UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA

Mestrado em Engenharia Informática

Dissertação

Geração de diagramas UML a partir da análise de textos que definam os requisitos de um sistema de informação

Ricardo Jorge Marques Borlas Frango

Orientador:
Paulo Quaresma
Coorientador:
Luís Arriaga

09/05/2012

Mestrado Engenharia Informática

Dissertação

Geração de diagramas UML a partir da análise de textos que definam os requisitos de um sistema de informação

Ricardo Jorge Marques Borlas Frango

Orientador:
Paulo Quaresma
Coorientador:
Luís Arriaga

Agradecimentos

Gostaria de agradecer aqueles que, direta ou indiretamente, contribuíram para a realização desta dissertação.

Gostaria então de agradecer:

- Ao professor Paulo Quaresma pela preciosa ajuda que me deu para ultrapassar os constantes obstáculos com os quais me deparei na realização desta dissertação;
- Ao professor Luís Arriaga pela, também não menos preciosa, ajuda nas questões mais teóricas;
- Aos meus pais, irmão e amigos que viveram comigo as dificuldades na realização desta dissertação e que tanto me incentivaram e apoiaram para seguir em frente, mesmos nos momentos de maior dificuldade.

Para todos eles, fica o meu bem-haja e o mais sincero pedido de desculpas pelos períodos de mau humor!

Resumo

A escolha do tema "Geração de diagramas UML a partir da análise de textos que definam os requisitos de um sistema de informação" deveu-se em parte à falta de aplicações com a capacidade de análise de textos e consequente construção de sistemas de informação.

Esta tese tem como objetivo facultar um apoio na construção de diagramas UML analisando um determinado texto para tal.

Para a construção destes diagramas foi então necessário, inicialmente, uma pequena análise sintática das frases de um texto que descreve uma determinada situação.

Através de um analisador sintático feito a esse mesmo texto, utilizando o site http://beta.visl.sdu.dk/ obtemos a analise sintática da frase que posteriormente será analisada utilizando a linguagem de programação Python.

Ao analisar então este resultado da análise sintática, aplicando determinadas regras, é criado o diagrama de classes UML.

Pretende-se com esta dissertação, que seja possível facultar um apoio à construção de diagramas UML.

Foi aplicada esta metodologia a um conjunto de textos de sistemas de informações tendo-se obtido resultados satisfatórios.

Abstract

This theme “Generation of UML diagrams through the analyses of texts that define the requisites of information system” was chosen because of the absence of applications white the capacity to analyze texts and construct the consequent information system.

This dissertation aims to provide support in the construction of UML diagrams, analyzing a particular text for this.

For the construction of these diagrams was then necessary, initially, a little parsing of the sentences of a text that describes a particular situation.

Through a parser made to that same text using de website http://beta.visl.sdu.dk/ we will get the parsing that will be later analyzed, using the Python programming language.

After analyzing the parser, applying some rules, it will be generated the UML class diagrams.

It’s pretended that with this dissertation, the user has a support to the construction of UML diagrams.

This methodology was applied to several examples of text information systems, yielding satisfactory results.

Índice de Ilustrações

Índice

Capitulo I - Introdução

Esta dissertação tem, como o próprio nome indica, o objetivo de gerar diagramas UML através da análise de textos.

Este tema foi escolhido por tudo o que envolve a programação orientada a objetos e o gosto por bases de dados que se assemelha ao UML. Juntando ao facto de haver este gosto pessoal por este tipo de programação, a possibilidade de fazer esta dissertação em Python, uma linguagem também orientada a objetos, e que era, na altura, desconhecida para mim, tornou ainda mais apetecível a vontade de optar por este tema.

Para começar a apresentar desta dissertação é necessário perceber o que é o UML e a linguagem Python.

O UML é uma linguagem que permite ao utilizador especificar, visualizar e documentar modelos de software orientados a objetos. O UML não tem como objetivo desenvolver aplicações, mas sim ajudar o utilizador na visualização do seu desenho [FundUML].

O UML disponibiliza uma forma padrão para moldar projetos de sistemas incluindo os seus aspetos conceptuais além dos itens concretos. O objetivo do UML é disponibilizar a possibilidade de descrever qualquer tipo de sistema. Estes sistemas são representados em diagramas orientados a objetos. O UML é, no entanto, para outros paradigmas de programação, de certa forma limitado.

Sendo o UML uma linguagem que permite ao utilizador especificar e visualizar modelos de software criados pelo utilizador, é uma linguagem que permite a criação de diferentes tipos de diagramas. O UML pode ser dividido em três grandes grupos. Esses grupos são [DiagUML]:

- Diagramas de Comportamento;
- Diagramas de Interacção;
- Diagramas de Estrutura.

Os Diagramas de Comportamento são diagramas que representam um sistema no que diz respeito às características do seu comportamento. Dentro deste grupo existe o diagrama de atividade, máquina de estado, "Use Cases" e interação.

Os diagramas de atividade são diagramas que tem como objetivo descrever regras de negócio de alto nível, descrevendo a sequência das suas atividades nesses sistemas e são de certa forma uma variação dos diagramas de estado [FundUML]. A máquina de estados ou diagrama de estados representa, ou descreve, os vários estados de um objeto, bem como as suas mudanças de estado [DiagUML]. No caso do diagrama de "Use Cases", este é utilizado para representar os relacionamentos e dependências entre os atores e os casos de uso. Por seu lado, os diagramas de interação são uma variação dos diagramas de atividade, mas nestes diagramas é onde é mostrado o controlo de fluxo dentro de um sistema e cada nó pode ser outro diagrama de interação [DiagUML].

No grupo de Diagramas de Interação existem três tipos de diagramas, que são eles o diagrama de comunicação, o diagrama de sequência e o diagrama de sincronismo.

Os diagramas de comunicação são diagramas focados nas classes. Nestes diagramas são mostradas as classes, os seus relacionamentos e o fluxo de mensagens trocadas entre essas mesmas classes [DiagUML].

Os diagramas de sequência são utilizados para moldar num negócio a sua lógica sequencial através da ordem das mensagens no tempo. Este diagrama para além de moldar essas mensagens mostra também essa mesma troca de mensagens entre os objetos. Nos diagramas de sincronismo é onde são descritas as mudanças de estado de uma instância através do tempo [DiagUML].

No grupo dos Diagramas de Estrutura estão incluídos diagramas em que os estados dos objetos não são alterados com o tempo. Dentro deste grupo estão incluídos os diagramas de classe, diagramas de estrutura, diagramas de componentes, distribuição e objetos.

Os diagramas de classe são diagramas onde estão representadas as classes e todos os seus atributos e as suas relações. Para representar a estrutura interna de uma classe bem como os pontos de interação das classes com outras partes do sistema são utilizados os diagramas de estrutura [DiagUML].

Os diagramas de componentes, tal como o próprio nome indica, são utilizados para representar os componentes de um sistema, bem como as interações e interfaces públicas.

Nos diagramas de distribuição são mostradas as arquiteturas de execução de um sistema.

Para realizar esta dissertação optou-se por gerar diagramas de classe UML através da análise de sistemas de informação por ser o tipo de diagramas UML mais utilizados.

O "Python" foi utilizado nesta dissertação como a linguagem utilizada para desenvolver esta aplicação capaz de gerar os diagramas de classe UML. O "Python" é uma linguagem de programação lançada por Guido Van Rossum em 1991 [WikiPy]. Esta é uma linguagem de programação de alto nível que é interpretada (ou seja, existe um interpretador que analisa o código fonte e é posteriormente executado pelo processador), imperativa (tem ações, enunciados ou comandos que variam) e orientada a objetos.

Esta linguagem, ao contrário de outras, não obriga à pré-declaração de variáveis o que permite que os seus tipos sejam determinados dinamicamente e não obriga ao aparecimento de determinados caracteres ou expressões para definir um bloco de código que compõe uma função ou condição.

1.1. Objetivos do estudo

No caso desta dissertação, propôs-se então a criação de diagramas de classes UML. Houve, então, a necessidade de se obter uma ferramenta capaz de fazer a análise detalhada do texto. Através dessa análise do texto é obtido um resultado, sendo esse resultado construído através da análise sintática do texto realizada de forma simples, ou seja analisado o texto palavra a palavra, e o resultado obtido fica então ilustrado na ilustração 1:

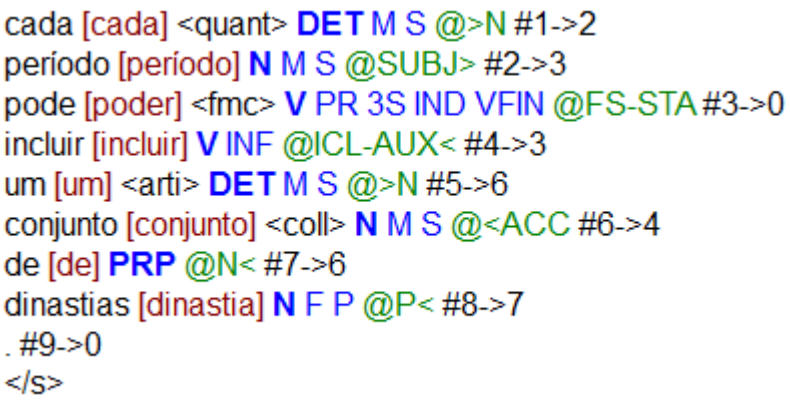

Ilustração 1 - Exemplo de uma análise sintática feito à frase: "Cada período pode incluir um conjunto de dinastias."

Sendo os diagramas de classe constituídos por classes, e sendo essas classes constituídas por atributos, existe sempre a necessidade de analisar cada frase do texto detalhadamente, ou seja palavra a palavra. Foi por esse motivo que se optou por este tipo de análise sintática. Como ilustrado na Ilustração 1, é fácil identificar os nomes, verbos e determinantes por ser uma análise feita palavra a palavra. Neste exemplo simples, podemos identificar claramente que desta frase resultaria uma classe "período" com um atributo "conjunto_de_dinastias".

O objetivo desta dissertação passa então por analisar, não uma frase, mas um texto completo, de forma a criar e ilustrar um diagrama de classes UML de fácil compreensão para o utilizador.

1.2. Relevância do estudo

A capacidade de desenvolvimento de diagramas UML sempre foi um tema algo abstrato, em que, dependendo das várias interpretações e perspetivas pessoais da análise do sentido literário do texto, podem resultar diferentes representações dos diagramas de classe.

Esta dissertação tem como objetivo abstrair-se um pouco dessa interpretação literal e focar-se mais na interpretação sintática da frase.

Esta análise sintática é feita através da identificação de verbos e de nomes que possam ocorrer nas frases ao longo do texto e dessa forma permite ligar os nomes das

classes aos seus atributos. O ser humano tem sempre uma perspetiva mais baseada na compreensão e sentido literário das frases, abstendo-se da construção gramatical. Utilizando uma análise baseada na construção gramatical dos textos, reduz-se a possibilidade de haver várias perspetivas distintas de diagramas de classe UML, e dessa forma pode ser apresentado um diagrama de classes baseado na análise gramatical do texto, mostrando, talvez, uma perspetiva que escapa por vezes ao olho humano.

Existem diversos trabalhos na internet que mostram diferentes perspetivas de como podemos construir classes e os atributos que as constituem, no entanto todas elas se baseiam numa perspetiva pessoal, tendo todas elas em comum e sempre presente a forma de representar um diagrama de classes.

O objetivo deste trabalho passa, não só, por fornecer um diagrama de classes construído com base no sentido sintático das frases, como também abrir perspetivas para a construção de outro tipo de diagramas UML, ou quem sabe para outro tipo de iniciativas baseadas em análises de texto.

Para atingir o objectivo final foi utilizado um analisador sintático através do VISL. Este analisador irá gererar um resultado obtido através da análise do texto de sistemas de informação. Este analise é feita a partir do link http://beta.visl.sdu.dk/visl/pt/parsing/automatic/dependency.php.

Após a obtenção deste resultado da análise sintatica do texto de sistemas de informação são então criadas as classes e os respectivos objectos de cada, bem como as associações entre estas classes.

Com base nestas classes criadas, é gerado um ficheiro XML. Este ficheiro XML é ficheiro criado com base na estrutura XML do software ArgoUML que é um software utilizado para a criação de diagramas UML.

Após a criação do ficheiro XML, este é importado para o ArgoUML e desta forma obtem o diagrama de classes UML, resultante do texto de sistemas de informação.

1.3. Estrutura da dissertação

Esta dissertação foi dividida em cinco capítulos.

No primeiro capítulo foi escrita a introdução onde é contextualizado este trabalho, ou seja, no que consiste e quais os objetivos que se a alcançar bem como a sua relevância. É feita também uma ligeira apresentação às ferramentas utilizadas. É ainda apresentado o ponto atual da leitura que é a descrição da sua estrutura.

No segundo capítulo é onde se apresenta o estado da arte, ou seja, é neste capítulo que se vão apresentar as ferramentas que foram utilizadas nesta dissertação e que permitiram que fosse possível a realização desta mesma dissertação.

O capítulo terceiro é o capítulo que apresenta a proposta. É neste capítulo que é contextualizada a forma de representação dos textos e de que forma se pretende analisar essa mesma forma. É neste ponto que é explicado detalhadamente como foi desenvolvida esta dissertação e como foi possível atingir os objetivos pretendidos e explicados na introdução.

Os casos de estudo estão descritos no capítulo quarto. É neste capitulo que são analisados todos os textos estudados e como estes ajudaram no desenvolvimento desta dissertação, enumerando as especificidades de cada um e como consequência as dificuldades que cada um apresentou. Para alem desta apresentação, no final de cada caso é ainda feita uma analise global ao resultado e se este é satisfatório, bem como a apresentação deste mesmo resultado final.

No último capitulo, o sexto, são apresentadas as conclusões finais desta dissertação e é também neste mesmo capítulo que é feita uma alusão aos trabalhos futuros, indicando o que poderia ser melhorado.

Capitulo II – Recursos para criação de Diagramas UML a partir da análise de textos

Para o desenvolvimento desta dissertação foi necessário analisar as ferramentas existentes para o seu desenvolvimento. Neste capítulo é apresentado o estado da arte desta dissertação bem como as ferramentas utilizadas

Tendo esta dissertação como objetivo o desenvolvimento de diagramas de classe UML foi necessário saber o que são estes diagramas UML e enquadrá-los neste trabalho.

Para a análise de textos de sistemas de informação foi necessário utilizar um analisador de textos para criar um esquema para posterior análise pela ferramenta a desenvolver nesta dissertação. Para isso foi utilizado o VISL que será também explicado neste capítulo.

O desenvolvimento desta dissertação foi feito utilizando a linguagem de programação Python, que permitiu a criação dos diagramas UML, bem como o desenvolvimento do código XML a ser importado para apresentação dos diagramas.

2.1. Diagramas UML (Unified Modeling Language)

O objetivo desta dissertação é o desenvolvimento de diagramas de classe UML através da análise de textos de sistemas de informação. Para iniciar esta dissertação foi necessário conhecer o UML, nomeadamente no que diz respeito ao seu conceito e às suas metodologias, essencialmente nos diagramas de classe UML.

Os diagramas de classe UML são diagramas onde são representadas as classes de um determinado sistema. Essas classes são constituídas por atributos e métodos e entre as classes existem associações de diversos tipos que mostram, tal como o próprio nome indica, as associações ou relações existentes entre as diferentes classes.

Nos diagramas de classe UML, as classes são utilizadas para representar determinados tipos de objectos e ao representarmos uma determinada classe temos que definidir as suas propriedades e acções.

Os atributos de uma classe representam as suas propriedades que são informações específicas de uma determinada classe e as suas acções são representadas por métodos dessa mesma classe.

Dentro dos diagramas UML existem diferentes tipos de associações, sendo que a mais usual é a associação simples que pretende apenas demonstrar, ou representar, um determinado relacionamento entre duas classes.

Existe depois a classe associativa que é criada quando se pretende manter uma determinada informação sobre essa associação. Dessa forma é criada uma classe associativa onde se coloca um atributo. É representada por uma linha a tracejado.

Para além destas associações existe ainda a associação de agregação, que é representada por um losango branco numa das extremidades ligada à classe mãe. Tem como característica o facto de representar uma associação em que o objeto é parte de um todo, no entanto esse objeto pode existir sem a existência do todo, e neste caso o todo seria a classe mãe.

A última associação é a associação de composição e esta associação é também representada por um losango, mas pintado de preto. Esta associação é semelhante à associação de agregação, no entanto a diferença é que a existência das partes depende do todo, sendo que o todo é a classe mãe e a classe mãe é onde está o losango preto.

