

# 需求文档结构化处理

# 模型一致性分析报告

**团 队 名 称 TEAM**

**团 队 成 员 BY1817164 孙竖敬**

**团 队 成 员 SY1806116 赵健宏**

**团 队 成 员 SY1806611 张鹏程**

**团 队 成 员 ZY1806106 刘 良**

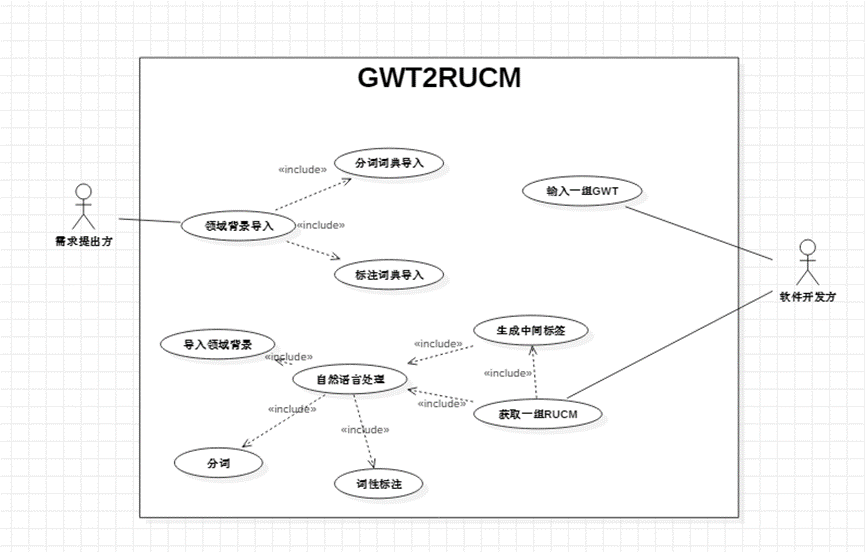
**2018年12月**

在本次项目中，我们主要建立了UML（用例图、类图、时序图、活动图、状态图、组件图）、RUCM、架构图和OCL模型。在模型的一致性方面，我们首先保证了UML的各个模型之间的一致性，架构图与UCL的一致性，然后保证RUCM与用例图、活动图的一致性，OCL与架构图、类图的一致性。

## UML模型内部的一致性：

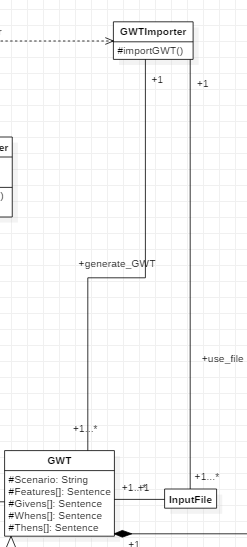
* 用例图

用例图描述的是系统所需要实现的功能，参与者与系统的交互关系。本项目中，系统对外部的实现的功能是：获取领域背景、获取GWT文档、输出产生的RUCM文档。

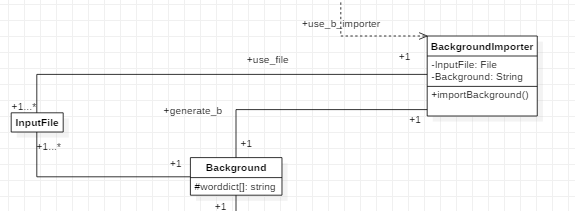
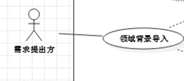


* 类图

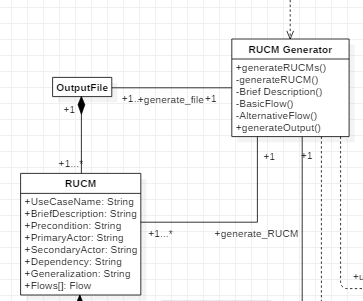
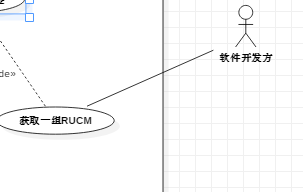
类图着重显示了模型的静态结构，描述了模型中存在的类、类的内部结构以及与它们与其他类之间的关系，而这些类最终体现的功能，则必须是用例图中描述的。



