阶段性需求阐述：2018/11/17

目录

[阶段性需求阐述：2018/11/17 1](#_Toc530434298)

[ 需求阐述 2](#_Toc530434299)

[.1 需求描述 2](#_Toc530434300)

[.2 需求规约 2](#_Toc530434301)

[.2.1 输入规约 2](#_Toc530434302)

[1.2.2 输出规约 3](#_Toc530434303)

[.3 需求目标 4](#_Toc530434304)

[.3.1 功能性需求 4](#_Toc530434305)

[.3.2 非功能性需求 4](#_Toc530434306)

[.3.3 系统质量判断标准 5](#_Toc530434307)

[2. 领域术语说明 6](#_Toc530434308)

[2.1 GWT 6](#_Toc530434309)

[2.2 RUCM 7](#_Toc530434310)

[2.3 机器学习相关术语解析 7](#_Toc530434311)

[2.4 自然语言处理相关术语解析 8](#_Toc530434312)

[3. 系统模型描述 10](#_Toc530434313)

[3.1 用例图 10](#_Toc530434314)

[3.2 类图 11](#_Toc530434315)

[3.3 文件状态图 12](#_Toc530434316)

[3.4 系统时序图 13](#_Toc530434317)

[3.5 系统活动图 14](#_Toc530434318)

[4. 系统架构设计 15](#_Toc530434319)

[4.1 中间标签设计 15](#_Toc530434320)

[GWT->中间标签规则： 16](#_Toc530434321)

[中间标签->RUCM规则： 17](#_Toc530434322)

[4.2 系统架构图 18](#_Toc530434323)

# 需求阐述

给定自然语言文档，设计一套标签，手工嵌入到需求文档中，使其成为流程式的需求。并使用机器学习方法来自动插入标签。然后从根据标签来提取信息，形成结构化的需求。

## 需求描述

现明确，给定的自然语言文档按照GWT形式进行描述，产生的结构化需求描述遵循RUCM格式并组成完整文档。由于这两种文档并不存在严格的一一对应关系，所以需要一组中间标签，从而实现GWT到中间标签，中间标签到RUCM的平稳转化。在此转化过程中，我们会使用自然语言处理以及机器学习相关技术。

## 需求规约

在这里，我们将对本次需求的输入和输出进行一些约束，明确输入和输出要求，从而保证系统第一版本能在规定时间内完成。

### 输入规约

本系统通过分析，具有两种输入文件，分别为GWT输入文档与领域背景输入文档。GWT输入文档是本系统的核心文档，用于转化成为RUCM。而领域背景则是作为转化过程中需求的补充文档，用于自然语言处理过程模型的建立与修改。

#### GWT规约

GWT输入文档由每一个独立的GWT所组成。存储方式与格式在当前阶段尚未确定。接下来，本文将对单个GWT的格式进行约束。

1. Given 完整表述当前状态，目前规定当前状态由原子状态组成，每个原子状态均为不能再通过自然语言处理或模式匹配继续拆分的状态。
2. When中每行是同步进行的动作
3. When中一行中每个分句主谓结构完整，且表述清晰。
4. When中每行在当前状态下全部都要执行，且仅执行一次。（另一种表达方式：GWT 中不存在任何分⽀，即每个输⼊的 GWT 只能有⼀个确定的流程和结果。）
5. Then完整表述结束状态，目前规定结束状态由原子状态组成，每个原子状态均为不能再通过自然语言处理或模式匹配继续拆分的状态。
6. Then与Given中的状态不存在判断语句，均为陈述句表述，或为肯定句，或为否定句，或为环境描述。
7. Given，When，Then中均可能存在关键词。关键词为RUCM限制规则的R17~R20、R23~R25、Use Case Name和GLOBAL。例如在全局分支测试时，Given中存在GLOBAL关键词。在When的action中，分支检测会在最后出现ABORT关键词。