Nesta dissertação foi utilizada apenas o tipo de associação simples para representar uma determinada relação existente entre duas classes.

2.2. Visual Interactive Syntax Learning

Como indicado no título VISL significa "Visual Interactive Syntax Learning" e este é um projeto de investigação e desenvolvimento do Instituto de Linguagem e Comunicação (ISK) da Universidade do Sul da Dinamarca (SDU) – Odense Campus. Este projeto está em desenvolvimento constante desde Setembro de 1996 por alunos e docentes desta universidade (ISK) e tem como objetivo o desenvolvimento de ferramentas gramáticas baseadas na Internet para educação e pesquisa [VISL].

Este projeto começou por ser desenvolvido inicialmente para quatro línguas diferentes, que eram elas Inglês, Francês, Alemão e Português, no entanto mais tarde acabou por ser desenvolvido também para dinamarquês, Espanhol e Esperanto. Esperanto é uma linguagem que tem como objetivo facilitar a comunicação a nível mundial [WikiEsp]. Com o tempo houve outros idiomas a aderirem ao projeto. Para todos estes idiomas que participam no projeto foram desenvolvidos "treebanks", ou seja, uma espécie de bancos de dados para criar florestas sintáticas o que permite que as ferramentas de ensino VISL funcionem com todas as línguas [VISL].

O VISL é um projeto em constante atualização e é um projeto que disponibiliza através da Internet as suas ferramentas gramáticas e linguísticas assim que o seu protótipo operacional esteja disponível. O resultado desta estratégia obriga a que os módulos estejam em constantes melhoramentos e atualizações.

O VISL tem uma arquitetura bastante dinâmica e foi desenhado para vários idiomas, no entanto o Inglês foi o escolhido como "linguagem mãe" para facilitar na manutenção de linguagens paralelas. As exceções são feitas por exemplo em casos de termos técnicos e termos abreviados.

O VISL foi desenvolvido utilizando uma vasta teia de páginas HTML, Java, bases de dados de texto e ferramentas para análise automática da gramática e consequente construção do output. O site oferece um ambiente gráfico do resultado da análise sintática o que facilita a análise do utilizador. Para além disto permite também uma análise do respetivo resultado automaticamente ou então de forma manual. Em

qualquer um destes dois tipos de pesquisa no centro da análise da palavra está a forma em que se encontra, a função que toma, o grupo a que pertence e o seu nível [VISL].

O Constraing Grammar é um paradigma metodológico para análise de linguagem natural. Na escrita linguística as regras dependentes do contexto são compilados para uma gramatica que atribui etiquetas gramaticas a palavras ou símbolos no texto. Um típico CG consiste em milhares de regras que são aplicadas em etapas progressivas para abranger as mais avançadas análises [VISL].

A descrição automática da gramática é baseada nesta metodologia (Constraint Grammar) e pode ser transformada em diferentes sistemas de notação especificados pelo utilizador como por exemplo essenciais ou de árvores de estruturas de dependência, texto com tags ou ainda texto com códigos coloridos.

O núcleo da linguagem VISL é os seus "Treebanks" que são bases de dados de linguagem.

O VISL é baseado em quatro conceitos: flexibilidade, interatividade, naturalidade e tutorial.

Uma importante característica do VISL é a sua capacidade de utilização do mesmo interface para diferentes fins:

- Aceder a meta textos que explicam termos gramaticais e símbolos;
- Para desfrutar de jogos de linguagem e quizzes;
- Para procurar bancos de dados.

Isto permite aos utilizadores criar sessões personalizadas de aprendizagem.

No caso desta ferramenta VISL, existem diferentes máquinas para análise de textos. As ferramentas que o VISL disponibiliza são:

- Flat Structure
- Tree Structure
- Dependency Links

A ferramenta "Flat Structure" é limitada para o que se pretende para esta dissertação porque apenas enquadra e caracteriza a palavra, não incluindo a sua

dependência. Na ilustração 2 fica ilustrada a forma de apresentação de uma frase analisada pela ferramenta "Flat Structure":

```
cada [cada] <quant> DET M S @>N
período [período] <per> N M S @SUBJ>
pode [poder] <fmc> V PR 3S IND VFIN @FAUX
incluir [incluir] <vt> V INF @IMV @#ICL-AUX<
um [um] <arti> DET M S @>N
conjunto [conjunto] <coll> <HH> N M S @<ACC
de [de] PRP @N<
dinastias [dinastia] <HH> N F P @P<
.
```

Ilustração 2 - Exemplo do resultado de uma análise sintática realizada utilizando a ferramenta "Flat Structure"

Na ferramenta "Tree Structure" o resultado é apresentado numa applet, sem possibilidade de retirar o resultado em forma de texto para que possa dessa forma ser analisado, o que impossibilitou o seu uso.

Restou então a "Dependency Links". Na ilustração 3 podemos ver o resultado da análise sintática feita à mesma frase que foi feita na ilustração 2. Na ilustração 3 podemos ver que são apresentadas as dependências de cada palavra.

```
Cada [cada] <quant> DET M S @>N #1->2
período [período] <per> N M S @SUBJ> #2->3
pode [poder] <fmc> <aux> V PR 3S IND VFIN @FS-STA #3->0
incluir [incluir] <mv> V INF @ICL-AUX< #4->3
um [um] <arti> DET M S @>N #5->6
conjunto [conjunto] <coll> <HH> <amount> N M S @<ACC #6->4
de [de] <np-close> PRP @N< #7->6
dinastias [dinastia] <HH> N F P @P< #8->7
. #9->0
</s>
```

Ilustração 3 - Resultado apresentado através da analise de uma frase utilizando o "Dependency Links"

Nesta ilustração ao analisarmos por exemplo a primeira palavra, verificasse que no fim da linha apresenta o resultado "#1->2" o que significa que esta é a palavra

numero 1 desta frase e que tem dependência da palavra numero 2. A palavra numero 2 é a palavra "período" porque apresenta o resultado "2->3" e que por seu turno tem dependência da palavra numero 3 que é o verbo "poder". Este verbo por apresentar o resultado "#3->0", significa que tem dependência da raiz da frase.

Nesta dissertação foi utilizada a ferramenta "Dependency Links" porque é a única que apresenta no seu resultado as dependências de cada palavra em forma de texto bem como a caracterização da palavra e que possibilita assim uma forma de analise do resultado da análise sintática.

No anexo I está indicado o quadro utilizado para se conseguir identificar o tipo de palavras e o que simboliza cada tag associada a cada palavra no VISL. Através desta tabela consegue-se caracterizar o tipo de palavra.

O VISL foi a ferramenta utilizada para fazer a ponte entre o texto de sistemas de informação e o desenvolvimento dos diagramas UML. Esta ferramenta foi utilizada para receber como input o texto de sistemas de informação e criar um output para ser analisado posteriormente.

2.3. Python

Para a realização desta dissertação havia a necessidade de optar por uma linguagem de programação onde se pudesse criar o diagrama de classes UML através da análise ao resultado da análise sintática, calculado através da análise do texto de sistemas de informação.

Inicialmente pensou-se na utilização da linguagem Prolog que é uma linguagem de programação em lógica matemática, no entanto mais tarde acabou por se desistir da ideia por haver uma certa dificuldade na análise do resultado da análise sintática e considerou-se a realização desta dissertação utilizando uma linguagem orientada a objetos.

Foi então que surgiu a ideia de se realizar a dissertação em Python porque era na altura uma linguagem desconhecida para mim e que preenchia os requisitos necessários para a realização desta dissertação.

O Python é uma linguagem de programação de alto nível que foi desenvolvida em 1991 por Guido Van Rossum. O Python é uma linguagem que suporta diversos tipos de programação, sendo estes tipos de programação escolhidos pelo utilizador. Nestes tipos de programação inclui-se a programação estruturada, programação orientada a objetos e possui ainda alguns elementos da programação funcional, sendo que a escolha do tipo de programação a usar fica ao critério do utilizador [CuPy].

O Python é uma linguagem de alto nível que possui diversas características diferentes. Destas características podemos enumerar as seguintes:

- Interpretada;
- Imperativa;
- Orientada a objectos;
- Tipo de dados dinâmico e forte.

Sendo uma linguagem interpretada, tal como todas as linguagens interpretadas, possui um interpretador que executa o código fonte e que é em seguida executado por um processador [WikiLInt].

A linguagem imperativa é uma linguagem que descreve a computação como ações, enunciados ou comandos que mudam as variáveis de um programa. Este é um tipo de programação que apresenta vantagens como a eficiência, flexibilidade, é intuitiva e possui um paradigma dominante [WikiPImp].

A linguagem orientada a objetos é uma linguagem que se tenta aproximar da realidade porque possui objetos e atributos, todos e partes, e classes e membros. Este tipo de linguagem permite que o envio de mensagens a objetos em que posteriormente o objeto responde a esta mensagem caso a conheça. Cada objeto pode fazer troca de mensagens com outros objetos [ProgOO].

Entre todas as características do Python, este possui algumas que se destacam bastante. O Python é uma linguagem de programação que apresenta como vantagens a qualidade do código, pois permite uma maior facilidade na leitura do código o que o torna mais fácil de reutilizar.

Outra grande vantagem do Python é a portabilidade. Ao contrário de muitas linguagens de programação, o Python pode correr em qualquer sistema operativo sem que obrigue à alteração do código fonte.

Uma vantagem que o Python disponibiliza aos seus utilizadores é a vasta biblioteca padrão que possui [CuPy].

Devido a estas características o Python permite uma rápida escrita de código e é uma linguagem de rápida execução do código por possuir um compilador byte bastante otimizado e pelas bibliotecas que o suportam [Py].

O sítio www.python.org é o sítio oficial da linguagem Python onde consta uma vasta documentação (em inglês) com informações sobre a linguagem para todo o tipo de utilizadores que pretendam utilizar esta linguagem de programação para desenvolvimento de software.

2.3. XML

A linguagem XML, ou Extensive Markup Language, é uma linguagem que derivou do SGML e tem um formato bastante simples e flexível [XML]. O XML é uma linguagem usada para criar linguagens de marcação para determinadas necessidades e tem como objetivo a facilidade de partilha de informação através da internet [WikiXML].

O XML foi criado pela World Wide Web Consortium (W3C) e foi criado para combinar a flexibilidade do SGML com a simplicidade do HTML, com o objetivo de poder ser lido por qualquer software e com capacidade de ser integrado e criado pelas mais variadas linguagens.

O XML é uma linguagem bastante semelhante ao HTML, isto porque ambas são linguagem de marcação. Apesar destas semelhanças, os objetivos destas linguagens são diferentes, isto porque o XML foi desenhado para transportar e armazenar dados baseando-se no tipo de dados e o HTML foi desenhado para mostrar os dados baseando-se na forma dos dados. No fundo o XML é uma linguagem para transportar informação enquanto o HTML é utilizado para mostrar informação [IntroXML].

O XML possibilita a criação de um número infinito de tags, tags estas que podem ser definidos pelo utilizador. O XML possibilita que esta criação seja feita ao critério do utilizador devido à inexistência de tags pré-definidas [IntroXML].

Devido à sua flexibilidade, o XML permite a interligação de diferentes bases de dados.

No XML os documentos são criados e organizados de forma hierárquica em que existe sempre um elemento que é a raiz da árvore, sendo que esta raiz pode ter vários filhos, e os seus filhos poderão ter outros filhos e assim por diante.

A ilustração 4 é um exemplo bastante simples, em que se percebe bem que a raiz "no" tem como filhos os elementos "para", "de", "cabecalho" e "corpo". A primeira linha identifica a versão do xml.

```
<?xml version="1.0"?>
<no>
    <para>Ricardo</para>
    <de>João</de>
    <cabecalho>Lembrança</cabecalho>
    <corpo>Não te esqueças!</corpo>
</no>
```

Ilustração 4 - Exemplo de código XML

Ao contrário do HTML, no XML é sempre obrigatório que os elementos tenham uma tag que termine o elemento. Outra característica importante a salientar é que o XML é "case sensitive", ou seja "<elemento>" não poderia ter como tag de fim a tag "</Elemento>".

No exemplo anterior não foi demonstrado, mas os elementos dos XML podem ter atributos, no entanto os seus atributos tem que estar sempre definidos entre aspas ou entre pelicas como nos exemplos seguintes:

- <pessoa sexo="feminino">
- <pessoa sexo='feminino'>

Num mundo em que cada vez mais os utilizadores e empresas gostam de ter liberdade para criar e impor o seu próprio estilo, o XML apresenta-se como um grande recurso para esta criatividade imposta por cada pessoa individual ou coletiva.

Nesta dissertação optou-se pela utilização desta linguagem porque o ArgoUML, apresentado no ponto seguinte, é uma ferramenta que possibilita a criação diagramas UML através da análise de documentos escritos em XML.

Tendo esta dissertação como objectivo a criação de diagramas de classe UML, o XML foi a linguagem utilizada para fazer a ponte entre a análise do texto de sistemas de informação e a representação gráfica do diagrama de classes no ArgoUML.

Para isso, após a análise do texto, é criado um ficheiro XML que posteriormente pode ser importado para o ArgoUML

2.3. ArgoUML

O ArgoUml é uma ferramenta opensource usada para modelar e desenhar diagramas UML. Para além de ser opensource, é uma ferramenta que pode ser executada em praticamente todas as plataformas pois foi implementada em java.

Atualmente é uma das ferramentas mais utilizadas para o desenvolvimento de diagramas UML e o sitio oficial desta ferramenta é http://argouml.tigris.org/ onde temos acesso aos manuais de utilização.

Optou-se por esta ferramenta, porque é uma ferramenta bastante intuitiva e também porque tem a capacidade de importar código XML. Como tal, foi adaptado o código XML à estrutura XML desta ferramenta.

Esta ferramenta é a ferramenta que vai ser utilizada para abrir os ficheiros XML que são criados através da análise dos textos de sistemas de informação. Para isso é utilizada uma opção do ArgoUML para importar ficheiros com código XML.

No anexo 10 está representado uma pequena estrutura de código XML que representa uma classe "Pessoa" com um atributo "nome". Na ilustração 5 está representado, no ArgoUML, o que o código XML no anexo 10 representa.

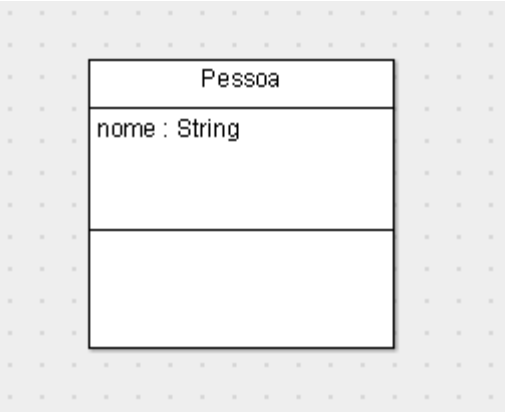

Ilustração 5 - Exemplo de uma classe com um atributo

Na ilustração 6 está representada uma imagem do ambiente de trabalho do ArgoUML.

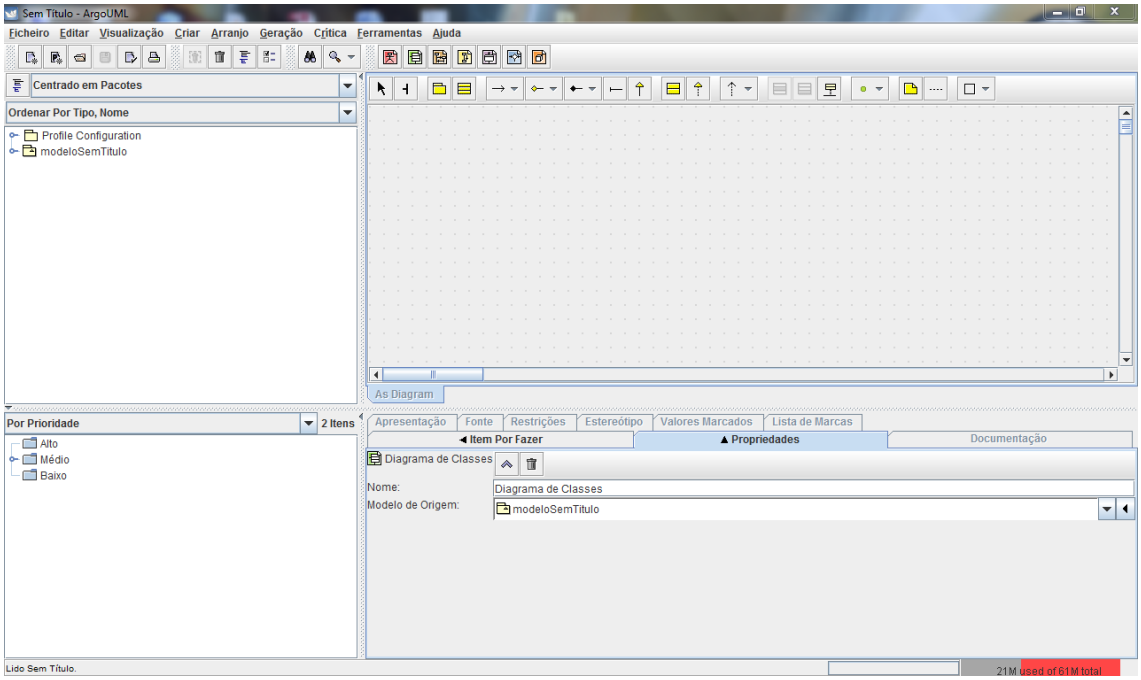

Ilustração 6 - Imagem do ArgoUML

Capitulo III – Geração de Diagramas UML com base em análise de textos que definam os sistemas de informação

3.1. Conceito

Como indicado anteriormente, esta dissertação tem como objetivo a geração de diagramas de classe, diagramas esses que muito se assemelham a bases de dados. Nestes diagramas temos a possibilidade de definir classes e dentro dessas classes definir os seus atributos.