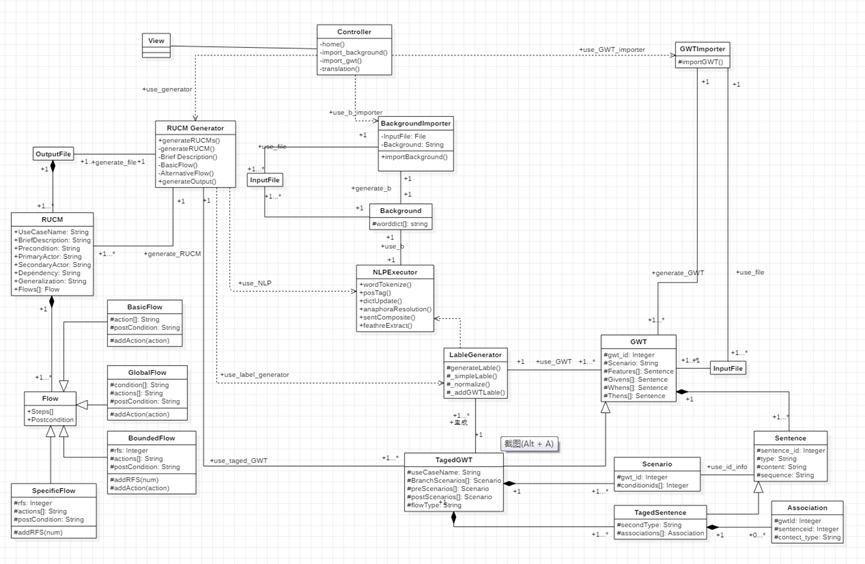
这是类图的GWTImporter类、GWT类、Inputfile类来对应用例图的输入一组GWT描述功能。



这是类图的BackgroundImporter、Background、InputFile类，与用例图中的领域背景导入相对应



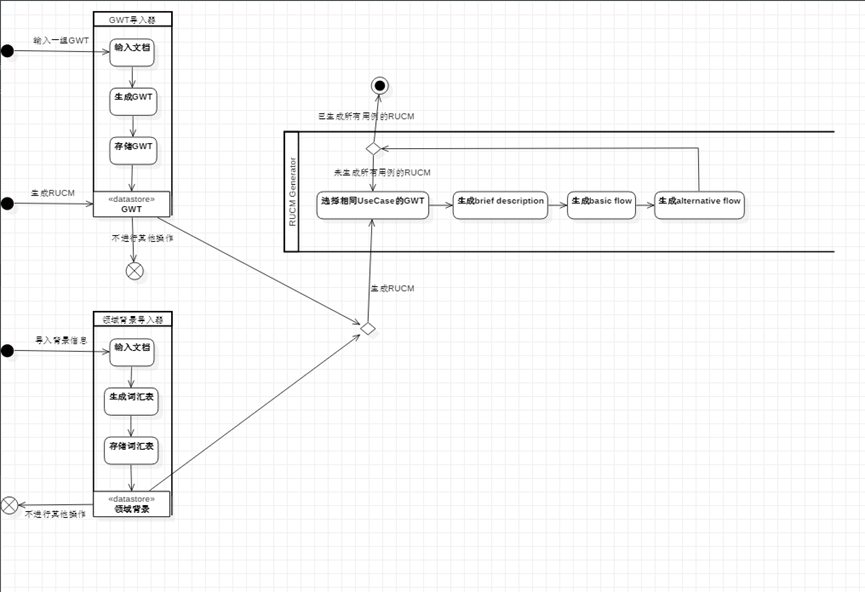
这是类图中的OutputFile类、RUCM Generator类、RUCM类来对应用例图的输入



