## 1、OCL约束

### (1)展示层

* 显示初始页面

context controller::home():file

pre:

post:return = index.html

### (2)控制层

* 调用领域背景导入，导入领域背景，显示导入结果

context controller::import\_background():(file, message)

pre: input\_file <> Null

index.html->领域背景导入->button->submit = true

post:if 导入成功

then return = ('index.html', message='领域背景导入成功')

else return = ('index.html', message='领域背景导入失败')

* 调用GWT导入，导入GWT，显示导入结果

context controller::import\_gwt():(file, message)

pre: input\_file <> Null

index.html->gwt导入->button->submit = true

post:if 导入成功

then return = ('index.html', message='GWT导入成功')

else return = ('index.html', message='GWT导入失败')

* 调用RUCMGenerator，首先选择需要导入的GWT，然后进行GWT->RUCM的转化，得到转化成功的RUCM，并且显示生成结果

context controller::translation():(file, message)

pre: index.html->开始转换->button->submit = true

post:if 选择GWT列表

then return = ('index.html', message=转化结果)

else return = ('index.html', message='已经取消转化')

RUCM.doc--或者这个不是在这里出现，而是在RUCMgenerator中出现

inv: notEmpty(转化结果)

### (3)响应层

* BackgroundImporter为领域背景信息导入器，将用户上传的领域背景信息传递给自然语言处理器，提高自然语言处理器的精度.

context: BackgroundImpoter

inv:self.NLPExecutor->size()=1

self.Background->size()>0

context: BackgroundImpoter::importBackground(segDict:file,posDict:file)

pre:segDict<>None or posDict<>None

self.NLPExecutor->size()=1

post: self.NLPExecutor<> self.NLPExecutor@pre

* GWTImporter为GWT描述导入器，将用户上传的GWT文件使用自然语言处理器NLPExecutor进行处理，提取主要内容以生成对应的GWT对象。GWTImporter每次读取一个用户输入的GWT文件。

context:GWTImporter

inv:self.inputFile->size()=1

self.NLPExecutor->size()=1

* 传入GWT文件路径或内存中保存的GWT文档内容，通过NLPExecutor提取出文档各字段内容转换成GWT对象。操作完成后GWTImporter关联的GWT数据对象数量增加。

context:GWTImporter::importGWT(filepath:string,filecontent:string )

pre:filepath.size>0 or filecontent.size()>0

self.NLPExecutor->size()=1

post: self.DataDispatcher.GWT -> size() >= (self.DataDispatcher.GWT -> size())@pre

* RUCMGenerator为RUCM描述生成器，对用户选择的GWT进行分析并添加中间标签，提取中间标签标记的内容生成RUCM对象，将一组RUCM对象输出为一个文本文档OutputFile。RUCMGenerator接受一组原始GWT对象，使用标签生成器LabelGenerator分析GWT对象属性并添加中间标签转换成TaggedGWT，使用自然语言处理器NLPExecutor分析一组TaggedGWT对象生成一个完整的RUCM描述。

Context RUCMGenerator

inv:

self.NLPExecutor->size()=1

self.LabelGenerator->size()=1

self.OutputFile->size()=1

self.RUCM->size()>=1

* 对输入的一组GWT数据进行分析，将其转换成一组对应的RUCM描述，生成输出文档内容。

Context RUCMGenerator::generateRUCMs(gwtList:set)

Pre: self.NLPExecutor->size()=1

self.LabelGenerator->size()=1

post: self.OutputFile->size()=1

self.RUCM->size()>=1

* 将输入的一组TaggedGWT对象转换成一个完整的RUCM描述。

Context RUCMGenerator::generateRUCM(taggedList:set)

Pre: self.NLPExecutor->size()=1

self.LabelGenerator->size()=1

taggedList->size()>0

post: RUCM -> size() = (RUCM -> size())@pre+1

* 由输入的一组TaggedGWT对象生成对应RUCM描述的Brief Description。

Context RUCMGenerator::briefDescription(taggedList:set，rucm:RUCM)

Pre: self.NLPExecutor->size()=1

taggedList->size>0

post:rucm.briefDescription<>None

* 由输入的一组TaggedGWT对象生成对应RUCM描述的Basic Flow。

Context RUCMGenerator::basicflow(taggedList:set，rucm:RUCM)