Com esta dissertação pretende-se gerar esses mesmos diagramas automaticamente através da análise de textos que definam sistemas de informação.

Para poder gerar diagramas de classe UML foi necessário viajar por várias fases do processo, fases essas que estão representadas na ilustração 7

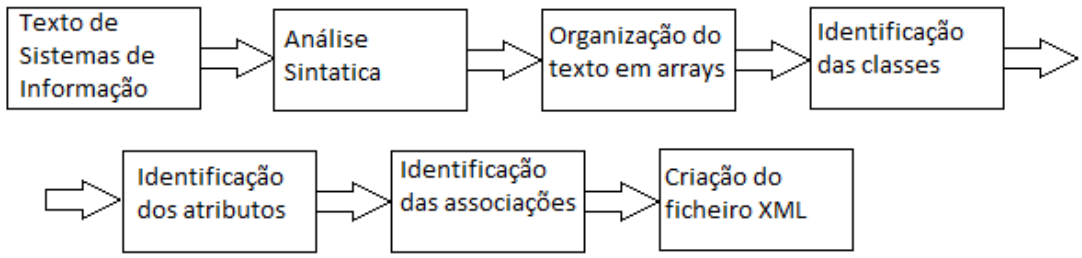

Ilustração 7 - Fases do processo

Numa primeira fase temos apenas o texto de sistemas de informação, e depois de feita a análise sintática é iniciado então o processo de geração de diagramas de classe UML, que é onde se inicia esta dissertação.

Numa fase inicial é organizada o texto de sistemas de informação em arrays para que mais tarde se possam identificar as classes dentro desse texto e posteriormente os atributos.

Mais tarde os atributos são convertidos para associações (caso as haja) e depois de indentificados todos estes objectos, é criado o ficheiro XML que possibilitará que o diagrama de classe seja importado para o ArgoUML.

Estas várias fases estarão descritas em pormenor nos próximos pontos neste mesmo capítulo.

3.2. Processamento do texto

Como indicado anteriormente, esta dissertação tem o objetivo de gerar diagramas de classe com base na análise de textos e como tal houve a necessidade de utilizar uma ferramenta que fizesse a análise palavra a palavra desses mesmos textos. Para isso utilizou-se o Visual Interactive Syntax Learnig através do sitio http://beta.visl.sdu.dk/.

Como explicado no ponto 2.2 desta dissertação, o Dependency Links foi o analisador sintático que mais argumentos apresentou para que se tornasse na opção mais viável, no entanto no inicio da realização desta dissertação apresentou alguns problemas na análise de determinadas palavras, pois apresentava erros na sua representação, ou seja, em vez de aparecer na forma como mostra a ilustração 8, apresentava-se na forma representada na ilustração 9. Como podemos ver na ilustração 8, é onde é caracterizada a palavra "realizador". A palavra "realizador" é um nome masculino no singular e depende da palavra 14 e na ilustração 9 não é caracterizado e é apenas numerado e a sua dependência é a raiz da árvore.

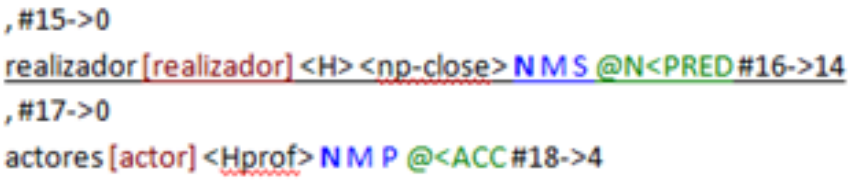

Ilustração 8 - Palavra "realizador" analisada correctamente

Ilustração 9 - Palavra realizador analisada incorrectamente

Por esse motivo foi então necessário utilizar, para além do “Dependency Links”, o “Flat Sctructure”.

O Flat Structure no entanto por si só não seria uma solução. Como demonstrado na ilustração 10, é um analisador que caracteriza apenas as palavras e não mostra de onde dependem, como tal houve a necessidade de utilizar dois analisadores por se completarem mutuamente.

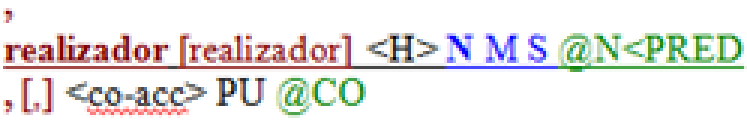

Ilustração 10 - Palavra "realizador" analisada pelo Flat Structure

Ao longo da realização desta dissertação houve melhoramentos no “Dependency Links” pelos seus criadores, o qual deixou de ter os problemas descritos anteriormente, passando a analisar corretamente todas as palavras, bem como as suas dependências. Foi por esse motivo que foi possível incluir a ilustração 8.

Devido a este melhoramento, foi então possível analisar e desenvolver este tema utilizando apenas o “Dependency Links”.

3.3. Organização do texto analisado

Para a criação dos diagramas de classe foi necessário o uso da ferramenta VISL, mais concretamente do analisador “Dependency Links” para fazer a análise dos textos. A ilustração 11 mostra o output realizado após a análise de uma frase de sistemas de informação.

Para analisar um texto de sistemas de informação houve então a necessidade de uma análise humana ao próprio texto e verificou-se que a melhor forma de o fazer seria analisar frase a frase para identificar nessa frase a classe e os respetivos atributos dessa mesma classe. Houve então a necessidade de organizar o texto frase a frase. A análise frase a frase por si só não seria suficiente, isto porque uma classe é identificada por uma

palavra ou por um conjunto de palavras e dessa forma a análise frase a frase também não seria conclusiva. Dessa forma houve a necessidade de agregar as duas formas e fazer a análise frase a frase e dentro dessa frase, palavra a palavra. Como a ferramenta utilizada foi o "Dependency Links", e como mostra a ilustração 11, o output é apresentado linha a linha, em que cada linha corresponde a uma palavra de uma determinada frase.

Ilustração 11 – Resultado da análise sintática feita à frase "Cada periodo pode incluir um conjunto de dinastias." utilizando o Dependency Links

Nesta ilustração 11, temos a oportunidade de verificar que o verbo "poder" é um verbo que tem dependência da raiz (#3->0) e que o substantivo "período" tem dependecia do verbo "poder" porque este verbo tem o numero 3 que fica representado por "#3" e o substantivo "período" tem dependência directa do verbo porque apresenta a expressão "#2->3". Por se encontrar antes do verbo, é identificado como uma classe.

Neste exemplo existe ainda o atributo "conjunto_de_dinastias" que é identificado pelo substantivo "conjunto" que tem dependência do verbo "poder" que identifica a classe. Esta dependência existe, mas de forma indirecta, pois existe através do verbo "incluir". O substantivo conjunto tem dependência directa do verbo "incluir" e este por sua vez tem dependência do verbo "poder" que identifica a classe. A palavra "dinastias" completa o nome do atributo porque tem dependência do substantivo "conjunto" através de uma proposição. Dessa forma é utilizado para completar o nome do atributo em vez de ser utilizado para identificar outro atributo.

Devido à necessidade de fazer a análise frase a frase e palavra a palavra, houve a necessidade de organizar a informação detalhadamente. Para isso foi então criado um array de arrays para guardar a informação do texto.

Criou-se então um array "Texto" em que cada posição desse array corresponde ao array "Frase" e dentro desse array são colocadas as palavras, que neste caso corresponde a cada linha do output gerado pela análise através do analisador "Dependency Links".

Esta informação seria representada esquematicamente da seguinte forma:

Texto = [frase1, frase2, frase3]

Frase1 = [frase1;palavra1, frase1;palavra2, frase1;palavra3,…]
Frase2 = [frase2;palavra1, frase2;palavra2, frase2;palavra3,…]
Frase3 = [frase3;palavra1, frase3;palavra2, frase3;palavra3,…]

3.3.1. Construção do array "Frase"

Inicialmente o processo implementado foi o acima descrito, no entanto houve a necessidade de alteração por haver frases que tem palavras entre parênteses, e que representam ou domínios dos atributos ou observações dos atributos e estas palavras teriam então que ser guardadas por outros motivos, descritos mais à frente.

Havia a obrigatoriedade de os guardar porque apesar de estarem entre parênteses, o "Dependency links" não faz essa distinção e muitas vezes os atributos seguintes dependiam de uma palavra entre parênteses, no entanto estas palavras entre parênteses nunca iriam ser uma classe ou um atributo, seriam apenas consideradas como informação adicional do atributo.

Por haver essa necessidade de distinguir uma palavra que está entre parênteses de outra que não está entre parênteses criou-se então uma classe do tipo "linhas". Esta classe "linhas" recebe então o resultado da análise do output relativo a uma determinada frase e uma variável booleana a True ou False, consoante a palavra esteja ou não entre parênteses.

Em pseudo código ficaria então da seguinte forma:

```
Se ( linha == "(" ) {
        linha <- linhaSeguinte
        Enquanto ( linha != ")" ) {
                novaLinha = linhas(True,linha)
                arrayLinhas.append(novaLinha)
                linha <- linhaSeguinte
        }
} senão {
        novaLinha = linhas(False,linha)
        arrayLinhas.append(novaLinha)
        linha <- linhaSeguinte
}
```

Desta forma os arrays "Frase" passam a ser arrays que contêm objetos do tipo "linhas". Neste pseudo código não foi indicado, no entanto, não são inseridos, nos arrays de frases, as linhas do resultado da análise sintática que simbolizem sinais de pontuações, como por exemplo pontos finais, virgulas, etc.

3.3.2. Construção do array "Texto"

Como indicado anteriormente, o array "Texto" seria um array que iria então guardar as frases que estão incluídas no texto. Pensou-se inicialmente que iria apenas guardar em cada posição o array "frase" correspondente a cada frase do texto, no entanto esse processo teve que ser, mais tarde, repensado.

Ao criar as classes, explicadas no ponto 3.4, houve a necessidade de ter acesso aos verbos que ajudam na identificação das classes e dos respetivos atributos. Não por motivos funcionais mas sim por motivos de processamento, ou seja, tendo já associado a cada frase do texto os seus verbos, identificamos mais rapidamente as suas classes.

Para identificar estes verbos que definem uma classe, foi necessário identificar que tipo de verbo está a ser analisado no momento, isto porque podemos estar perante um verbo auxiliar. Neste caso o verbo auxiliar não nos interessa para identificar uma classe. Verificou-se então que poderia ser encontrada uma classe ao encontrar um verbo finito, isto porque o verbo finito é um verbo que define claramente algo, ou seja, define claramente uma classe. Na ilustração 12 (retirado também do resultado da análise sintática ilustrada na ilustração 11) podemos verificar que o verbo "poder" está definido como verbo finito e fica representado simbolicamente através da expressão "VFIN". Desta forma sabemos que vai existir um nome antes do verbo que define uma classe.

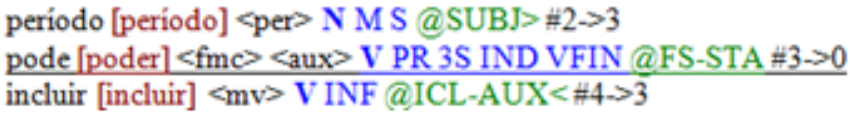

Ilustração 12 - Exemplo da representação de um verbo finito

Para guardar estas frases e associar-lhes os seus verbos, foi necessário, à semelhança do que aconteceu com as palavras, criar uma classe do tipo "frases" onde se guardar o array de palavras e o respetivo array de verbos.

Faltava então definir a forma de terminar uma frase. Através da ilustração 11, podemos ver que a frase termina sempre com a expressão "</s>", e dessa forma conseguimos identificar o fim da frase e assim colocamos o array "frase" no array "texto".

Em pseudo código pode representar-se da seguinte forma:

```
se (linha == "</s>") {
        frase = frases(arrayVerbos,arrayLinhas)
        texto.append(frase)
} se (linha contem "VFIN") {
        novaLinha = linhas(False,linha)
        arrayLinhas.append(novaLinha)
        arrayVerbos.append(linha)
} senão {
        novaLinha = linhas(False,linha)
        arrayLinhas.append(novaLinha)
```

```
}
linha <- linhaSeguinte
```

De frisar que estas condições estariam dentro de um ciclo até ao final do resultado da análise sintática.

3.4. Identificação das Classes

Para a criação de um diagrama de classes existe sempre a necessidade de identificar as classes num texto de sistemas informativos. Como tal, foi necessário desenvolver uma forma para encontrar essas mesmas classes nos textos.

Após a construção do array "texto", explicado anteriormente, podemos então iniciar a construção das classes. Para isso foi então necessário criar um array do tipo "classes", que foi uma classe criada para guardar as informações necessárias sobre as classes e identificar posteriormente os seus atributos.

Esta classe foi criada com o intuito de poder guardar todas as classes, bem como os seus atributos correspondentes. Os atributos serão apenas identificados no passo seguinte, no entanto, ao criar uma classe, existe no imediato a necessidade de guardar também o verbo à qual ficou associado, para que mais tarde, na identificação atributos, se possa calcular a dependência destes e associar à classe correspondente.

Para além da necessidade desta classe receber o nome e o verbo correspondente, existe também a necessidade de receber o número da frase em que aparece, isto porque uma classe pode aparecer em várias frases distintas. Desta forma, quando é criada uma classe, é logo associado por defeito o nome da classe, o verbo associado à classe e o número da frase em que se encontra. São então criados por defeito 3 array, que são o array de verbos, porque se a classe estiver definida em várias frases, vai estar associado a vários verbos nas diferentes frases, o array de frases que contem os números das frases em que o verbo aparece e o array de atributos inicializado sem atributos. Para além

desta informação, é também gerado um id que fica associado à classe. Este id é criado como uma string, criado através do nome da classe e do método time() do python.

Em pseudo código, o construtor desta classe ficaria da seguinte forma:

```
construtor(nome,verbo,numeroFrase,id){
    nome <- nome
    arrayVerbos <- [verbo]
    arrayFrases <- [numeroFrase]
    arrayAtributos <- []
    id <- id
}
```

Esta classe para além deste construtor tem também funções para adicionar novos verbos, novos números de frases e novos atributos, para além ainda das funções para devolver os valores já associados à classe.

Depois de criada esta classe é então necessária a pesquisa das classes no array "texto".

Para iniciar esta pesquisa, foi então necessário perceber que tipo de verbos podem aparecer na frase e neste caso é necessário distinguir dois tipos de verbos:

- Verbos que tem dependência da raiz;
- Verbos que tem dependência de um nome.

Cada frase pode conter uma ou mais classes e é importante distinguir os verbos que as identificam para mais tarde se conseguir identificar os seus atributos. No caso dos verbos que dependem da raiz, estes contem sempre a expressão "->0" e no caso dos verbos que dependem de um nome são sempre antecedidos de um nome, nome esse que identifica uma classe como mostra a ilustração 13.

Na ilustração 13 podemos ver o exemplo em que o verbo "ser" marcado com o número dois é um verbo que depende da raiz e identifica a classe "dinastia" marcada com o número um.

No outro caso, ainda na ilustração 13, podemos ver que o verbo “interessar” marcado com o número 4 é um verbo que depende diretamente de um nome, sendo que este nome tem que ser considerado como uma classe. Ambos contem a expressão “VFIN” que os identifica como verbos finitos.