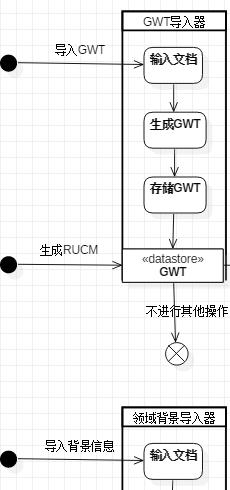
这是最终版本的类图。

* 活动图

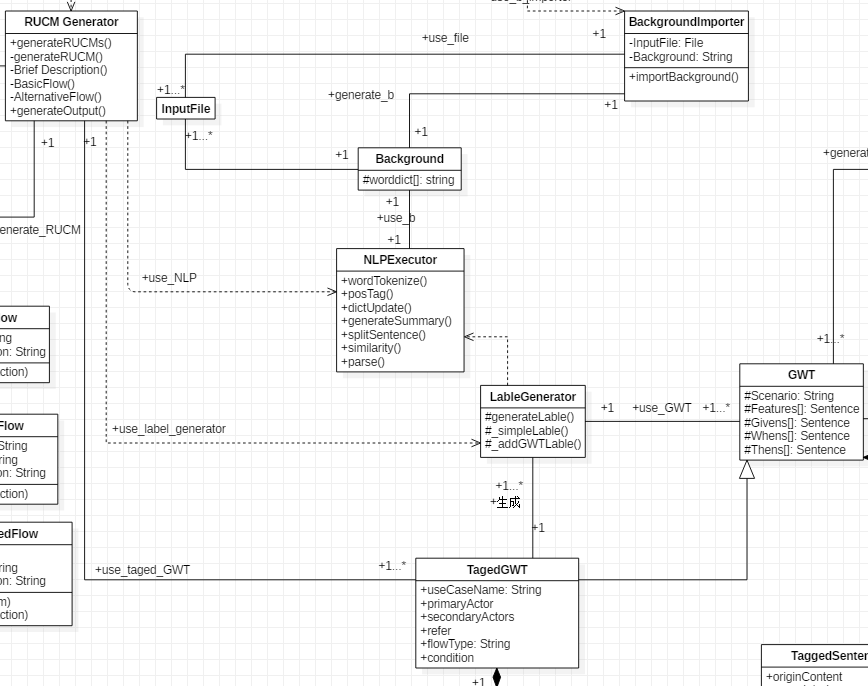
活动图：表述过程基理、业务过程以及工作流的技术。它可以用来对业务过程、工作流建模，也可以对用例实现甚至是程序实现来建模。我们用活动图来表示文档转化的具体流程。由于活动图是动态描述流程，故在此部分仅仅做简单的一些类图功能上的对应，重点动态行为的对应则放在活动图、时序图中阐述。