Pre: self.NLPExecutor->size()=1

taggedList->size>0

post: rucm.basicFlow<>None

* 由输入的一组TaggedGWT对象生成对应RUCM描述的Specific Alternative Flow、Bounded Alternative Flow、Global Alternative Flow(若RUCM含有这一种或几种可选事件流)。

Context RUCMGenerator::alternativeFlow(taggedList：set，rucm:RUCM)

Pre: self.NLPExecutor->size()=1

taggedList->size>0

post: rucm.specificAlt<>None or rucm.boundedAlt<>None or rucm.globalAlt<>None

* 将生成的一组RUCM对象转换成输出文件文本内容。

Context RUCMGenerator::generateOutput()

Pre: self.RUCM->size>0

post: self.OutputFile->size()=1

### (5)数据层

* LableGenerator为中间标签生成器，对输入的一组GWT进行分析并添加中间标签，转换成TaggedGWT对象。

Context LableGenerator

inv:self.NLPExecutor->size()=1

self.GWT->size()>0

* 对传入的一组GWT对象进行分析，添加标签生成对应的一组TaggedGWT对象

Context generateLable(gwtlist:set):set

pre:gwtList->notEmpty()

self.NLPExecutor->size()=1

post:self.TaggedGWT->size()>（self.TaggedGWT->size())@pre

* 对传入的一组GWT对象进行分析，添加GWT级的标签UseCaseName、FlowType，句子级的标签wordlist、actor、action

Context \_simpleLable(gwtList:set):set

pre: gwtList->notEmpty()

self.NLPExecutor->size()=1

post: self.TaggedGWT->forAll(useCaseName<>None or flowType<>None)

self.TaggedGWT.Whens->forAll(wordlist<>None and actor<>None and action<>None)

* 对传入的一组GWT对象进行分析，添加GWT级的标签refer、condition、FlowType，句子级的标签normalContent、type、associated

Context \_addGWTLable()

pre: gwtList->notEmpty()

self.NLPExecutor->size()=1

post: self.TaggedGWT->forAll(flowType<>None )

self.TaggedGWT.Whens->forAll(normalContemt<>None type<>None)

self.TaggedGWT.Thens->forAll(normalContemt<>None type<>None)

self.TaggedGWT.Givens->forAll(type<>None)

* NLPExecutor为自然语言器，向上为RUCMGenerator和LabelGenerator提供自然语言处理功能支持

Context NLPExecutor

inv:self.Background->size()=1

self.nlpTool->size()=1

* 对输入的中文句子进行分词，返回句子词汇的集合。

Context NLPExecutor::wordTokenize(sentence:string):set

pre: self.nlpTool->size()=1

sentence.size()>1

post:return=wordlist

* 对输入的中文句子分词结果进行词性标注，返回句子词汇词性标注的集合。

Context NLPExecutor::posTag(wordlist:set)

pre: self.nlpTool->size()=1

sentence.size()>1

post: return=wordtaglist

* 使用用户输入的自定义词典重新训练自然语言处理模型，可输入的自定义词典为分词词典和标注词典。

Context NLPExecutor::dictUpdate(segDict:file,posDict:file)

pre: self.Background->size()>0

self.nlpTool->size()=1

segDict<>None or posDict<>None

post: self.nlpTool<> [self.nlpTool@pre](mailto:self.nlpTool@pre)

* 将输入的两个句子向量化，计算并返回句子的相似度。

Context NLPExecutor::similarity(sent1:string,sent2:string):real

pre: self.nlpTool->size()=1

sent1.size()>0

sent2.size()>0

post:return=similarity

* 将输入的中文文本切分成句子，返回句子的集合。

Context NLPExecutor::splitSentence(text:string)：set

pre: self.nlpTool->size()=1

text.size()>0

post:return=sentlist

* 从输入的文本中提取重要信息，生成文本摘要。

Context NLPExecutor::generateSummary(text:string):string

pre: self.nlpTool->size()=1

text.size()>0

post:return=summary

* 根据句子的分词和标注结果做依存句法分析，返回句子的依存关系集合。

Context NLPExecutor::parse(wordlist:set , wordtaglist:set):set

pre: self.nlpTool->size()=1

wordlist.size()>0

wordtaglist.size()>0

post:return=parseset