Como podemos ver, as classes podem ser encontradas de duas formas diferentes e dessa forma houve a necessidade de distinguir estes dois tipos de verbos quando estes aparecem numa frase.

Em pseudo código ficou definido da seguinte forma:

```
Se ((palavra not entreParenteses) and linha contem(“->0”) and linha contem(“VFIN”)):
        arrayClasses <- verboRaiz()
Se (palavra not entreParenteses and linha contem(“VFIN”) and
verboDependeDirectamenteDeNome() == Verdade ):
        arrayClasses <- verboNaoRaiz()
```

```
cada [cada] <quant> DET M S @>N #1->2
tipo [tipo] <meta> <H> <ac-sign> N M S @SUBJ> #2->5
de [de] <np-close> PRP @N< #3->2
1 dinastia [dinastia] <HH> N F S @P< #4->3
2 é [ser] <fmc> <aux> V PR 3S IND VFIN @FS-STA #5->0
numerada [numerar] <vH> <mv> V PCP F S @ICL-AUX< #6->5
e [e] KC @NPHR #7->0
corresponde- [corresponder] <hyfen> <fmc> <mv> V PR 3S IND VFIN @FS-STA #8->5
lhe [ele] PERS M/F 3S DAT @<DAT #9->8
um [um] <arti> DET M S @>N #10->12
determinado [determinar] <vH> V PCP M S @>N #11->12
número [número] <ac> <ac-sign> <sem> <ac-cat> N M S @<ACC #12->8
de [de] <np-close> PRP @N< #13->12
3 governadores [governador] <Hprof> N M P @P< #14->13
, #15->0
em [em] <clb> PRP @PIV> #16->18
que [que] <clb-fs> <rel> SPEC M S @P< #17->16
4 interessa [interessar] <mv> <np-close> V PR 3S IND VFIN @FS-N< #18->14
saber [saber] <mv> V INF @ICL-<ACC #19->18
o [o] <artd> DET M S @>N #20->22
seu [seu] <si> <poss 3S> DET M S @>N #21->22
nome [nome] <percep-f> <ac-cat> N M S @<ACC #22->19
```

Ilustração 13 - Exemplo de frase com duas classes

Ainda nesta ilustração 13 tivemos a oportunidade de analisar a classe "tipo_de_dinastia". O nome desta classe é constituído por dois nomes, como tal foi necessário desenvolver uma forma de identificar apenas uma classe e não duas, visto a palavra "tipo" e a palavra "dinastia" serem dois nomes que dependem do verbo "ser" e que se encontram antes do verbo. Para isso foi necessário verificar se depois do primeiro nome não existe uma preposição. Neste exemplo existe a preposição "de" e que liga os dois nomes. Desta forma foi possível criar apenas uma classe com o nome "tipo_de_dinastia" em vez de duas classes, uma com o nome "tipo" e outra com o nome "dinastia".

Este processo é utilizado sempre que é encontrado o nome de uma classe composta por dois nomes.

3.4.1. Identificação das classes através de verbos que tem dependência da raiz

Identificada então esta diferença entre as formas de representação dos verbos, podemos então iniciar a pesquisa para procurar as classes. Para isso percorre-se o array "texto" frase a frase, e em cada frase, palavra a palavra, ou se se pretender, linha a linha se estivermos a imaginar o ficheiro do resultado da análise sintática como input.

Ao encontrarmos um verbo que seja finito e que não esteja entre parênteses, é necessário então verificar se depende da raiz ou de um verbo. Caso dependa da raiz iria ser necessário pesquisar do início da frase até ao verbo, ou seja na ilustração 13 iriam ser analisadas as primeiras 4 linhas para procurar o (s) nome (s) antes do verbo e assim criar a (s) classe (s). No caso da ilustração 13, a classe a criar seria "tipo_de_dinastia". Neste caso em específico, poderia optar-se por criar uma classe apenas como "dinastia", no entanto nunca se poderia ignorar a palavra "tipo" senão haveria o risco de criar duas classes, uma com o nome "tipo" e outra com o nome "dinastia" porque neste caso ambas são nomes e ambas dependem do verbo "ser".

Durante essa pesquisa das classes, iria existir uma condição para adicionar um nome como uma classe e essa condição seria a obrigatoriedade de depender do verbo, caso contrário não seria adicionado o nome como uma classe. Para verificar se depende de facto do verbo, é utilizado um método recursivo chamado "nomeDerivaVerbo".

Este método, é um método recursivo que devolve um valor booleano caso confirme se o nome tem dependência ou não do verbo.

Após a confirmação é necessário saber, tal como acontece na ilustração 13, se existe uma preposição, ou seja, uma palavra que liga dois termos de oração [WikiPre] como por exemplo "de" ou "em" e no caso do VISL é representado pela expressão "PRP". Nos casos em que existe uma preposição a seguir a um nome é chamada uma função que constrói o nome conforme ele é adicionado ao array de classes, ou seja, vai procurar até encontrar o próximo nome que no caso da ilustração 13, é a palavra "dinastia".

Depois de construído o nome segundo estas regras, é então necessário adicionar a classe e para isso existe a necessidade de verificar se a classe já existe ou não. Para isso foi implementada uma função que devolve "True" ou "False" caso exista ou não,

respetivamente, a classe. Se existir a classe é apenas adicionado o verbo associado à classe e o número da frase em que se encontra aos respetivos arrays (arrayVerbos e arrayFrases da classe "classes"), caso contrário é adicionado ao array de classes uma nova classe inicializada com o verbo e número de frase correspondente.

3.4.2. Identificação das classes através de verbos que tem dependência do nome

A forma de identificação das classes associadas a verbos que tem dependência do nome tem que ser obrigatoriamente diferente, isto porque é necessário procurar o nome que representa a classe e este nome pode estar no meio dos atributos. Dessa forma é necessário criar uma função que faça a pesquisa até encontrar o nome de onde depende o verbo. Se não encontrar o nome que identifica a classe é porque esse verbo não está a identificar uma classe.

Na ilustração 13 podemos verificar que o verbo "interessar" tem no fim da linha a expressão "#18->14", ou seja significa que é a palavra numero 18 e que depende da palavra 14. Se verificarmos, a palavra 14 é a palavra "governador" que é um nome e que nesta frase identifica uma classe.

Ao encontrar a classe, o processo seria idêntico ao anterior, ou seja, procura-se a classe no array de classes e caso exista associa-se apenas o verbo e o número da frase, caso não exista, adiciona-se uma nova classe ao array de classes.

3.5. Identificação dos Atributos

A pesquisa dos atributos é a ultima fase na construção das classes. Esta pesquisa foi feita de forma diferente da pesquisa feita para encontrar as classes. Com o final da pesquisa das classes ficaram então a conhecer-se as classes e as frases em que cada classe se encontra, bem como os verbos associados a essa mesma classe. Essas classes, como indicado anteriormente, ficam guardadas num array de classes.

Para pesquisar os atributos de cada classe foi então necessário utilizar esse array de classes e procurar classe a classe os atributos associados a essa classe. É nesta fase que é necessária a utilização dos arrays de verbos e de frases. Imaginemos por exemplo uma classe "dinastia" que aparece em duas frases distintas e com dois verbos que a identificam diferentes. Neste caso o array de verbos e de classes estaria enquadrado da seguinte forma:

```
Classe: "Dinastia"
arrayFrases = [1,3]
arrayVerbos = ["têm [ter] <fmc> <mv> V PR 3P IND VFIN @FS-STA #3->0",
"podem [poder] <fmc> <aux> V PR 3P IND VFIN @FS-STA #5->0"]
```

Neste caso a informação que dispomos é que a classe "dinastia" aparece na frase número 1 e nessa mesma frase está associada ao verbo "ter" e aparece ainda na frase numero 3 associada ao verbo "poder". Tendo esta informação já associada à classe, vai ser necessário procurar os atributos na frase número 1 e 3, que neste caso pertencem às posições 1 e 3 do array "Texto" e nessas posições do array "Texto" encontrar os nomes que tem dependência desse verbo.

Desta forma ficou implementado que se iria percorrer o array de classes, e depois em cada classe percorre o arrayFrases.

Em pseudo código ficaria implementado da seguinte forma:

```
i recebe tamanho do array de classes
de 0 até i fazer{
        arrayFrases <- recebe array de frases da classe na posicao i
        arrayVerbos <- recebe array de verbos da classe na posicao i
        j recebe tamanho do arrayFrases
        de 0 até j fazer {
                arrayClasses <-
adicionaAtributos(arrayClasses,i,arrayFrases[j],arrayVerbos[j],texto)
        }
```

}

Ao entrar na função "adicionaAtributos" vamos iniciar então o processo de pesquisa dos atributos, nessa frase, associados à classe em questão. Para isso é necessário saber se o verbo em questão tem dependência direta da raiz ou não. Caso tenha dependência da raiz, teremos que iniciar a pesquisa depois do verbo. É necessário iniciar a pesquisa depois do verbo no caso de depender da raiz (contem a expressão "->0"), porque pode acontecer por exemplo o caso em que existem duas classes definidas antes do verbo e não interessa colocar essas classes como atributos, ou seja, imaginemos o seguinte exemplo:

"Os actores e realizadores têm nome e morada"

Neste exemplo podemos ver que existem duas classes, que são elas "actor" e "realizador" e em ambas estas classes existem os atributos "nome" e "morada", ou seja são duas classes distintas mas com os mesmos atributos.

Se não houvesse essa condição de procurar os atributos apenas após encontrar o verbo iríamos ter na mesma duas classes "actor" e "realizador" mas para além dos atributos "nome" e "morada", a classe "actor" iria ter mais um atributo que neste caso seria "realizador" e a classe "realizador" iria ter mais um atributo que seria "actor", isto porque ao procurar os atributos todos os nomes (actor, realizador, nome e morada) têm dependência do verbo ("ter"). A pesquisa pode ser feita desta forma, ou seja, procurar os atributos depois do verbo porque sabemos que as frases se encontram na forma ativa e dessa forma os atributos irão encontrar-se sempre depois do verbo.

No caso em que os verbos não têm dependência da raiz, mas sim de um nome, ai os verbos poderão encontrar-se tanto antes como após o verbo. Na ilustração 14 está representado um exemplo de atributos que aparecem antes e depois do verbo. Está representada a classe "governador" identificada pelo verbo "interessar" e estão ainda representados os atributos "nome" e "tipo_de_cargo" antes do verbo, bem como os atributos "data_inicial" e "data_final" a seguir ao verbo.

Ilustração 14 - Exemplo em que aparecem atributos antes e depois do verbo que identifica a classe

Esta distinção apenas vai condicionar o início da pesquisa dos atributos, ou seja, se o verbo tiver dependência direta da raiz, inicia-se a pesquisa no verbo, se não tiver dependência da raiz mas sim de um nome, a pesquisa inicia-se no início da frase.

Após ser determinado o início da pesquisa na frase, é iniciada então a pesquisa.

3.5.1. Tipos de atributos

Os atributos podem ser encontrados de duas formas distintas, que neste caso podem ser caracterizados como nomes ou então como adjetivos.

Ao percorrer a frase existe sempre a necessidade de confirmar várias características sobre a palavra que estamos a analisar. A primeira característica que é necessário confirmar é se a palavra não está entre parênteses, caso esteja não verifica se é atributo. É também necessário confirmar que não é a própria classe. Estas são as características em comum, tanto no caso de ser adjetivo como nome. Após determinar

esta informação é necessário verificar se é nome. Caso seja nome é necessário confirmar se o nome tem dependência no verbo que identifica a classe.

Para confirmar se este nome tem dependência do verbo, foi criada uma função recursiva que só pára para devolver "True" caso encontre o verbo, se chegar ao fim da frase sem encontrar o verbo devolve "False" e determina-se que não é atributo.

Ao encontrar um nome que tenha dependência do verbo associado à classe existem três formas distintas de poder construir um nome ou atributo composto, ou seja atributos na forma "tipo_de_cargo" que aparecem no resultado da análise sintática como ilustrado na ilustração 15:

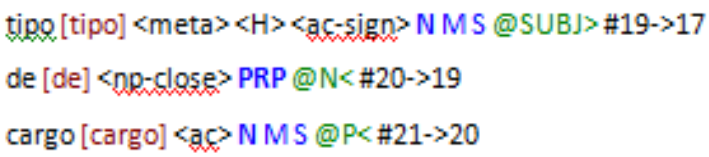

Ilustração 15 - Exemplo de dois nomes que formam um atributo

Como podemos verificar no exemplo anterior, a palavra "cargo" tem dependência do "de" e o "de" tem dependência do "tipo", sendo que o "tipo" é o nome que tem dependência do verbo, que não ficou exemplificado na ilustração 15.

Para construir nomes e representá-los na forma "tipo_de_cargo" é necessário analisar a linha seguinte do resultado da análise sintática para saber se é uma preposição, ou seja, representado por "PRP" e construir o nome composto até ao nome seguinte que tenha dependência do nome anterior. Na mesma situação das preposições, estão os advérbios para construir nomes como por exemplo "governador_mais_proeminente", que neste caso seria o "mais".

Para além da possibilidade de um nome ser seguido de uma preposição ou advérbio, pode ainda ser seguida de um adjetivo e nesse caso a forma de construção dos nomes seria idêntica, no entanto com uma ligeira diferença. No caso dos adjetivos, seria necessária a pesquisa de uma preposição que tivesse dependência do mesmo nome e se houvesse preposição com dependência do mesmo nome, é porque haveria um nome a seguir para terminar o nome composto. Caso não houvesse nome, não haveria essa

preposição e o nome composto seria constituído apelas pelo nome e pelo adjetivo e ficaria por exemplo na forma "data_inicial" como mostra a ilustração 16:

```
data [data] <temp> <Ltop> <amount> N F S @<ADVL #27->23
inicial [inicial] <nh> <np-close> ADJ F S @N< #28->27
. #29->0
</s>
```

Ilustração 16 - Exemplo de um nome seguindo de um adjectivo

Caso houvesse uma preposição iria haver um nome para terminar de definir o atributo. Imaginemos a ilustração 17:

```
data [data] <temp> <Ltop> <amount> N F S @<ADVL #27->23
inicial [inicial] <nh> <np-close> ADJ F S @N< #28->27
de [de] <sam-> <np-close> PRP @N< #29->27
o [o] <-sam> <artd> DET M S @>N #30->32
seu [seu] <poss 3S> DET M S @>N #31->32
reinado [reinado] <per> N M S @P< #32->29
. #33->0
</s>
```

Ilustração 17 - Exemplo de um nome seguindo de um adjetivo e de um nome

Na ilustração 17 existe a preposição "de" que depende do nome "data" e existe ainda o nome "reinado" que depende da preposição. Neste caso o nome composto para um novo atributo a adicionar à classe seria "data_inicial_reinado".

Existe ainda uma terceira forma de criar um nome composto após encontrar um nome, que neste caso seria um nome, que tivesse dependência no verbo, ser seguido de um outro verbo. Neste caso seriam criados nomes compostos como por exemplo "valor_licidado" em que o nome "valor" é seguido do verbo "licitar".

Caso fosse encontrado um nome que não fosse seguido de preposições, advérbios, adjetivos ou verbos, seria apenas adicionado o nome. Em todos estes casos a

forma de construção dos nomes é através da palavra que está entre parênteses retos no resultado da análise sintática.

Existe apenas um caso que é exceção, ou seja, imaginemos o seguinte exemplo representado na ilustração 18:

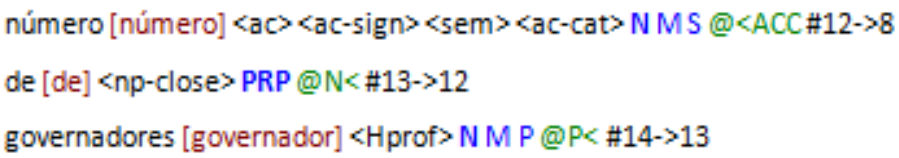

Ilustração 18 - Exemplo de um nome no plural

Neste caso, representado na ilustração 18, se o atributo fosse construído através do valor que está entre parênteses retos iria ficar com nome "numero_de_governador" e neste caso o nome do atributo deveria ser "numero_de_governadores", isto porque colocar o número do governador (que poderia ser o primeiro, segundo, terceiro, etc.) é diferente de dizer que se pretende o número de governadores. Assim para construir o nome dos atributos, foi criada uma função que distingue se o nome está no plural ou não. Caso esteja no plural escreve o nome até ao primeiro espaço, caso esteja no singular, escreve o que está entre parentes retos.