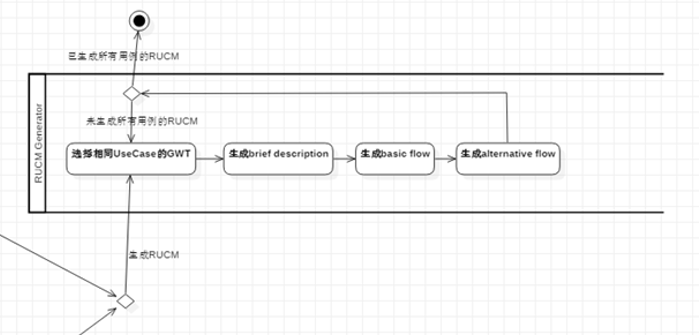


这是最终版本的活动图，可以看到，活动的流程很清晰，分为3个大的流程，当前两个过程处理完毕，才能够获得足够信息，支持第三部分完成最终的处理。



以上是三个初始状态，分别对应着用例图中的三个功能,由使用者进行命令的发布，当三者条件具备之后，方能进行正确的文档转换。

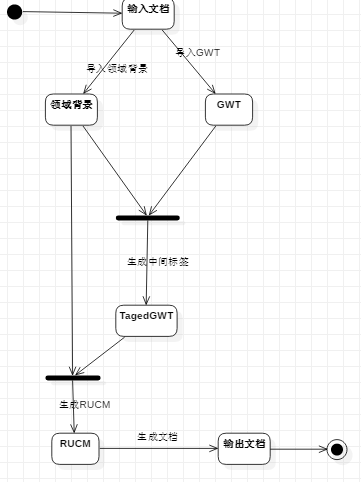




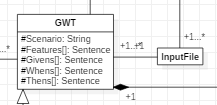
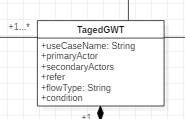
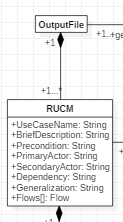
以上是类图中的生成RUCM所涉及到的类和活动图中的RUCMGenerator，可以发现，活动图并没有详细地描述生成RUCM的具体细节，而是将类图中产生RUCM的过程用RUCMGenerator简要地描述出来，具体实现是由类图对应的类所实现，活动图没有必要，也不应该设计其中的实现细节，侧重的是流程的描述。

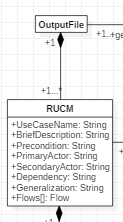
* 状态图

由对象属性定义，对关键对象行为的建模，基于事件反应的动态行为，是描述一个实体基于事件反应的动态行为，显示了该实体如何根据当前所处的状态对不同的事件做出反应。



这个状态图主要反映的是原始的文档在系统进行处理时状态的转换。

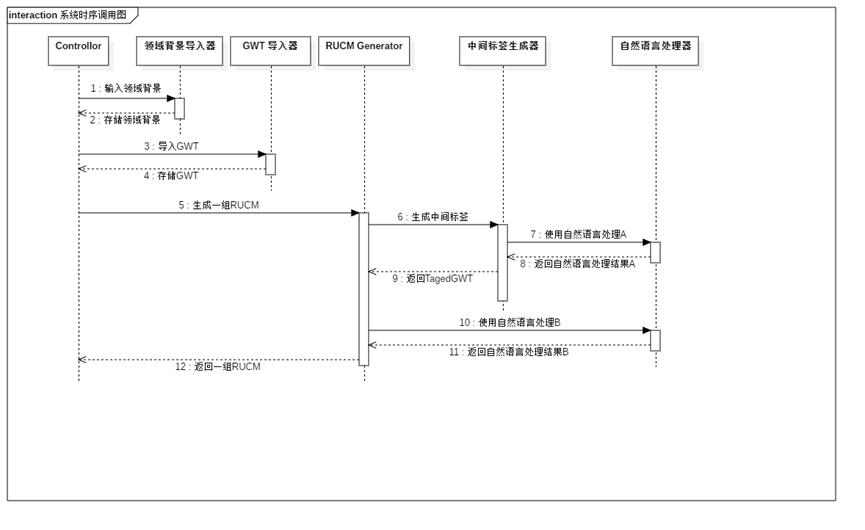
  



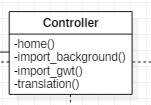
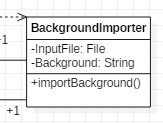
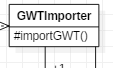
状态图中文档所处的每一个状态，类图中都有与之相对应的类。动态转化流程则与活动图的描述相一致。

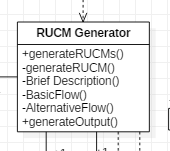
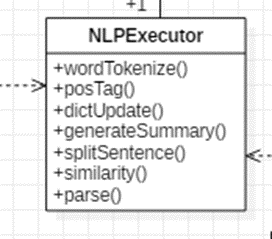
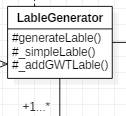
* 时序图