#### 领域背景输入文档规约

领域背景输入文档应为文档类文件，其由单个单词组成，单词与单词之间需要可以使用模式识别用于区分的标识。

### 1.2.2 输出规约

即RUCM规约，因为最后的输出文档是由RUCM所组成。简单可以描述为RUCM的26种语法限制。详细描述如下：

1. 基本信息
   1. 用例名称：
      1. 描述：以动词开头的用例名称
      2. 格式：动词短语，动词+名词
   2. 简要描述：
      1. 描述：简要概括用例
      2. 格式：一句话或多句话。
   3. 前置条件：
      1. 描述：本用例执行所需要的条件（what should be true）
      2. 格式：一句话或多句话。
   4. 主要活动者：
      1. 描述：本用例的发起者
      2. 格式：名词
   5. 次要活动者：
      1. 描述：完成用例提供的服务系统所依赖的其他活动者
      2. 格式：名词或“None”
2. 关系描述：
   1. 依赖关系：
      1. 描述：包含和扩展其他用例
      2. 格式：
         1. “None”或
         2. 关键词”INCLUDE USE CASE”/”EXTENDED BY USE CASE” + 其他用例名
   2. 泛化关系
      1. 描述：对其他用例的泛化
      2. 格式：“None”或？？？（暂时没有实例和关键词）
3. 基本事件流：
   1. 步骤号：
      1. 描述：顺序排列的数字
      2. 格式：“STEP”+数字
   2. 事件流：
      1. 描述：主要的“成功”路径的说明
      2. 格式：
         1. <step>：一行
         2. <action>：简单句，主语+谓语（+宾语），不能出现代词、否定词
         3. <step>：(the system) VALIDATES THAT <condition>为一句话
         4. <step>：<action> MEANWHILE <action> | MEANWHILE <action>
         5. <step>：DO <steps> UNTIL <condition>
            1. steps占一行，condition为一句话
         6. IF <condition> THEN <steps> ENDIF
         7. IF <condition> THEN <steps> ELSE <steps> ENDIF
         8. IF <condition> THEN <steps> ELSEIF <condition> THEN<steps> ENDIF
   3. 后置条件：
      1. 描述：基本事件流执行后产生的变化
      2. 格式：一句话或多句话
4. 分支事件流：
   1. 格式：
      1. 事件流内部要求相同。
      2. 后置条件要求相同
      3. 所有分支事件流必须以“ABORT”或“RESUME”+”RFS”+数字结尾，占据一行
   2. 分类：
      1. 特定分支流：“RFS”+单个数字
      2. 有界分支流：“RFS”+数字+“-”+数字
      3. 全局分支流：
         1. “IF”+一句话
         2. +步骤
         3. “END IF”

## 需求目标

在这里我们将对于系统所要达成的目标进行描述。

### 功能性需求

* 系统需要提供GWT转为RUCM的功能
* 系统需要对用户提供的输入文档进行保存
* 系统需要向用户提供可以操作的窗口
* 系统需要可以向用户提供展示生成的RUCM的窗口
* 系统需要将生成的RUCM保存成文件，以供用户查阅

### 非功能性需求

* 系统需要便于安装，减少学习成本。
* 系统对硬件系统要求不应超过当前主流电脑要求，若放置云端应以最小的成本进行运行。
* 系统应减少在运行过程中崩溃的可能性
* 系统应采取一定空间换时间的方式提高运行效率。
* 系统应给予用户一定的自定义特性，或作为开源项目以方便自定义
* 系统应在高等软件工程结课之前开发完version 1
* 系统代码应包含必要的注释，以方便修改。
* 系统应依赖自包含，或者通过依赖管理工具以提高可维护性和分布式开发效率。

### 系统质量判断标准

由于本系统是课程系统，所以我们对于第一版的系统质量判断标准为：

* 能够将规定格式的GWT转化为标准的RUCM文档。若生成的RUCM与原先的RUCM表述意思相同，则为合格的转化过程。可通过转化率（合格RUCM/总RUCM）判断系统质量。