Para evitar a repetição de atributos na mesma classe, a pesquisa de novos nomes é sempre feita a partir do ponto onde se terminou a construção do nome e não do local onde se encontrou o primeiro nome, ou seja imaginemos a ilustração 19:

```
a [o] <artd> DET F S @>N #8->9
base [base] <Labs> N F S @<ACC #9->7
de [de] <np-close> PRP @N< #10->9
licitação [licitação] <occ> N F S @P< #11->10
, #12->0
o [o] <artd> DET M S @>N #13->15
seu [seu] <si> <poss 3S> DET M S @>N #14->15
valor [valor] <am> <ac> <unit> N M S @<ACC #15->9
de [de] <np-close> PRP @N< #16->15
venda [venda] <act> <inst> <cloH> N F S @P< #17->16
```

Ilustração 19 - Exemplo de um resultado de uma análise sintática feita a uma frase

Na ilustração 19 ao encontrar a palavra "base" vai construir o atributo "base_de_licitação" e em seguida continua a pesquisa na linha "seu [seu] <si> <poss 3S> **DET** M S @>N #14->15". Dessa forma evita-se a construção de dois atributos, que neste caso seria "base_de_licitacao" e "licitacao", criando apenas o atributo "base_de_licitacao".

Para criar os atributos existe ainda a possibilidade de se encontrar adjetivos soltos nas frases e neste caso é sempre necessário verificar se existe um nome associado.

Na ilustração 20 podemos verificar que o primeiro atributo identificado seria "data_inicial" e depois existe o adjetivo "final" e neste caso o que se pretende é obter dois atributos que são eles "data_inicial" e "data_final".

```
data [data] <temp> <Ltop> <amount> N F S @<ADVL #27->23
inicial [inicial] <nh> <np-close> ADJ F S @N< #28->27
e [e] KC @NPHR #29->0
final [final] <nh> <np-long> ADJ M/F S @N< #30->27
. #31->0
</s>
```

Ilustração 20 - Exemplo de um resultado de uma análise sintática com dois adjetivos com dependência de um nome

Neste caso ao encontrar o nome "data" identifica a seguir que tem um adjetivo que depende do nome e depois de construir o nome inicia a pesquisa na linha "e [e] KC@NPHR #29->0" e encontrar a palavra "final" encontra um adjetivo que tem dependência de um nome. Ao verificar esta dependência falta apenas confirmar que se o nome tem dependência do verbo da classe. Caso tenha é então adicionado novo atributo, para além do atributo "data_inicial".

Em todos estes casos, quando se faz as chamadas a funções recursivas para confirmar se o nome tem dependência do verbo que identifica a classe, no caso de frases com mais do que uma classe, todas as palavras acabam por ter dependência do verbo que tem dependência da raiz. Nestes casos não se pode correr o risco de adicionar atributos a outras classes, como tal é sempre confirmado se não "passam" por verbos que já identificam outras classes.

Depois de identificado um atributo, este atributo é associado à classe. Neste caso é adicionado ao array de atributos da classe e por defeito é associado como atributo, não sendo associado a uma associação. Para a criação do XML foi necessário criar a classe "Associacao". Assim ao criar o atributo, é necessário criar primeiro o objeto "associacao" que é inicializado por defeito como não sendo uma associação e mais tarde ao serem identificadas as associações, são alterados os valores iniciados por defeito na classe "associacao" da seguinte forma:

```
assoc = associacao("","","");
```

Após definir o objeto "assoc" este é adicionado ao objeto do tipo "atributo" e este é criado da seguinte forma:

```
atributo = atributos(nomeAtr,assoc,nomeAtr+_+id)
```

O construtor da classe "associacao" ficou definido da seguinte forma:

```
associacao(idAssociacao,classe,idClasse){
        idAssociacao = idAssociacao;
        classe = classe;
```

```
        idClasse = idClasse;
}
```

No construtor da classe "associacao" são criados os objetos "idAssociacao" que representa mais tarde o id da associação que vai ser necessário para a construção do XML, o objeto "classe" que mais tarde recebe o nome da classe onde vai ligar e ainda o objeto "idClasse" que é o id da classe onde irá ligar. Não foi necessário adicionar o id da própria classe, porque esse já se sabe qual é quando estiver a ser construído o XML.

Na classe "atributos" o atributo é criado por defeito na seguinte forma:

```
atributo(atributo,associacao,id){
        nome = atributo;
        associacao = associacao;
        id = id
}
```

No construtor da classe "atributo" o objeto "associacao" é do tipo "associacao" e os objetos "nome" e "id" são strings.

3.6. Associações

O cálculo das associações é o último passo para a construção dos diagramas de classe UML.

No UML existem quatro tipos de associações entre classes, cada uma com as suas características específicas [UMLBas].

A associação simples foi o tipo de associação que foi usada nesta dissertação porque é o tipo de associação mais usado no UML e porque seria o mais correto, embora se pudessem adaptar outros tipos de associações. Este tipo de associação tem apenas o objetivo de mostrar o relacionamento e a dependência de uma classe com outra.

Nesta dissertação foi apenas utilizada a associação simples e para encontrar as associações foi necessário criar uma função que percorra o array de classes e em cada posição do array de classes, chama uma função auxiliar que percorre o array de atributos dessa classe e vai comparar cada atributo com o nome das classes já existentes.

Ao encontrar um atributo que contenha no seu nome o nome da classe ou uma classe que contenha no seu nome o nome do atributo vai chamar um método da classe "atributos" que é o método "setAssociacao(associacao)", sendo que o objeto "associacao" é um objeto do tipo "Associacao" e onde foram definidos os seguintes valores:

- Id da associação
- Nome da classe onde vai ligar
- Id da classe onde vai ligar

No método "setAssociacao" é atribuído ao atributo as informações necessárias, passando assim de atributo para associação.

3.7. Criação do XML

Após a construção dos arrays onde constam as tabelas e os seus atributos, é iniciada a ultima fase do processo que é a fase que vai permitir a construção do ficheiro XML que mais tarde poderá ser importado pelo ArgoUML.

Após a análise de ficheiros XML gerados pelo ArgoUML, verificou-se que é necessário calcular a data e hora do mesmo. Para isso, e antes de iniciar a construção do ficheiro XML, é calculada a data e hora em que se iniciou este processo, para mais tarde a registar no ficheiro XML a ser importado pelo ArgoUML.

Já com a hora guardada numa variável, é criado, ou aberto, o ficheiro, que por defeito ficou com o nome "exemplo.xmi", pronto para escrita do código XML.

Para a criação do XML é então iniciado o documento XML, que ficou criado no objeto "doc". No ArgoUML existem sempre dois aspetos a definir no XML, que são o cabeçalho e o conteúdo.

O cabeçalho ficou criado da seguinte forma:

```
# cria o documento
doc = Document()

# cria o element o "xmi" com o nome "XMI"
xmi = doc.createElement('XMI')

# define quais os atributos de xmi
xmi.setAttribute("xmi.version",'1.2')
xmi.setAttribute("xmlns:UML",'org.omg.xmi.namespace.UML')
xmi.setAttribute("timestamp",data_str)

# define xmi como filho de "doc"
doc.appendChild(xmi)

# cria o element o "xmiHeader" com o nome "XMI.header"
xmiHeader = doc.createElement("XMI.header")

# define "xmiHeader" como filho de "xmi"
xmi.appendChild(xmiHeader)

# cria o element o "xmiDocumentation" com o nome "XMI.documentation"
xmiDocumentation = doc.createElement("XMI.documentation")

# define "xmiDocumentation" como filho de "xmiHeader"
xmiHeader.appendChild(xmiDocumentation)

# cria o elemento "xmiExporter" com o nome "XMI.exporter"
xmiExporter = doc.createElement("XMI.exporter")
```

```
# cria uma nota no documento
nota = doc.createTextNode("ArgoUML (using Netbeans XMI Writer version 1.0)")

# define a nota como filho de “xmiExporter”
xmiExporter.appendChild(nota)

# define “xmiExporter” como filho de “xmiDocumentation”
xmiDocumentation.appendChild(xmiExporter)

# cria o elemento “xmiExporterVersion” com o nome “XMI.exporterVersion”
xmiExporterVersion = doc.createElement("XMI.exporterVersion")
nota2 = doc.createTextNode("0.32.2(6) revised on $Date: 2010-01-11 22:20:14 +0100
(Mon, 11 Jan 2010) $ ")
xmiExporterVersion.appendChild(nota2)

# define o element “xmiExporterVersion” como filho de “xmiDocumentation”
xmiDocumentation.appendChild(xmiExporterVersion)

# cria o elemento “xmiMetaModel” com o nome “XMI.metamodel”
xmiMetaModel = doc.createElement("XMI.metamodel")

# define os atributos de “xmiMetaModel”
xmiMetaModel.setAttribute("xmi.name",'UML')
xmiMetaModel.setAttribute("xmi.version",'1.4')

# define o elemento “xmiMetaModel” como filho de “xmiHeader”
xmiHeader.appendChild(xmiMetaModel)

# cria o elemento “xmiContent” com o nome “XMI.content”
xmiContent = doc.createElement("XMI.content")

#define “xmiContent” como filho de “xmi”
xmi.appendChild(xmiContent)
```

No caso do conteúdo, este foi definido da seguinte forma:

```
# cria o elemento “umlContent” com o nome “UML:Model”
umlContent = doc.createElement("UML:Model")

# define os atributos de “umlContent”
umlContent.setAttribute("xmi.id",str(time.time())) # cria o atributo e associa-lhe um id
que é gerado através da função time() e é passado para string através da função str()
umlContent.setAttribute("name",'modeloSemTitulo')
umlContent.setAttribute("isSpecification",'false')
umlContent.setAttribute("isRoot",'false')
umlContent.setAttribute("isLeaf",'false')
umlContent.setAttribute("isAbstract",'false')

# define “umlContent” como filho de “xmiContent”
xmiContent.appendChild(umlContent)

# cria o elemento “nameSpaceOwnedElement” com o nome
"UML:Namespace.ownedElement”
nameSpaceOwnedElement = doc.createElement("UML:Namespace.ownedElement")

# define “nameSpaceOwnedElement” como filho de “umlContent”
umlContent.appendChild(nameSpaceOwnedElement)
```

Após a criação do cabeçalho e do conteúdo, é iniciado o processo que vai definir os diagramas de classe UML. Para a criação do diagrama UML vai ser utilizado o array de classes definido nos passos anteriores. Para isso foi criado um for que vai percorrer o array de classes e sempre que cria uma classe no XML vai defini-la da seguinte forma:

```
# cria o elemento “umlClass” com o nome “UML:Class”
umlClass = doc.createElement("UML:Class")

# define os atributos de “umlClass”
```

```
umlClass.setAttribute("xmi.id",classes[i].getID()) # cria o atributo com o id da classe
umlClass.setAttribute("name",classes[i].getNome()) # cria o atributo com o nome da
classe
umlClass.setAttribute("visibility",'public')
umlClass.setAttribute("isSpecification",'false')
umlClass.setAttribute("isRoot",'false')
umlClass.setAttribute("isLeaf",'false')
umlClass.setAttribute("isAbstract",'false')
umlClass.setAttribute("isActive",'false')

# define “umlClass” como filho de “nameSpaceOwnedElement”
nameSpaceOwnedElement.appendChild(umlClass)

# cria o elemento “umlClassifierFeature” com o nome "UML:Classifier.feature"
umlClassifierFeature = doc.createElement("UML:Classifier.feature")


# define o elemento “umlClassifierFeature” como filho de “umlClass”
umlClass.appendChild(umlClassifierFeature)
```

Após criar a classe no XML, é necessário definir os seus atributos, e esta é a fase mais complexa do processo de criação do XML. Para definir os atributos foi necessário nos passos anteriores definir se eram atributos ou associações, e é nesta fase que vai ser definido no XML se vão aparecer como atributos, ou se vão aparecer como associações no diagrama de classes.

Após a criação da classe no XML, é iniciado então um for que vai percorrer o array de atributos dessa classe. Para isso foi criado um for para percorrer esse array de atributos, e ao percorrer o array foi necessário criar uma forma de distinguir os atributos das associações, pois tem formas de representação no XML diferentes, como tal caso o atributo seja efetivamente um atributo, é definido no XML da seguinte forma:

```
#cria o elemento “umlAttribute” com o nome “UML:Attribute”
umlAttribute = doc.createElement("UML:Attribute")
```

```
# define os atributos do elemento “umlAttribute”
umlAttribute.setAttribute("xmi.id",atrs[j].getID()) # cria o atributo com o id do atributo
umlAttribute.setAttribute("name",atrs[j].getNome()) # cria o atributo com o nome do
atributo
umlAttribute.setAttribute("visibility",'public')
umlAttribute.setAttribute("isSpecification",'false')
umlAttribute.setAttribute("ownerScope",'instance')
umlAttribute.setAttribute("changeability",'changeable')
umlAttribute.setAttribute("targetScope",'instance')

# define o elemento “umlAttribute” como filho de “umlClassifierFeature”
umlClassifierFeature.appendChild(umlAttribute)

# cria o elemento “umlStructuralFeatureMultiplicity” com o nome
“UML:StructuralFeature.multiplicity”
umlStructuralFeatureMultiplicity =
doc.createElement("UML:StructuralFeature.multiplicity")

# define o elemento “umlStructuralFeatureMultiplicity” como filho de “umlAttribute”
umlAttribute.appendChild(umlStructuralFeatureMultiplicity)

# cria o elemento “umlMultiplicity” com o nome “UML:Multiplicity”
umlMultiplicity = doc.createElement("UML:Multiplicity")

# define os atributos de “umlMultiplicity”
umlMultiplicity.setAttribute("xmi.id","multiplicity_" + atrs[j].getNome() + "_" +
str(time.time())) # define o id deste atributo atraves do nome do atributo e de um id
gerado através da função time() e convertido para string através da função str()

# define “umlMultiplicity” como filho de “umlStructuralFeatureMultiplicity”
umlStructuralFeatureMultiplicity.appendChild(umlMultiplicity)

#cria o elemento “umlMultiplicityRange” com o nome “UML:Multiplicity.range”
```

```
umlMultiplicityRange = doc.createElement("UML:Multiplicity.range")

# define o elemento “umlMultiplicityRange” como filho de “umlMultiplicity”
umlMultiplicity.appendChild(umlMultiplicityRange)

# cria o elemento “umlMultiplicityRange2” com o nome “UML:MultiplicityRange”
umlMultiplicityRange2 = doc.createElement("UML:MultiplicityRange")

# define os atributos de “umlMultiplicityRange2”
umlMultiplicityRange2.setAttribute("xmi.id","multiplicityRange_" +atrs[j].getNome() +
"_" + str(time.time()))# define o id deste atributo atraves do nome do atributo e de um
id gerado através da função time() e convertido para string através da função str()
umlMultiplicityRange2.setAttribute("lower",'1')
umlMultiplicityRange2.setAttribute("upper",'1')

# define “umlMultiplicityRange2” como filho de “umlMultiplicityRange”
umlMultiplicityRange.appendChild(umlMultiplicityRange2)
```

Se for uma associação, esta vai ter sempre duas ligações essenciais, ou seja, a ligação da classe de onde parte a associação, que neste caso é a própria classe e ligação de fim que é a classe onde vai ligar esta mesma associação. O id da classe onde a associação vai ligar, está definido em cada atributo.

No caso das associações, são então necessários definir 3 pontos distintos no XML. É necessário definir a associação, é necessário definir a classe de onde parte a associação e é necessário definir a classe onde vai ligar a associação.