时序图：时序图（Sequence Diagram），又名序列图、循序图，是一种UML交互图。它通过描述对象之间发送消息的时间顺序显示多个对象之间的动态协作。

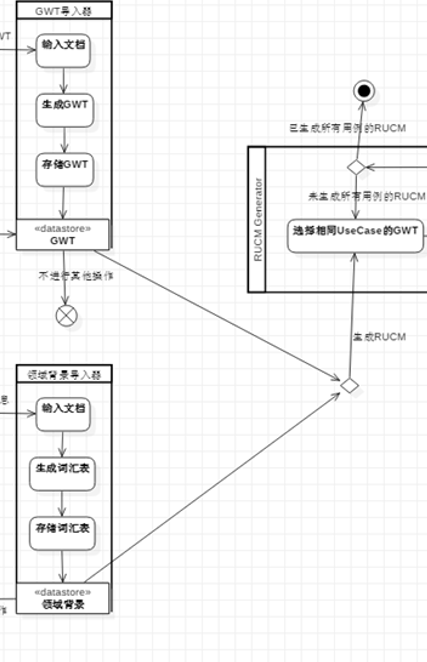


这是最终版本完整的时序图，描述的是系统中的各个对象按照时间的顺序交换信息，以完成文档的转换。

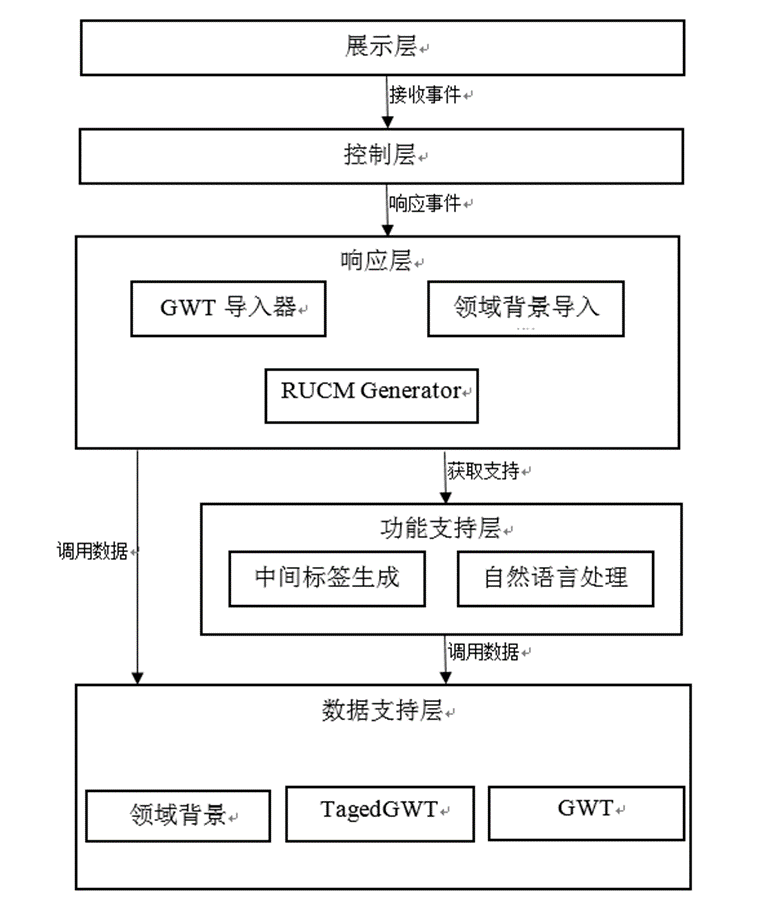
对于时序图中的每一个对象，类图中都有相应的对象与之对应，互相协作的动作，则与类图中的类的方法相对应。



活动图中虽然描述了软件运行的主要流程，但是并没有对对象间的协同作用的时序作严格的描述，活动图表明GWT导入完毕、领域背景导入完毕后开始生成RUCM，而在时序图中，对GWT导入、领域背景导入处理的流程规定了时序，而将活动图中条件达成这一动作隐藏在不同对象协作的时序之中。故而从内在逻辑上来讲，这几个描述系统运行的动态模型之间具有一致性，同时，也和已建立的静态模型具有一致性，故而最终确保了这些模型总体上的一致性。、