# 2. 领域术语说明

## 2.1 GWT

即Given，When，Then。描述内容如图1所示。

图1 GWT描述规范

## 2.2 RUCM



图2 RUCM文档描述规范

RUCM即受限用例建模（Restricted use case modeling）

## 2.3 机器学习相关术语解析

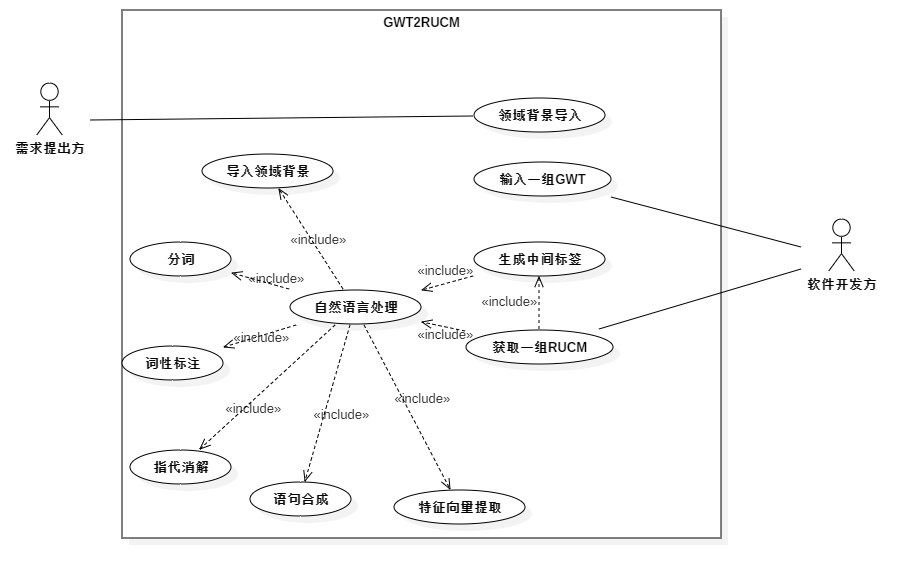
1. 机器学习：机器学习是用数据或以往的经验，以此优化计算机程序的性能标准。机器学习目前已广泛应用于数据挖掘、计算机视觉、自然语言处理、生物特征识别、搜索引擎、证券市场分析、战略游戏和机器人等领域。
2. 深度学习：深度学习（deep learning）是机器学习的分支，是一种试图使用包含复杂结构或由多重非线性变换构成的多个处理层对数据进行高层抽象的算法。深度学习是机器学习中一种基于对数据进行表征学习的算法。深度学习的好处是用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。
3. 监督学习：从给定的训练数据集中学习出一个函数，当新的数据到来时，可以根据这个函数预测结果。监督学习的训练集要包括输入和输出，也可以说是特征和目标。训练集中的目标是由人标注的。常见的监督学习算法包括回归分析和统计分类等。
4. 无监督学习：无监督学习与监督学习相比，训练集没有人为标注的结果。常见的无监督学习算法有生成对抗网络（GAN）、聚类等。
5. 半监督学习：介于监督学习与无监督学习之间的学习方式。
6. 人工神经网络（Artificial Neural Network，ANN）：简称神经网络（Neural Network，NN）或类神经网络，在机器学习和认知科学领域，是一种模仿生物神经网络（动物的中枢神经系统，特别是大脑）的结构和功能的数学模型或计算模型，用于对函数进行估计或近似。神经网络由大量的人工神经元联结进行计算。大多数情况下人工神经网络能在外界信息的基础上改变内部结构，是一种自适应系统，通俗的讲就是具备学习功能。现代神经网络是一种非线性统计性数据建模工具。

## 2.4 自然语言处理相关术语解析

1. 自然语言处理：（Natural Language Processing，NLP）是人工智能和语言学领域的分支学科。此领域探讨如何处理及运用自然语言；自然语言处理包括多方面和步骤，基本有认知、理解、生成等部分。自然语言认知和理解让电脑把输入的语言变成有意义的符号和关系，然后根据目的再处理。
2. 中文分词：中文分词(Chinese Word Segmentation) 指的是将一个汉字序列切分成一个个单独的词。分词就是将连续的字序列按照一定的规范重新组合成词序列的过程。在英文的行文中，单词之间是以空格作为自然分界符的，而中文只是字、句和段能通过明显的分界符来简单划界，唯独词没有一个形式上的分界符，虽然英文也同样存在短语的划分问题，不过在词这一层上，中文比之英文要复杂得多、困难得多。
3. 词性标注（Label）：机器理解自然语言文本时往往需要对词的词性进行标注。标注用于表征词的一种隐状态，隐藏状态构成的转移就构成了状态转移序列。例如：苏宁易购/n 投资/v 了/u 国际米兰/n。其中，n代表名词，v代表动词，n、v都是标注，以此类推。
4. 信息抽取或信息提取（Information Extraction）：通常处理对象为非结构化文本，在一定程度分析和理解文本的基础上，从大量文字数据中自动抽取名称、事件等特定消息（Particular Information）。
5. 自动摘要（Automatic Summarization）：从一个或多个文本中自动摘取包含了原文中最重要信息的部分，通常用于提供已知领域的文章摘要，例如产生报纸上某篇文章之摘要。