A associação é definida da seguinte forma:

```
# cria o elemento “umlAssociation” com o nome “UML:Association”
umlAssociation = doc.createElement("UML:Association")

# define os atributos do elemento “umlAssociation”
```

```
umlAssociation.setAttribute("xmi.id",associ.getIDAssociacao()) # define o atributo com o id da associação
umlAssociation.setAttribute("name",atrs[j].getNome()) # define o atributo com o nome da associação
umlAssociation.setAttribute("isSpecification",'false')
umlAssociation.setAttribute("isRoot",'false')
umlAssociation.setAttribute("isLeaf",'false')
umlAssociation.setAttribute("isAbstract",'false')

# define o elemento “umlAssociation” como filho de “nameSpaceOwnedElement”
nameSpaceOwnedElement.appendChild(umlAssociation)

# cria o elemento “umlAssociationConnection” com o nome “UML:Association.connection”
umlAssociationConnection = doc.createElement("UML:Association.connection")

# define o elemento “umlAssociationConnection” como filho de “umlAssociation”
umlAssociation.appendChild(umlAssociationConnection)
```

O início da associação é definido da seguinte forma:

```
# define o elemento “umlAssociationEndInicio” com o nome “UML:AssociationEnd”
umlAssociationEndInicio = doc.createElement("UML:AssociationEnd")
# define os atributos do elemento “umlAssociationEndInicio”
umlAssociationEndInicio.setAttribute("xmi.id",str(time.time())+"_inicio") # define o atributo através da função time() e com o seu resultado convertido para string através da função str() mais a expressão “_inicio” (por ser o inicio da associação)
umlAssociationEndInicio.setAttribute("visibility",'public')
umlAssociationEndInicio.setAttribute("isSpecification",'false')
umlAssociationEndInicio.setAttribute("isNavigable",'true')
umlAssociationEndInicio.setAttribute("ordering",'unordered')
umlAssociationEndInicio.setAttribute("aggregation",'none')
umlAssociationEndInicio.setAttribute("targetScope",'instance')
umlAssociationEndInicio.setAttribute("changeability",'changeable')
```

```
# define o elemento “umlAssociationEndInicio” como filho de
“umlAssociationConnection”
umlAssociationConnection.appendChild(umlAssociationEndInicio)

# define o elemento “umlAssociationEndParticipantInicio” com o nome
“UML:AssociationEnd.participant”
umlAssociationEndParticipantInicio =
doc.createElement("UML:AssociationEnd.participant")

# define “umlAssociationEndParticipantInicio” como filho de “umlAssociationEndInicio”
umlAssociationEndInicio.appendChild(umlAssociationEndParticipantInicio)

# cria o elemento “umlClassInicio” com o nome “UML:Class”
umlClassInicio = doc.createElement("UML:Class")

# define o atributo do elemento “umlClassInicio”
umlClassInicio.setAttribute("xmi.idref",classes[i].getID()) # define o id através do id da
classe

# define o elemento “umlClassInicio” como filho de
“umlAssociationEndParticipantInicio”
umlAssociationEndParticipantInicio.appendChild(umlClassInicio)
```

O fim da associação é definido da seguinte forma:

```
# cria o elemento “umlAssociationEndFim” com o nome “UML:AssociationEnd”
umlAssociationEndFim = doc.createElement("UML:AssociationEnd")

# define os atributos do elemento “umlAssociationEndFim”
umlAssociationEndFim.setAttribute("xmi.id",str(time.time())+"_fim") # define o
atributo através da função time() e com o seu resultado convertido para string através
da função str() mais a expressão “_fim” (por ser o fim da associação)
umlAssociationEndFim.setAttribute("visibility",'public')
```

```
umlAssociationEndFim.setAttribute("isSpecification",'false')
umlAssociationEndFim.setAttribute("isNavigable",'true')
umlAssociationEndFim.setAttribute("ordering",'unordered')
umlAssociationEndFim.setAttribute("aggregation",'none')
umlAssociationEndFim.setAttribute("targetScope",'instance')
umlAssociationEndFim.setAttribute("changeability",'changeable')

# define o elemento “umlAssociationEndFim” como filho de
“umlAssociationConnection”
umlAssociationConnection.appendChild(umlAssociationEndFim)

# cria o elemento “umlAssociationEndParticipantFim” com o nome
“UML:AssociationEnd.participant”
umlAssociationEndParticipantFim =
doc.createElement("UML:AssociationEnd.participant")

# define o elemento “umlAssociationEndParticipantFim” como filho de
“umlAssociationEndFim”
umlAssociationEndFim.appendChild(umlAssociationEndParticipantFim)

# cria o elemento “umlClassFim” com o nome “UML:Class”
umlClassFim = doc.createElement("UML:Class")

# define os atributos do elemento “umlClassFim”
umlClassFim.setAttribute("xmi.idref",associ.getIDClasse()) # atribui o id da classe onde
a associação vai ligar

# define o elemento “umlClassFim” como filho de “umlAssociationEndParticipantFim”
umlAssociationEndParticipantFim.appendChild(umlClassFim)
```

Depois de definidos o cabeçalho, o conteúdo, as classes, os atributos e as associações, é criado então o XML. Depois de criado o código XML, este é escrito no ficheiro e este é fechado.

Esta construção dos nós e respectivos filhos e atributos é construída segundo a análise da estrutura dos ficheiros XML que são criados pelo ArgoUML.

Depois de criado o ficheiro “exemplo.xmi”, este está pronto para ser importado pela ferramenta ArgoUML.

Capitulo IV – Casos de estudo

Para a realização de qualquer trabalho, existe sempre a necessidade de analisar diversos exemplos práticos para testar o que foi realizado. Ao realizar esta dissertação tive a oportunidade de analisar quatro textos diferentes que representam diferentes sistemas de informação.

Em cada um desses exemplos tive a oportunidade de me deparar com diferentes características e diferentes formas de representação de sistemas de informação. Nos próximos pontos irão ser apresentados estes quatro textos de sistemas de informação e em cada texto será explicado frase a frase, como se procedeu à identificação das classes e dos seus atributos.

Estes casos de estudo foram realizados até ao dia 10 de Outubro de 2011, data de entrega desta dissertação. Devido a constantes atualizações no VISL, a partir dessa data alguns dos textos de sistemas de informação geraram um resultado diferente após a análise sintática.

4.1. Clube de vídeo

Este exemplo representa um clube de vídeo onde existe a possibilidade de criar uma aplicação para armazenar e consultar informações sobre filmes. Neste caso o texto apresentado foi o seguinte:

"Pretende-se uma aplicação para armazenar e consultar os filmes.
Os filmes contem título, género, país de origem, ano de realização, realizador, actores e duração em minutos. Os realizadores e actores têm endereço na Internet. O endereço contem link. O endereço contém uma descrição."

Analisando o texto à primeira vista conseguimos identificar de imediato que a primeira frase é meramente informativa, sendo que as restantes frases explicam e caracterizam o sistema de informação. Seria então necessário analisar este texto utilizando o "Dependency Links". Deste texto de sistemas de informação resulta o esquema anexado como anexo II a esta dissertação.

Neste exemplo ao analisar o enunciado saltam logo à vista as classes "filme", "realizador", "actor" e "endereço".

A primeira frase do texto inicia-se com um verbo, que neste caso é o verbo "Pretender" e por esse motivo esta frase não vai ser analisada, isto porque ao encontrar o verbo na primeira posição não vai encontrar nomes antes do verbo e dessa forma não pesquisa neste frase por classes e no passo seguinte, visto não ter nenhuma classe a apontar para a primeira frase, também não irá procurar atributos. A análise do texto é feita como no tratamento de arrays, ou seja, inicia-se a 0, ficando desta forma esta frase como frase 0 e no array "texto" ficaria guardada na posição 0.

Na frase seguinte está identificada a classe "filme", isto porque filme é um nome e encontra-se antes do verbo e é ainda um nome que tem dependência do verbo. Ao identificar esta classe, esta seria criada e associada ao verbo "contem". Ao ser analisada a segunda frase, verificamos que esta já tem duas classes numa só frase e que tem ambas os mesmos atributos. Por este motivo é que foi necessário terminar apenas a pesquisa ao encontrar o verbo, porque ao terminar a pesquisa após encontrar a classe, poderia ser ignorada uma classe identificada no texto. Como podemos ver no Anexo II, são ambos nomes e estão antes do verbo. No caso da palavra "realizador" tem dependência direta do verbo, e a palavra "actor" tem também dependência do verbo, mas através do nome "realizador". Estas duas classes ao serem criadas, iriam ficar ambas associadas ao verbo "ter" e em ambas iria guardar-se a posição da frase, que neste caso seria a frase 2. Nas últimas duas frases temos a oportunidade de ver que a classe é a mesma, no entanto são identificados atributos diferentes em cada uma das frases. Neste caso em específico, a classe seria criada na frase número 3 em que fica associada na frase 3 ao verbo contem ao ser analisada a frase 4, vai ser detetada já a existência desta classe e é então adicionado ao array de frases o número 4 (numero da ultima frase) e o verbo "ter" da última frase.

Após termina a pesquisa das frases, inicia-se o processo de pesquisa dos atributos. O resultado da construção das classes ficou guardado da seguinte forma:

filme

[1]

['contem [conter] <fmc> <mv> V IMP 2S VFIN @FS-COM #3->0\n']

realizador

[2]

['têm [ter] <fmc> <mv> V PR 3P IND VFIN @FS-STA #5->0\n']

actor

[2]

['têm [ter] <fmc> <mv> V PR 3P IND VFIN @FS-STA #5->0\n']

endereço

[3, 4]

['contem [conter] <fmc> <mv> V IMP 2S VFIN @FS-COM #3->0\n',

'tem [ter] <fmc> <mv> V PR 3S IND VFIN @FS-STA #3->0\n']

Neste esquema podemos ver as classes detetadas bem como o array de frases e o array de verbos aos quais ficaram associados. Os arrays de número de frases e de verbos, são sempre construídos em simultâneo.

Após encontradas as classes, são então procurados os atributos utilizando o número da frase colocado numa determinada posição do array de frases, e na mesma posição do array de verbos encontra-se o verbo dessa frase que identifica a classe.

4.1.1. Avaliação

Para este texto de sistema de informação o resultado foi bastante satisfatório porque conseguiu-se para um texto relativamente simples, um resultado que representa claramente as classes identificadas no texto, bem como os seus atributos.

O resultado apresentado foi o seguinte:

Classe: filme, ID: filme_1330287568.52

Atributo(s):

titulo: ID = titulo_1330287568.567

genero: ID = genero_1330287568.598

pais_de_origem: ID = pais_de_origem_1330287568.614

ano_de_realizacao: ID = ano_de_realizacao_1330287568.645

realizador (associacao da classe: realizador): ID = realizador_1330287568.676

actores (associacao da classe: actor): ID = actores_1330287568.692

duracao_em_minutos: ID = duracao_em_minutos_1330287568.739

Classe: realizador, ID: realizador_1330287568.52

Atributo(s):

endereco_em_Internet (associacao da classe: endereco): ID =

endereco_em_Internet_1330287568.754

Classe: actor, ID: actor_1330287568.536

Atributo(s):

endereco_em_Internet (associacao da classe: endereco): ID =

endereco_em_Internet_1330287568.77

Classe: endereco, ID: endereco_1330287568.536

Atributo(s):

link: ID = link_1330287568.786

descricao: ID = descricao_1330287568.832

Neste resultado apresentam-se as classes e os seus id's. No caso dos atributos, os que apresentam a expressão "associação da classe:" significa que são associações dessa mesma classe dentro dos parênteses. Depois de representados os atributos, aparece também o seu id.

Depois de calculado este resultado, é criado o XML, que está representado no anexo VI, que depois de importado pelo ArgoUML apresenta o resultado na ilustração 21.

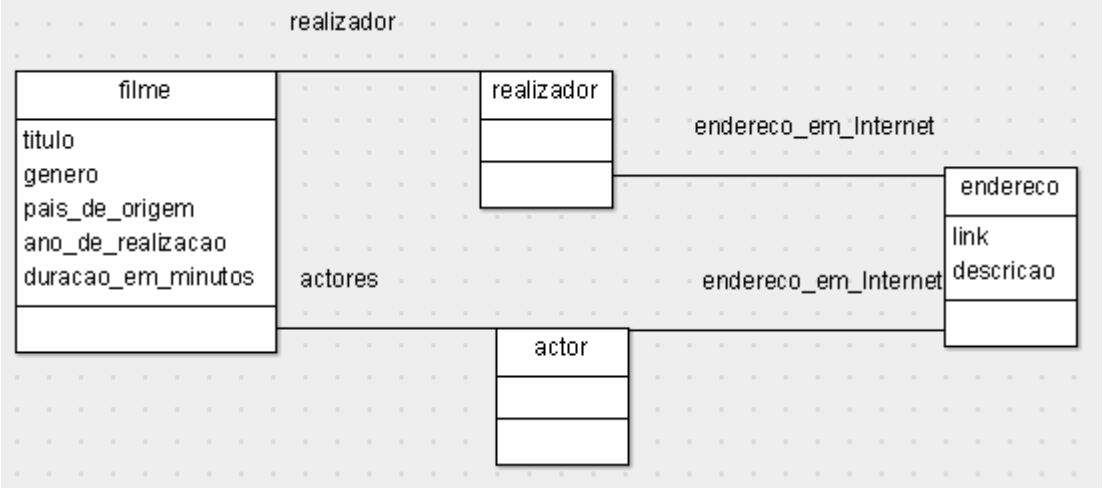

Ilustração 21 - Resultado do primeiro exemplo

Este resultado apresentado era o esperado tendo em conta o texto apresentado.

4.2. Leilão

Este exemplo representa um leilão de artigos feitos online. Neste caso o objetivo é construir as classes "artigo" e "licitação". Para este caso, o texto apresentado foi o seguinte:

"Pretende-se desenvolver uma aplicação que disponibilize artigos através de leilões online. Para cada artigo leiloado deve existir a sua descrição e categoria (música, mobiliário, decoração, etc …). Para cada artigo é também necessário conhecer a base de licitação, o seu valor de venda e a data e a hora a partir da qual não são possíveis mais licitações. Sobre as licitações é necessário conhecer o artigo, a data, a hora, o valor licitado e a identificação do licitador."

Neste caso aparecem já casos particulares e diferentes do primeiro exemplo do clube de vídeo. Neste exemplo podemos identificar de imediato a classe artigo em duas frases distintas e na última frase está caracterizada a classe "licitação". O resultado da análise sintática realizada a este texto ficou identificado como Anexo III.

À semelhança do que aconteceu no primeiro exemplo do clube de vídeo, este texto de sistemas informativos, apresenta a primeira frase como uma frase meramente informativa, sem qualquer influência para a construção do diagrama de classes. Neste

caso esta frase também iria ser ignorada por se iniciar por um verbo e por consequência, não tem um nome antes do verbo.

A primeira frase tem como diferença o facto de ter palavras entre parênteses e neste ao analisar o anexo III, as palavras entre parênteses são caracterizadas como nomes ou adjetivos, no entanto ao ser analisado o texto de sistemas de informação é percetível que não podem ser considerados como atributos. Desta forma houve a necessidade de se diferenciar as palavras que estão entre parênteses das palavras que não estão entre parênteses.

Isto levou à criação da classe "linhas" e esta classe guarda a linha do resultado da análise sintática e associa uma variável booleana a cada linha que fica a "True" se está entre parênteses e fica a "False" se não está entre parênteses.

Após esta distinção pode então ser iniciada a pesquisa das classes. O resultado após pesquisar as classes é a seguinte:

```
artigo
[1, 2]
['deve [dever] <fmc> <aux> V PR 3S IND VFIN @FS-STA #5->0\n',
 'é [ser] <vK> <fmc> <mv> V PR 3S IND VFIN @FS-STA #4->0\n']

licitação
[3]
['é [ser] <vK> <fmc> <mv> V PR 3S IND VFIN @FS-STA #4->0\n']
```

Através deste resultado apresentado verifica-se que a classe "artigo" está caracterizada em duas frases distintas, que neste caso é na frase 1 e 2 e a classe "licitação" na frase 3.

Na fase seguinte, ou seja na procura dos atributos, este texto de sistemas informativos apresentou algumas dificuldades. Para além de apresentar palavras entre

parentes, apresenta também no resultado da análise sintática, nas frases 2 e 3, dois verbos que tem dependência direta da raiz. Na frase 2 apresenta os verbos:

- é [ser] <vK> <fmc> <mv> **V** PR 3S IND VFIN @FS-STA #4->0
- conhecer [conhecer] <vH> <mv> **V** INF @ICL-UTT #7->0

Na frase 3 apresenta os verbos:

- é [ser] <vK> <fmc> <mv> **V** PR 3S IND VFIN @FS-STA #4->0
- conhecer [conhecer] <vH> <mv> **V** INF @ICL-UTT #6->0

Ao ser analisado o resultado apresentado e ao calcular as classes, verificamos que a classe artigo na frase 2 fica associada ao verbo "ser" e na frase 3 a classe "licitações" fica também associado ao verbo "ser".

Ao serem calculados os verbos, verificou-se que nas frases 2 e 3 os artigos tem dependência do segundo verbo dessas mesmas classes que tem dependência direta da raiz. Como tal, nesta situação nunca se conseguiria encontrar os atributos explícitos nessas mesmas frases.

Houve a necessidade de fazer uma alteração à forma de encontrar os atributos nestes casos específicos.