* 架构图

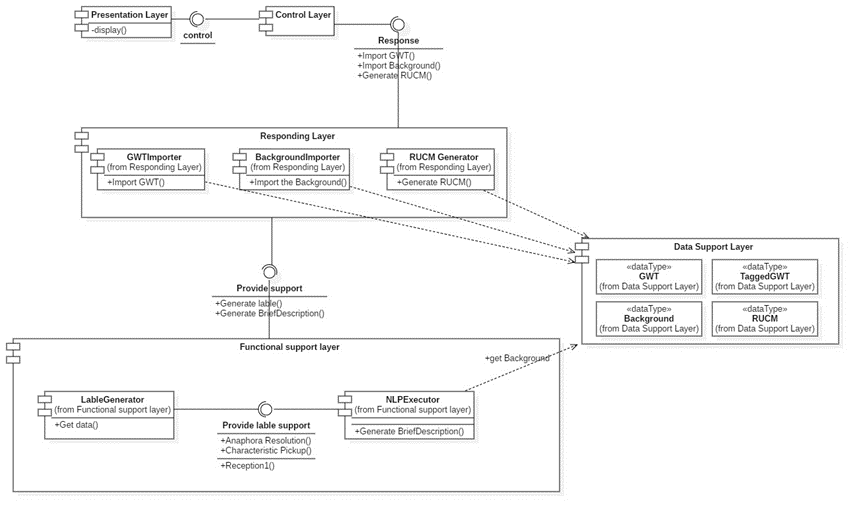
架构图：软件架构是一系列相关的抽象模式，用于指导大型软件系统各个方面的设计。软件架构是一个系统的草图。软件架构描述的对象是直接构成系统的抽象组件。各个组件之间的连接则明确和相对细致地描述组件之间的通讯。在实现阶段，这些抽象组件被细化为实际的组件，比如具体某个类或者对象。在面向对象领域中，组件之间的连接通常用接口\_（计算机科学）来实现。



架构图是我们基于已经建立的模型，结合软件的实际使用场景，进行层次上的设计，具体的细节体现在组件图中，与之前模型的一致性不在赘述，主要通过组件图来说明。

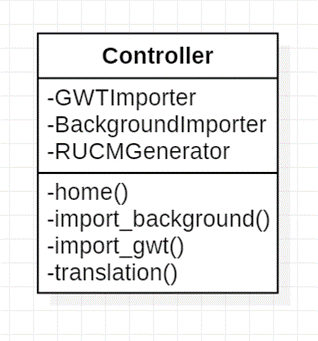
* 组件图

组件图又称为构件图(Component Diagram) 。组件图中通常包括组件、接口，以及各种关系。组件图显示组件以及它们之间的依赖关系，它可以用来显示程序代码如何分解成模块或组件。