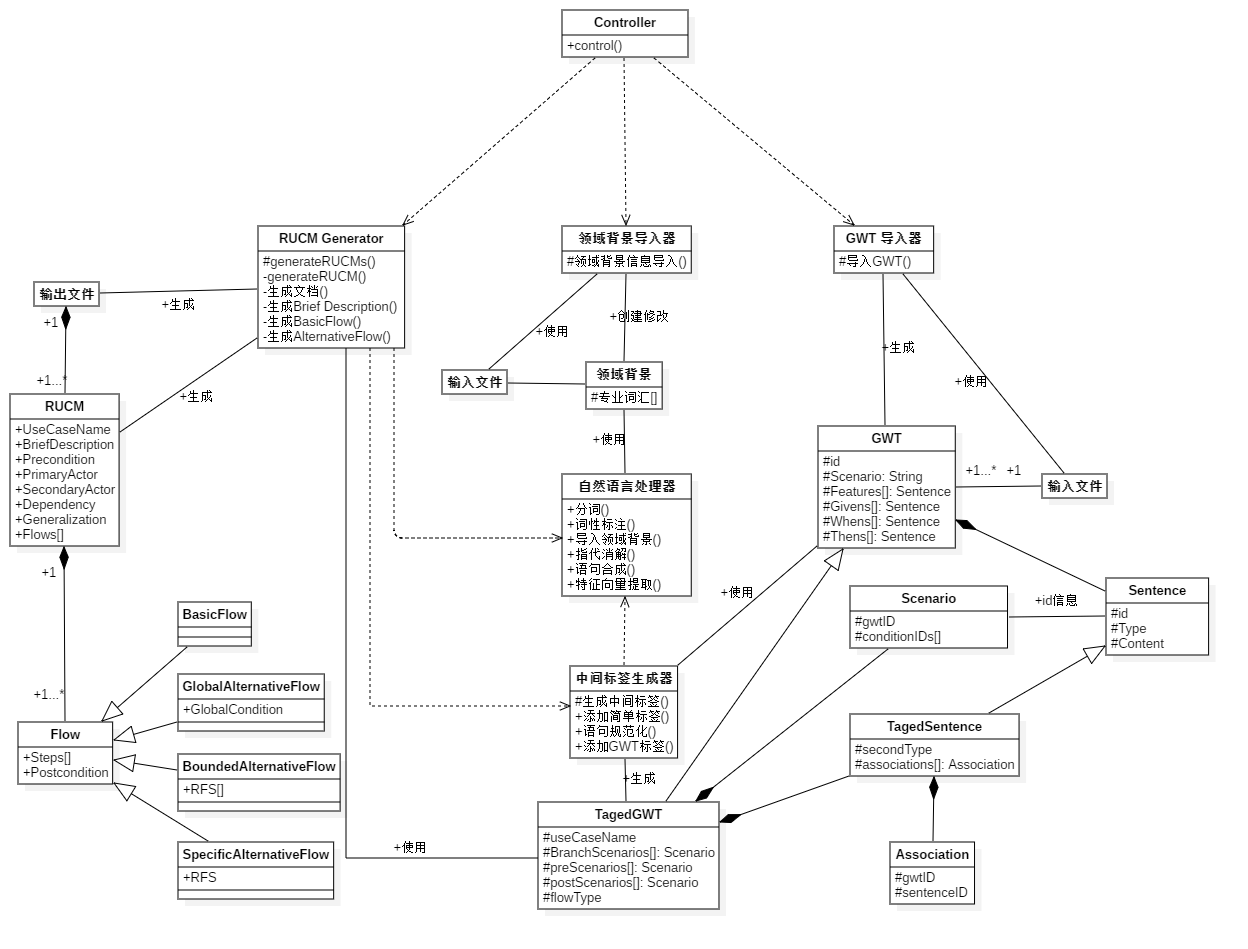
# 3. 系统模型描述

## 3.1 用例图



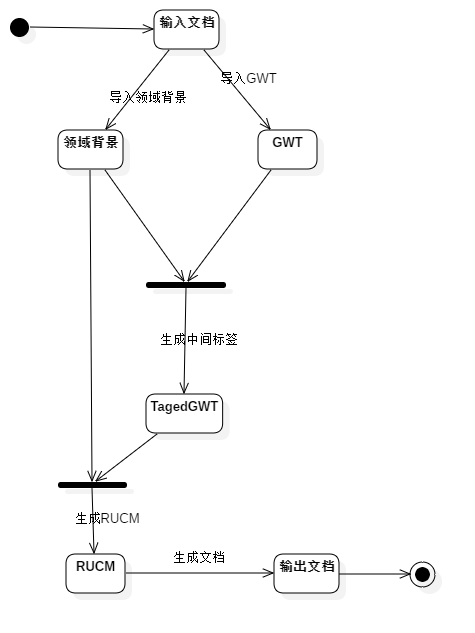
* Actors
  + 需求提出方：提出需求，因为是在其领域对于领域背景的了解，所以由其提供领域背景信息
  + 软件开发方：是RUCM的需求方，同时也是需求文档的撰写者。可能包含多种角色，例如产品经理、程序开发人员等。
* Use Cases
  + 领域背景导入：用户输入领域背景输入文档，系统以格式化方式存储领域背景信息。
  + 输入一组GWT：用户输入GWT输入文档，系统以格式化方式存储GWT。
  + 获取一组RUCM：根据存储的GWT和领域背景信息，用户选择一部分GWT文档转化为RUCM。
  + 生成中间标签：通过选择的GWT和存储的领域背景信息，系统将原始GWT添加标签。
  + 自然语言处理：包含一系列的自然语言处理操作，在此进行综合的处理。
    - 导入领域背景
    - 分词
    - 词性标注
    - 指代消解
    - 语句合成
    - 特征向量提取