O verbo "ser" pode ser considerado um verbo auxiliar e nesse caso, existe a possibilidade de haver outro verbo que identifique os atributos da classe. Neste caso ao ser encontrado um atributo, verifica se esse atributo depende de um verbo que tem dependência direta da raiz. Ao depender da raiz, significa que nunca irá depender do verbo que identifica a classe. Neste caso em específico, o que é feito é verificar se o verbo de onde dependem os atributos tem dependência da raiz e se tiver vai confirmar se o verbo que identifica a classe é o verbo auxiliar "ser". Caso cumpra estes dois requisitos é adicionado à classe.

4.2.1. Avaliação

A avaliação deste texto de sistemas de informação pode considerar-se positiva apesar de precisar de dois reparos

No primeiro reparo, ao analisarmos o texto humanamente a frase numero dois que diz:

"Para cada artigo é também necessário conhecer a base de licitação, o seu valor de venda e a data e a hora a partir da qual não são possíveis mais licitações."

Nesta frase podemos identificar os dois últimos atributos que são eles:

- data_a_partir_da_qual_não_são_possiveis_mais_licitações;
- hora_a_partir_da_qual_não_são_possiveis_mais_licitações.

Tentou-se que o objetivo desta dissertação fosse a apresentação dos atributos o mais corretamente possível, no entanto no caso da "data" conseguiu-se apenas que ficasse representada como "data" porque se verificarmos o anexo III onde se representa o resultado da análise sintática deste exemplo, verificou-se que não existe dependência das restantes palavras que se pretendia incluir no nome do atributo do próprio nome. Conseguiu-se apenas que a "hora" ficasse com o nome completo porque existe a tal dependência.

O segundo reparo que poderia ser feito era o facto do atributo "base_de_licitação" ser associação quando também poderia não o ser.

O resultado apresentado após a análise do texto foi o seguinte:

Classe: artigo, ID: artigo_1330288124.126

Atributo(s):

descricao: ID = descricao_1330288124.251

categoria: ID = categoria_1330288124.282

base_de_licitacao (associacao da classe: licitacao): ID =

base_de_licitacao_1330288124.376

valor_de_venda: ID = valor_de_venda_1330288124.407

data: ID = data_1330288124.438

hora_a_partir_de_o_qual_nao_ser_possiveis_licitaces: ID =

hora_a_partir_de_o_qual_nao_ser_possiveis_licitaces_1330288124.485

Classe: licitacao, ID: licitacao_1330288124.173

Atributo(s):

artigo (associacao da classe: artigo): ID = artigo_1330288124.532

data: ID = data_1330288124.532

hora: ID = hora_1330288124.547

valor_licitado: ID = valor_licitado_1330288124.563

identificacao_de_licitador: ID = identificacao_de_licitador_1330288124.61

Através deste resultado, foi gerado o ficheiro XML, apresentado no anexo VII, e após importado para o ArgoUML o resultado está ilustrado na ilustração 22.

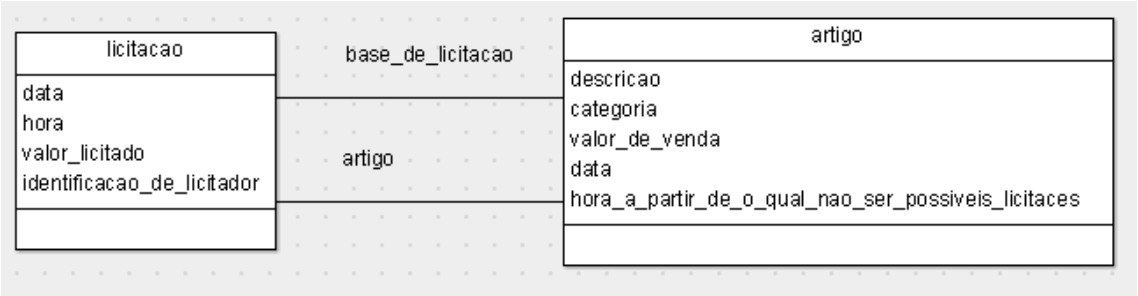

Ilustração 22 - Resultado do segundo exemplo

Tendo em conta o texto apresentado, este seria o resultado esperado.

4.3. Tesouraria

Neste exemplo o objetivo era informatizar uma tesouraria. O primeiro exemplo apresentado foi o seguinte:

"Pretende-se informatizar uma tesouraria de uma instituição.
À tesouraria dirigem-se pessoas que pagam facturas.

As facturas têm um número único.
As pessoas podem pagar facturas em numerário, ou por POS, ou cheque.
Numerário, ou por POS, ou cheque são meios de pagamento."

Este primeiro exemplo apresentou algumas dificuldades, em primeiro porque ao serem analisadas as duas últimas frases verificamos que na frase numero 3 são apresentadas as várias formas de pagamento e na frase 4 aparecem repetidas essas mesmas formas de pagamento numa frase na forma passiva.

Neste caso haveria a necessidade de se criar uma função para otimizar o diagrama, ou seja, que verificasse todas as classes e os seus atributos e verificasse se todos esses atributos da classe que se está a verificar estão incluídos numa outra classe, caso estejam, esse mesmos atributos são apagados e criado um novo atributo com o nome da classe. Dessa forma a classe "Pessoa" em vez de apresentar os três atributos "numerário", "POS" e "cheque" passaria a apresentar apenas um atributo que seria "meios_de_pagamento" que é a classe identificada na frase seguinte.

A segunda dificuldade apresentou-se na última frase, ou seja, se verificarmos a frase, temos a oportunidade de verificar que se encontra na forma passiva e ao analisar a última frase como ela se encontra seriam criadas três classes e todas elas com o atributo "meios_de_pagamento". Não era isto que se pretendia, como tal houve a necessidade de alterar a frase para a seguinte forma:

"Os meios de pagamento podem ser por numerário, POS ou cheque."

Desta forma cria apenas uma classe "meios_de_pagamento" com os três atributos "numerário", "POS" e "cheque".

A última dificuldade neste exemplo foi a expressão "POS", isto porque "POS" fica caracterizado da seguinte forma (também mostrado no anexo IV):

POS [POS] <party> **PROP** M S @<SC #10->7

Sendo uma palavra caracterizada na forma "PROP", esta não é viável que seja aceite como um atributo, isto porque a expressão "PROP" significa que estamos perante

um nome próprio e se aparecesse no texto, por exemplo, o nome de uma pessoa seria caracterizado desta forma e um nome próprio não deve ser considerado como um atributo. Desta forma a expressão "POS" teve que ser substituída por "Ponto de serviço".

Chegou-se então à conclusão que o texto final de sistemas de informação deveria ser apresentado na seguinte forma:

> "Pretende-se informatizar uma tesouraria de uma instituição.
> À tesouraria dirigem-se pessoas que pagam facturas.
> As facturas têm um número único.
> As pessoas podem pagar facturas em numerário, ou por ponto de serviço, ou cheque.
> Os meios de pagamento podem ser por numerário, ponto de serviço ou cheque."

O resultado da análise sintática realizada ao texto final está apresentado no anexo IV.

Apesar das dificuldades enumeradas anteriormente, verificou-se que estavam a ser atribuídos à mesma classe, atributos idênticos, ou seja, à classe "Pessoas" estava a ser atribuído o atributo "facturas" indicado na frase número 1 e na frase três era associado o verbo "facturas_em_numerario" e neste caso são idênticos, como tal houve a necessidade de atribuir apenas o segundo por ser mais detalhado, então na classe "classes" onde se define a classe e os seus atributos, foi alterada a função para incluir novo atributo, ou seja antes de incluir novo atributo vai verificar se este já existe para não incluir em duplicado.

Por último, para completar a análise deste exemplo, havia agora então a necessidade de otimizar o diagrama para deixar de ter a classe "Pessoa" com três atributos e transformá-los num só com o nome "meios_de_pagamento".

Ao analisar as duas classes verificou-se que ambas são idênticas, alterando apenas o nome da classe porque os atributos são, no seu todo, idênticos, como tal pensou-se de que forma se poderia diferenciar as duas classes para mostrar à aplicação qual seria a classe em que deveriam ser alterados os atributos. Talvez fizesse sentido alterar na classe que tem mais atributos removendo nessa classe os atributos que estão incluídas na segunda classe e adicionando um atributo com o nome da segunda classe, mas e neste caso em que ambas têm o mesmo número de atributos?

Teoricamente essa substituição poderia ser feita na primeira classe que se encontra porque tal como neste exemplo, apenas numa frase seguinte se explica a que classes podem pertencer esses mesmos atributos, mas seria sempre de certa forma subjetivo.

4.3.1 Avaliação

Neste exemplo apesar das dificuldades mais evidentes relativamente aos exemplos anteriores, foi apresentado um resultado satisfatório, apesar de ter deixado a ideia de que este resultado já iria requerer uma análise humana mais detalhada.

O resultado apresentado foi o seguinte:

Classe: tesouraria, ID: tesouraria_1330288890.129
Atributo(s):
pessoas (associacao da classe: pessoa): ID = pessoas_1330288890.223

Classe: pessoa, ID: pessoa_1330288890.145
Atributo(s):
facturas_em_numerario (associacao da classe: factura): ID =
facturas_em_numerario_1330288890.426
ponto_de_servico: ID = ponto_de_servico_1330288890.473
cheque: ID = cheque_1330288890.504

Classe: factura, ID: factura_1330288890.145
Atributo(s):
numero_unico: ID = numero_unico_1330288890.535

Classe: meios_de_pagamento, ID: meios_de_pagamento_1330288890.161
Atributo(s):
numerario: ID = numerario_1330288890.613
ponto_de_servico: ID = ponto_de_servico_1330288890.629
cheque: ID = cheque_1330288890.675

O resultado no ArgoUML, depois de criado o XML, anexado no anexo VIII, está ilustrado na ilustração 23.

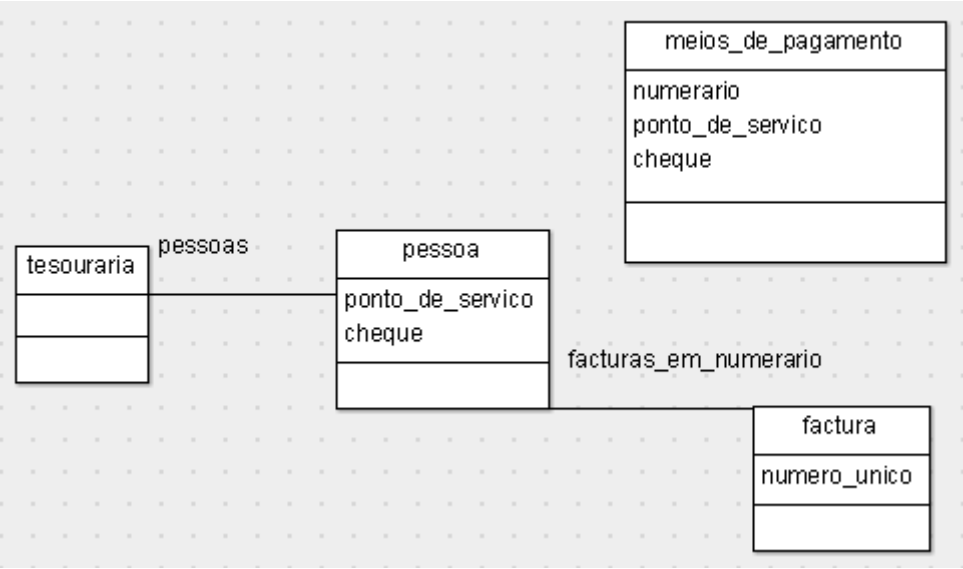

Ilustração 23 - Resultado do terceiro exemplo

Na ilustração 23 podemos verificar que é feita a criação da classe meios de pagamento, no entanto a classe fatura fica associada à classe "pessoa" porque a classe "pessoa" é que contem a associação "facturas_em_numerario". O mais correto seria a classe "pessoa" estar associada à classe "meios_de_pagamento" e a classe "meios_de_pagamento" é que ficaria por sua vez associada à classe "factura" porque também possui o atributo "numerário" que devia ser "facturas_em_numerario" que passava a ser a associação entre a classe "meios_de_pagamento" e a classe "factura".

Este resultado, tendo em conta o texto apresentado, também teve o resultado esperado, no entanto, não da forma mais correta.

4.4. Museu do Cairo

Neste último exemplo pretende-se representar a história do antigo Egipto e os seus períodos e dinastias. Este último exemplo é talvez aquele que apresenta mais

diferentes formas de representar um texto de um sistema de informação. O texto apresentado foi o seguinte:

"A história do Antigo Egipto, inclui um conjunto de períodos históricos que vão desde 2920 AC (início do Período Dinástico Inicial) até 395 DC (fim do Período Greco-Romano).
Cada período pode incluir um conjunto de dinastias.
Cada dinastia é numerada (1, 2, … até à 30) e corresponde-lhe um determinado número de governantes, cujo nome e tipo de cargo (rei, faraó, regente), a data inicial e final do seu reinado interessa saber.
Para cada dinastia interessa saber uma descrição da arte de cada dinastia (texto curto), bem como quem foi o seu governante mais proeminente."

Neste exemplo inicial, e sendo o objetivo inicial, associar também aos atributos um domínio, houve a necessidade de remover a informação entre parênteses desnecessária, que neste exemplo se encontra na primeira frase. Foi decidido que apenas poderia constar no texto informação importante para a construção do diagrama e como tal a informação a título informativo ao olho humano seria desnecessária. Na frase 2 encontramos ainda uma informação entre parênteses que é a seguinte: "(1, 2, … até à 30)". Nestes casos pensou-se que deveria haver um certo padrão para facilitar a análise do resultado da análise sintática mas que no fundo indicasse o mesmo. Para este tipo de informação criou-se o seguinte padrão "(1 até 30)". Desta forma o texto de sistemas de informação ficou então escrito da seguinte forma:

"A história do Antigo Egipto, inclui um conjunto de períodos históricos que vão desde 2920 AC até 395 DC.
Cada período pode incluir um conjunto de dinastias.
Cada dinastia é numerada (1 até 30) e corresponde-lhe um determinado número de governantes, cujo nome e tipo de cargo (rei, faraó, regente), a data inicial e final do seu reinado interessa saber.
Para cada dinastia interessa saber uma descrição da arte de cada dinastia (texto curto), bem como quem foi o seu governante mais proeminente."

O resultado da análise sintática feita a este texto está apresentado no anexo V. Neste exemplo a maior dificuldade foi representar a classe "Governante" por ser classe e porque é atributo em simultâneo. Houve ainda uma segunda dificuldade que não se conseguiu ultrapassar que foi o facto de na frase 2 existir a palavra "numerada" que se analisarmos o resultado da análise sintática no Anexo V está caracterizada como verbo e neste caso os verbos não podem ser considerados atributos.

A última dificuldade maior foi também na frase 2 em que existem dois atributos diferentes mas camuflados no mesmo nome, ou seja a data inicial e final de reinado. Inicialmente era apenas identificado o atributo "data_inicial_reinado" no entanto pretendia-se também o atributo "data_final_reinado". Foi então necessário criar uma alternativa em que ao encontrar dois ou mais adjetivos entre dois nomes teria que devolver um array de atributos em que o número de atributos a devolver seriam todos idênticos, alterando apenas no adjetivo no meio do atributo. Quando este tipo de atributos é identificado é construído um array de linhas que vai desde o primeiro nome, até ao último nome que fecha esses atributos. Para isso seria necessário verificar se existe mesmo um nome final para fechar o nome de cada atributo ou não. Para confirmar isso é necessário que exista uma preposição, ou seja, depois dos adjetivos existe um "de" então o array de linhas para construir os vários atributos vai terminar apenas no nome. Se não houvesse preposição, construiria por exemplo "data_inicial" construído o array até à palavra inicial e depois nos restantes adjetivos apenas confirmaria se tem dependência direta de nome ou não. Caso tenha dependência direta do nome e o nome tenha dependência do verbo da classe fica então o nome associado à classe.

Desta forma podem ser construídos vários atributos na forma "data_inicial_reinado","data_final_reinado", etc., ou "data_inicial", "data_final", etc..