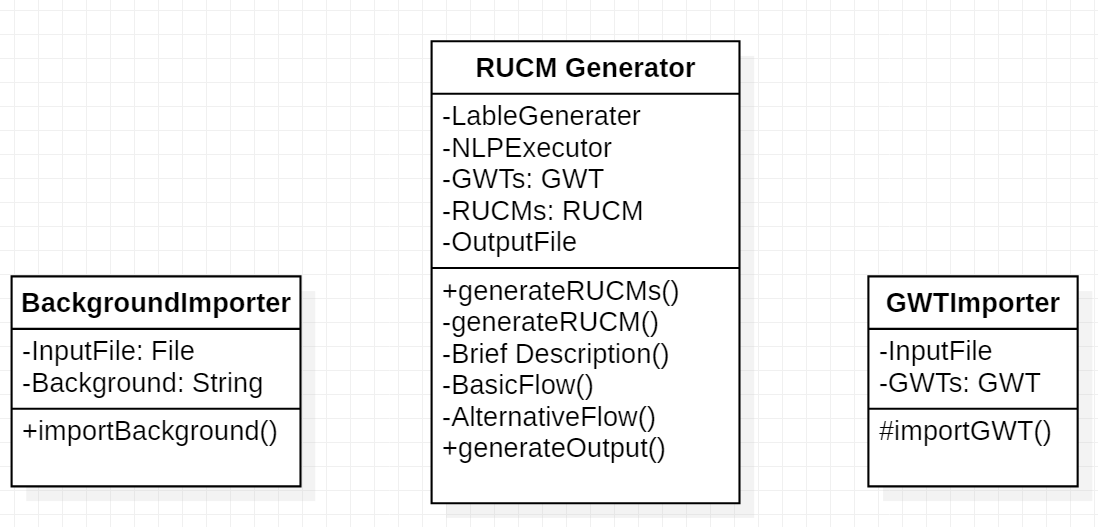


这是最终的组件图，在层次结构上，组件图与架构图保持一致，因为组件图就是按照架构图来设计，并且将类图中的类按照组件图不同功能的组件进行对应，并且对不同组件之间的联系、组件模块之间的联系，用接口描述的方式详细给出。以下则是组件图与类图的详细对应：

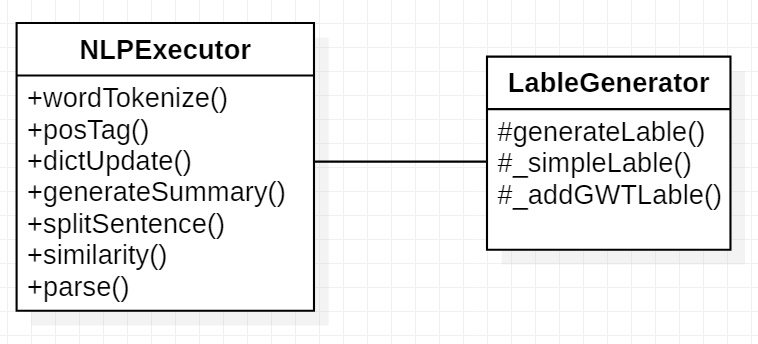
控制层：



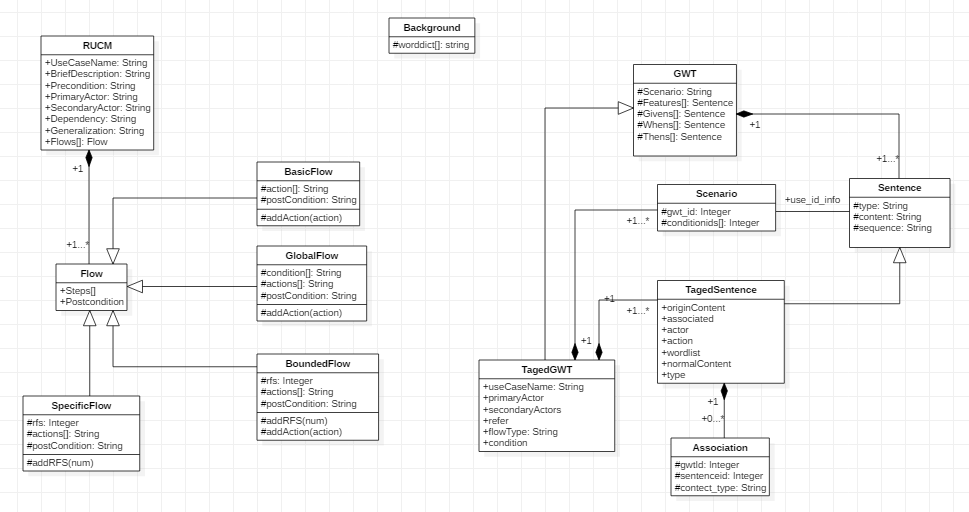
响应层：



功能支持层：



数据层：



## RUCM

RUCM是基于UML的用例图的一种建模方法，它有一套定义良好规则和模板来对用例图进行详细、精确的描述。在与用例图保持一致的情况下，RUCM所描述的场景交互的流程应当与软件内部处理逻辑保持一致。

以下是本项目所建立的三个RUCM模型，分别和用例图中三个用例相对应：







## OCL

OCL是对象约束语言，主要是根据组件图和类图进行建立模型的约束，因此从内在上就与类图、组件图保持了一致性，并且不涉及动态的程序执行流程，所以也没有必要进行与其他模型一致性的检查。