## 3.2 类图



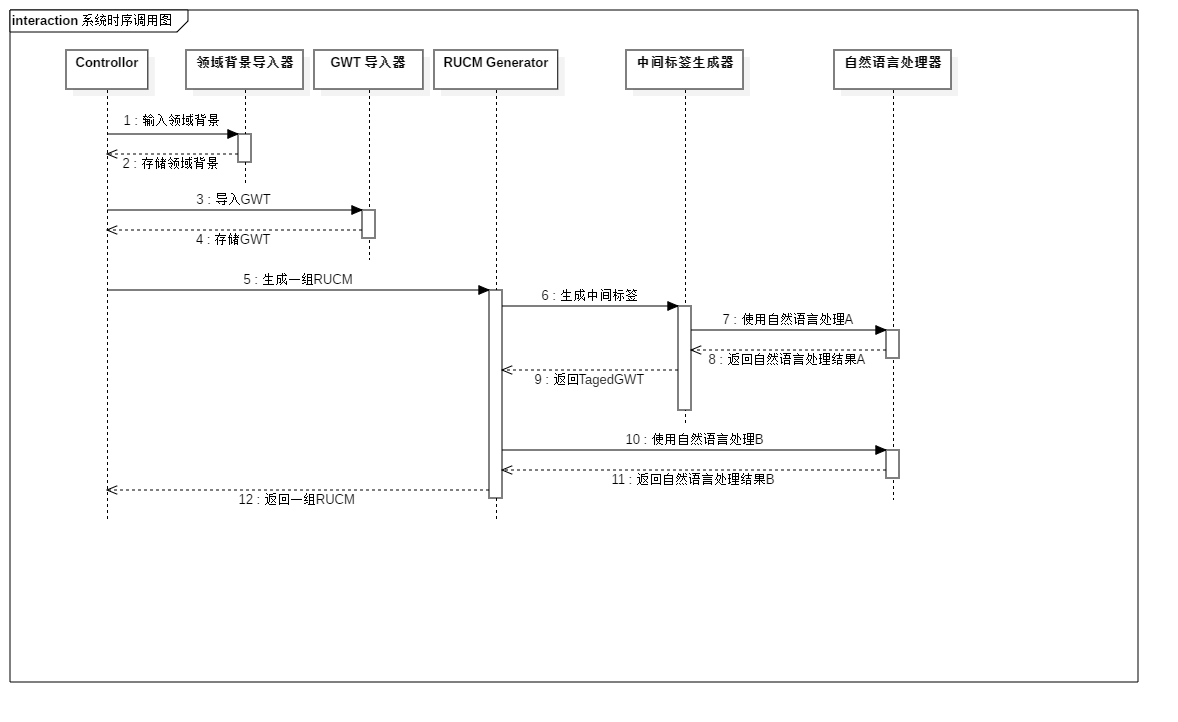
* Controller：负责相应外部事件，产生相应动作，负责调用RUCM Generator::generateRUCMs()，领域背景导入器::领域背景信息导入()，GWT导入器::导入GWT。
* RUCM Generator：提供将GWT转化为RUCM的接口。
* 领域背景导入器：提供导入领域背景的接口，将领域背景信息结构化存储。
* GWT导入器：提供导入GWT的接口，将输入文档中的GWT文档，按照GWT类进行存储。
* 自然语言处理器：提供自然语言处理的一系列操作的接口，现在均为public为了使调用方便。但是在未来确认了自然语言处理器和RUCM Generator所使用的方法之后，可以进一步进行封装。
* 中间标签生成器：将GWT转化为TagedGWT，便于接下来的完全转化过程。
* Sentence：
  + Id：独一的表示出GWT中的语句。
  + Type：Feature、Given、When、Then的二级级标签。
  + Content：例：Given中的一行。或Feature中的一段话。
* TagedGWT：添加标签后的GWT。在GWT的基础，添加GWT之间的关系，确定自身所在的Flow类型。每个GWT由Scenario进行表示，存储id和与当前GWT相关的condition的id。
* TagedSentence：TagedGWT中的Sentence会变为TagedScence。该类除了GWT本身的内容意外，还存储二级标签下的三级标签以及句子之间的关系。
* Association：句子之间的关系，首先需要由另一个句子的id，另外存储关系的类型，比如两句意思相同，或两句意思相反等。

