产品范围是组织绩效、个人绩效，还是包含两者的政府绩效？
   1. 组织绩效和个人绩效的共同点和差异点有哪些？二者是否有做到一个产品的可能？业务和商业理由充分？
      1. 个人绩效受领导的管理思路影响较大，我们是否有影响领导管理思路的能力呢？
   2. 最终的产品输出物和内部的产品划分是两件事
2. 重点项目和领导驾驶舱的管理模式与组织绩效有较大差异，启动流程是否也由绩效办启动？
3. 一个政府单位的绩效业务，单位以外的人谁关心？参与到流程中吗？
4. 指标分解的职责是否可以固定？（共性指标由绩效办做，个性指标一部分由绩效办设计大的指标项，由被考核部门细化分解，另一部分完全由被考核部门设计）
5. 填报和打分是否一定配置审批流程是否应该成为一个可变点？
6. 年度部门成绩比较接近，谁不愿看到？
7. 对单位的指标考核是在一个上级单位系统上完成还是跨系统完成呢？
8. 对多组被考核单元的考核工作是通过创建多个考核方案来完成还是在一个考核方案下建立多个被考核单元分组来完成考核工作呢？