## 3.3 文件状态图



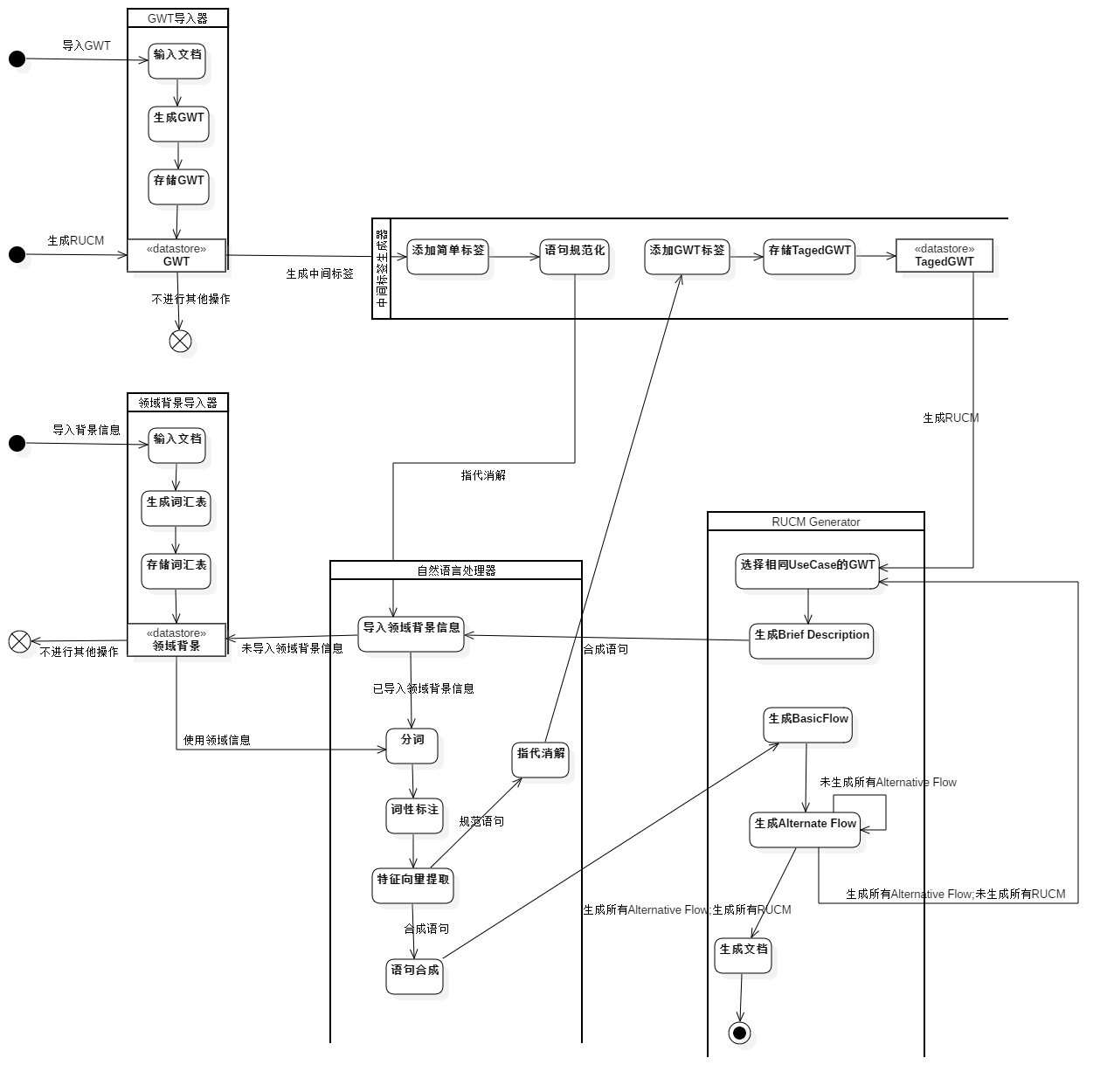
本系统的主要目的是文档的转化，而该状态图展示了在这个系统中文档的状态变化，声明了每种文档状态生成之前所需要的文档，表明了文档之间的相互依赖。

## 3.4 系统时序图



该图展示了整个系统的时序过程，展示了系统通过controller对整个系统的调度。同时中间标签生成器与自然语言处理器是通过递归方式进行调用，从而与controller隔离开来，减少了类与类之间的耦合关系，使得整个系统更加结构鲜明。

## 3.5 系统活动图



该图为整个系统的活动图。其依据用例图，具有三个起点，分别为导入背景信息，导入GWT，生成RUCM。其中导入背景信息与导入GWT都较为简单，我们接下来会对生成RUCM进行描述。Ps：此仅为需求设计版本，尚有一些技术问题和细节问题未解决，例如数据操作、自然语言处理的具体流程等。

1. 从存储中，获取GWT
2. 生成中间标签
   1. 添加中间标签
   2. 语句规范化
   3. 自然语言处理
      1. 导入领域背景信息
      2. 根据情况，获取领域背景
      3. 分词
      4. 词性标注
      5. 特征向量提取
      6. 指代消解
   4. 添加GWT标签
   5. 存储TagedGWT
3. 生成RUCM
   1. 从内存和存储中，获取TagedGWT
   2. 选择相同UseCase的GWT
   3. 生成Brief Description
   4. 自然语言处理
      1. 导入领域背景信息
      2. 分词
      3. 词性标注
      4. 特征向量提取
      5. 指代消解
      6. 语句合成
   5. 生成Basic Flow
   6. 生成Alternative Flow
      1. 未生成当前所有Alternative Flow
         1. 返回3 f
      2. 生成当前所有Alternative Flow，但未生成所有RUCM
         1. 返回3 b
   7. 生成所有Alternative Flow且生成所有RUCM的情况下，生成文档
4. 结束

# 4. 系统架构设计

## 4.1 中间标签设计

对于一句话(允许关键词)，标签设计如下：

* 内容
  + Given
    - Precondition
      * PreconditionType
    - fixed data
  + When
    - action
      * position
      * actionType
    - input data
  + Then
    - Postcondition
      * postconditionType
    - output data

对于一个GWT，标签设计如下：

* Use Case:
* Name
* BranchScenarios
  + BranchScenario
    - Name
    - BranchConditions
      * Branchcondition
* preScenarios:
  + preScenario:
    - Name
    - conditions
      * condition
* postScenarios
  + postScenario:
    - Name
    - conditions
      * condition
* FlowType: Basic Flow/Specific Alternative Flow/Global Alternative Flow/Bounded Alternative Flow

### GWT->中间标签规则：

1. 输入文档存储时，将输入的GWT按照feature,scenario,given,when,then四个部分进行存储。
2. 将GWT中原先的每一行作为一个句子来进行拆分
3. 给该句话指定初始标签，包括对于语句的类型进行标注，对于INCLUDE、 EXTENDED、GLOBAL、 Use Case Name这类具有关键词的语句进行标注
4. 如果该初始标签处于需要自然语言处理的标签列表中（例如该句话是Given中的一句肯定句或否定句（在这里，因为我们设计GWT中无分支，所以action在这里先不用进行处理），则进行以下过程，否则直接进行接下来的过程：
   1. 对语义相近句子进行统一化，对该语句进行独一相近标注，并修改其Type值
   2. 对语义相反语句进行规范化，对这两个语句进行独一相反标注，并修改其Type值
5. 根据GWT中的单个语句与其从属Use Case，按照一下规则对GWT进行标签化：
   1. 给定GWT文档G1，G2
   2. 提取两个文档中，标签中存在precondition和postcondition的语句
   3. 判断条件
      1. 若G1与G2中，标签为precondition的语句，部分语义相同，部分语义相反，则G1与G2彼此添加为BrachScenario标签，并且添加用于进行分支的conditions
      2. 若G1中标签为precondition的语句与G2中标签为postcondition的语句完全完全相同，则G1的preScenario为G2，G2的postScenario为G1

### 中间标签->RUCM规则：

1. 下面对Use Case标签相同的GWT进行处理（以下描述均为正常状况）
2. 提取Use Case标签作为RUCM的Use case Name
3. 将GWT的scenario描述进行自然语言处理，进行结合，形成Brief Description
4. 提取无preScenario的GWT的所有标签为precondition的语句作为precondition
5. 提取无preScenario的GWT的所有标签为action的语句中的人物提取，作为Actor，存储以便于标准化
6. 提取GWT中的关键词，放至dependency和Gerneralization
7. 将无preScenario的GWT的所有标签为action的语句，按照action中的position标签放到Basic Flow的最前面。
8. 以无preScenario的GWT的postScenario作为线索，形成整个BasicFlow，与此同时进行Actor名称的统一化表述。
9. Basic Flow的postcondition为最后一个添加的GWT的所有标签为postcondition的语句

10、以Basic Flow中的GWT作为线索，寻找其BranchScenario标签，从而填写其他三种类型的Flow，并且修改Basic Flow原始语句，添加Validate that 等关键词。

## 4.2 系统架构图

展示层

控制层

接收事件

响应层

GWT导入器

领域背景导入器

RUCM Generator

响应事件

获取支持

调用数据

数据层

领域背景

TagedGWT

GWT

功能支持层

自然语言处理器

中间标签生成器

数据调度器

* 展示层：用于显示界面用于用户操作和查看
* 控制层：接收展示层所产生的事件，并调用相应操作
* 响应层：真正用于响应事件的实现
* 功能支持层：提供支持用于响应层中组件实现响应（类图中，现在不存在数据调度器，因为当前类图已经十分庞大，接下来在设计阶段会进行分解）
* 数据支持层：存储数据