4.4.1. Avaliação

O resultado final foi o seguinte:

Classe: historia_de_Antigo_Egipto, ID: historia_de_Antigo_Egipto_1330290239.03
Atributo(s):
conjunto_de_periodos_historicos (associacao da classe: periodo): ID =
conjunto_de_periodos_historicos_1330290239.216

Classe: periodo, ID: periodo_1330290239.03
Atributo(s):

conjunto_de_dinastias (associacao da classe: dinastia): ID =
conjunto_de_dinastias_1330290239.569

Classe: dinastia, ID: dinastia_1330290239.061
Atributo(s):
numero_de_governantes (associacao da classe: governante): ID =
numero_de_governantes_1330290239.739
descricao_de_arte: ID = descricao_de_arte_1330290240.209
governante_mais_proeminente (associacao da classe: governante): ID =
governante_mais_proeminente_1330290240.509

Classe: governante, ID: governante_1330290239.096
Atributo(s):
nome: ID = nome_1330290240.599
tipo_de_cargo: ID = tipo_de_cargo_1330290240.629
data_inicial_reinado: ID = data_inicial_reinado_1330290240.669
data_final_reinado: ID = data_final_reinado_1330290240.679

O resultado desta construção de classes, após a criação do XML, anexado no anexo IX, ficou representado no ArgoUML como ilustra a ilustração 24.

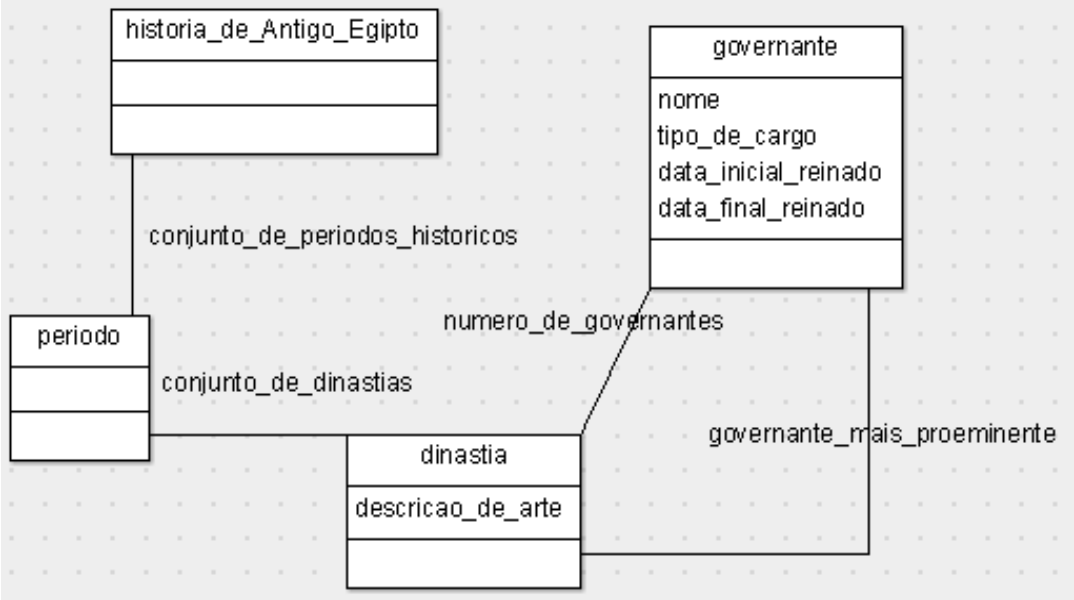

Ilustração 24 - Resultado do quarto exemplo

Neste exemplo conseguiu-se obter o resultado pretendido, no entanto houve alguma confusão na análise das associações, isto porque no resultado final a classe "dinastia" apresenta os atributos "numero_de_governadores" e "governante_mais_proeminente" como associações no entanto ao analisarmos o texto de sistema de informação a ideia que fica é que se pretende saber mesmo o numero de governadores na altura, ou seja se se colocar por exemplo "20" seria porque na altura existiram 20 governadores, no entanto por outro lado será que não se poderia analisar como o conjunto dos governantes, ou seja este atributo iria receber o numero de todos os governantes da altura, isto é, receberia por exemplo [1, 2, 3] e os números 1, 2 e 3, seriam os governadores identificados na classe "governante" com esses mesmos números. No fundo funcionaria como os "BI's" dos governadores. Neste segundo caso o atributo "numero_de_governantes" seria associação, caso fosse o número de governantes que governaram não seria associação.

Este tipo de situações são o tipo de situações que requer uma interpretação humana do texto porque em termos informáticos não se conseguiria dizer à máquina como interpretar o caso, até porque para encontrar as chaves estrangeiras compara-se o nome do atributo com as classes já existentes.

Apesar deste pequeno conflito considera-se que o resultado obtido foi também satisfatório.

Capitulo V – Conclusões

5.1. Conclusões

Nesta dissertação considerou-se que se conseguiram atingir os objetivos pretendidos, que eram, no fundo, a geração de diagramas de classe UML com base em textos de sistemas de informação. Tendo em conta os textos de sistemas informativos apresentados, os resultados obtidos foram satisfatórios, apesar de não se apresentarem num estado perfeito por diversas razões.

Na realidade poderiam apresentar-se resultados mais satisfatórios no que diz respeito à atribuição dos atributos, no entanto existe alguma dificuldade porque humanamente conseguem-se identificar atributos que através do resultado da análise sintática realizada aos textos dificilmente se conseguem identificar. Refiro-me mais especificamente aos atributos que se conseguem identificar através da inteligência humana e que no resultado da análise sintática se apresentam como verbos e não se conseguem identificar, pois ao definir que os verbos também podem ser atributos, muitos mais verbos seriam associados às classes como atributos erradamente.

Para além desta dificuldade na atribuição dos atributos, notou-se também uma grande dificuldade em encontrar as associações porque as associações são atributos que são identificados pela interpretação no texto e neste caso requerem também uma análise por parte do utilizador. A nível funcional existe essa dificuldade ao nível da interpretação do texto de sistemas de informação e como tal teve que se adotar outra forma de identificar essas mesmas associações. Dessa forma optou-se por encontrar as associações comparando o nome do atributo com o nome das restantes classes. Desta forma existe uma grande probabilidade de encontrar as associações, no entanto não pode ser suficiente porque as associações podem não ter necessariamente um nome semelhante ao da classe.

Apesar de os resultados não serem sempre os mais corretos, considerou-se que os objetivos foram atingidos porque esta dissertação pretende facultar um primeiro apoio na construção dos diagramas de classe UML ao utilizador, no entanto não dispensa uma

análise à construção destes mesmos diagramas para possíveis correções e ajustamentos no caso dos atributos que não se conseguem identificar e também nas suas associações em que existe também uma certa dificuldade para as definir.

5.2. Trabalho Futuro

Tal como em todos os tipos de desenvolvimento de software, existe sempre a possibilidade de novos desenvolvimentos e melhoramentos no software desenvolvido, e esta dissertação não foge à regra, e que pode no futuro ser melhorada.

Entre os vários melhoramentos que poderiam existir detetaram-se alguns que poderiam ser bastante importantes na construção destas classes para apresentação de resultados mais satisfatórios e sempre com o intuito de um resultado o mais completo e correto possível.

Ao analisarmos um texto de sistemas de informação, nem sempre nos deparamos na forma em que as frases estão escritas e nestes textos tive a oportunidade de analisar frases na forma ativa. Ao ser analisado o texto de sistemas de informação, agimos intuitivamente para encontrar as classes e atributos, no entanto para encontrar as classes e atributos numa frase, ao nível do desenvolvimento de uma ferramenta para analisar esses mesmos textos, teríamos que ter uma forma de verificar se a forma de identificação das classes e atributos numa frase, seria feita sobre uma frase na forma passiva ou ativa. Por serem frases com construções diferentes, na forma ativa encontramos primeiro as classes e depois do verbo os atributos, no caso das frases passivas, encontramos primeiro os atributos e só depois do verbo, as classes.

No futuro poderia aplicar-se uma solução para resolver este tipo de questão em que, antes de fazer a análise da frase, teria que se descobrir em que forma se encontra a frase e caso se encontrasse na forma ativa, seria feito da forma que foi apresentado nesta dissertação, no caso da forma passiva, iríamos encontrar as classes após o verbo e os atributos antes do verbo. Este resultado não permite distinguir as formas em que se encontra a frase. Teria que se encontrar ou desenvolver uma aplicação que detetasse a forma da frase para saber como encontrar as classes e atributos.

Outra situação que pode ser melhorada no futuro, seria a forma de detetar associações, porque existe a necessidade de se identificar as associações com uma certa análise intuitiva. Encontrou-se uma forma mais "forçada" que seria encontrar o nome da classe no atributo, no entanto nem sempre é viável. Uma forma poderia ser utilizada, seria questionando ao utilizador se determinado atributo deveria ser considerado uma associação entre classes após realizar a pesquisa como foi feita nesta dissertação. Para além de detetar as associações, tal como explicado no ponto 3.6., existem vários tipos de associações e seria importante também definir uma forma de atribuir o tipo de associação mais correta.

Um ultimo caso que se poderia considerar viável para futuro desenvolvimento neste tipo de construção de diagramas de classes UML seria uma forma de calcular os valores das associações. Acabou por não se desenvolver porque poderia necessitar de um segundo texto, em que nesse texto de sistemas de informação se iriam identificar os valores das associações. Essas associações poderiam ser calculadas, talvez, através da forma em que se encontravam os verbos, ou seja, se dissermos que "um filme tem vários actores", as palavras "um" e "vários" são determinantes que se encontram no singular e no plural respetivamente e no resultado da análise sintática tem dependência de "filme" e "actores" respetivamente, conseguiríamos detetar que seria uma ligação de 1 para N.

Para concluir, tive a oportunidade de verificar que após a entrega desta dissertação no dia 10 de Outubro, houve alterações nos resultados das análises sintáticas realizadas aos textos analisados possivelmente por atualizações feitas pelos criadores do VISL o que levou a resultados diferentes. Por esse motivo, decidi continuar a utilizar os resultados descritos nos anexos, no entanto de futuro esta dissertação terá que ser sujeita a adaptações às atualizações do VISL.

Referências

[FundUML] http://docs.kde.org/stable/pt_BR/kdesdk/umbrello/uml-basics.html

[WikiPy] http://pt.wikipedia.org/wiki/Python

[WikiPre] http://pt.wikipedia.org/wiki/Preposi%C3%A7%C3%A3o

[VISL] http://beta.visl.sdu.dk/visl/about/

[WikiEsp] http://pt.wikipedia.org/wiki/Esperanto

[CuPy] http://www.fc.up.pt/pessoas/jsilva/python/curso_python.pdf

[WikiLInt] http://pt.wikipedia.org/wiki/Linguagem_interpretada

[WikiPImp] http://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_imperativa

[ProgOO] http://www.jack.eti.br/www/arquivos/apostilas/java/poo.pdf

[Py] http://www.python.org/about/

[DiagUML] http://www.fernandoamaral.com.br/Default.aspx?Artigo=54

[XML] http://www.w3.org/XML/

[WikiXML] http://pt.wikipedia.org/wiki/XML

[IntroXML] http://www.w3schools.com/xml/xml_whatis.asp

[UMLBas]
http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/

Anexos

Anexo I – Quadro das categorias do VISL para a língua portuguesa

Símbolo		Categoria	Exemplos
N		Nome	árvores **n**(F P) um oitavo **n**(<num> M S)
PROP		Nome próprio	Estados=Unidos **prop**(M P) Dinamarca **prop**(F S)
ADJ		Adjectivo	belas **adj**(F P) terceiros **adj(<num> M P)**
V	VFIN	Verbo finito	fizessem **v-fin**(IMPF 3P SUBJ)
	INF	Infinitivo	fazermos **v-inf**(1P)
	PCP	Particípio	comprados **v-pcp**(M P) [attributive] tem comprado **v-pcp** [verbal]
	GER	Gerúndio	correndo **v-ger**
pron	PERS	Pronome pessoal	mim **pron-pers**(1S PIV) tu **pron-pers(2S NOM)**
	DET	Pronome determinativo	estas **pron-det**(<dem> F P) [demonstrative] muita **pron-det**(<quant> F S) [indefinite] cujos **pron-det**(<rel> M P) [relative] quantos **pron-det**(<interr> M P) [interrogative] minhas **pron-det**(<poss 1P> F P) [possessive]
	SPEC	Pronome independente	isto **pron-indp**(<dem> M S) [demonstrative] algo, nada **pron-indp**(<quant> M S) [indefinite] os=quais **pron-indp**(<rel> M P) [relative] quem **pron-indp**(<interr> M S) [interrogative]
ADV		Advérbio	facilmente, devagar **adv** [modals] aqui, lá **adv** [pronominals] muito, imensamente **adv** [intensifiers] onde, quando, como **adv** [relatives, interrogatives] não, até, já **adv** [operators]
NUM		Numero	duas **num**(F P) 17 **num** (<cif> M/F P)
PRP		Preposição	contra **prp** em=vez=de **prp**

Anexo II – Resultado da análise sintática feita ao primeiro texto utilizando o "Dependency Links"

Pretende-se [Pretende-pretender] <fmc> <mv> **V** UK VFIN @FS-STA #1->0

uma [um] <arti> **DET** F S @>N #2->3

aplicação [aplicação] <act> <sem-r> **N** F S @<SUBJ #3->1

para [para] **PRP** @<ADVL #4->1

armazenar [armazenar] <vH> <mv> **V** INF @ICL-P< #5->4

e [e] KC@NPHR #6->0

consultar [consultar] <vH> <mv> **V** INF @ICL-P< #7->5

os [o] <artd> **DET** M P @>N #8->9

filmes [filme] <sem-w> <cc> **N** M P @<ACC #9->7

. #10->0

</s>

Os [o] <artd> **DET** M P @>N #1->2

filmes [filme] <sem-w> <cc> **N** M P @SUBJ> #2->3

contem [conter] <fmc> <mv> **V** IMP 2S VFIN @FS-COM #3->0

título [título] <f-c> <sem-r> <f-right> **N** M S @<ACC #4->3

, #5->0

género [género] <ac-cat> <meta> <cc> **N** M S @<ACC #6->4

, #7->0

país [país] <Lciv> **N** M S @<ACC #8->4

de [de] <np-close> **PRP** @N< #9->8

origem [origem] <Labs> <f-h> **N** F S @P< #10->9

, #11->0

ano [ano] <dur> <temp> <per> <unit> **N** M S @<ACC #12->4

de [de] <np-close> **PRP** @N< #13->12

realização [realização] <act> **N** F S @P< #14->13

, #15->0

realizador [realizador] <H> <np-close> **N** M S @N<PRED #16->14

, #17->0

actores [actor] <Hprof> **N** M P @<ACC #18->4

e [e] KC@NPHR #19->0

duração [duração] <am> **N** F S @<ACC #20->4

em [em] **PRP** @<ADVL #21->3

minutos [minuto] <dur> <unit> **N** M P @P< #22->21

. #23->0

</s>

Os [o] <artd> **DET** M P @>N #1->2

realizadores [realizador] <H> **N** M P @SUBJ> #2->5

e [e] KC@NPHR #3->0

actores [actor] <Hprof> **N** M P @SUBJ> #4->2

têm [ter] <fmc> <mv> **V** PR 3P IND VFIN @FS-STA #5->0

endereço [endereço] <L> <ac> **N** M S @<ACC #6->5

em [em] <sam-> **PRP** @<ADVL #7->5

a [o] <-sam> <artd> **DET** F S @>N #8->9

Internet [Internet] <top> **PROP** F S @P< #9->7

. #10->0

</s>

O [o] <artd> **DET** M S @>N #1->2

endereço [endereço] <L> <ac> **N** M S @SUBJ> #2->3

contem [conter] <fmc> <mv> **V** IMP 2S VFIN @FS-COM #3->0

link [link] <ac> **N** M S @<ACC #4->3

. #5->0

</s>

O [o] <artd> **DET** M S @>N #1->2

endereço [endereço] <L> <ac> **N** M S @SUBJ> #2->3

contem [conter] <fmc> <mv> **V** IMP 2S VFIN @FS-COM #3->0

uma [um] <arti> **DET** F S @>N #4->5

descrição [descrição] <act> <sem-r> **N** F S @<ACC #5->3

. #6->0

</s>

Anexo III – Resultado da análise sintática feita ao segundo texto utilizando o "Dependency Links"

Pretende-se [Pretende-pretender] <fmc> <mv> **V** UK VFIN @FS-STA #1->0

desenvolver [desenvolver] <mv> **V** INF @ICL-<SUBJ #2->1

uma [um] <arti> **DET** F S @>N #3->4

aplicação [aplicação] <act> <sem-r> **N** F S @<ACC #4->2

que [que] <clb> <clb-fs> <rel> **SPEC** M S @SUBJ> #5->6

disponibilize [disponibilizar] <vH> <mv> <np-close> **V** PR 3S SUBJ VFIN @FS-N< #6->4

artigos [artigo] <sem-r> **N** M P @<ACC #7->6

através [através] **ADV** @<ADVL #8->6

de [de] **PRP** @A< #9->8

leilões [leilão] <occ> **N** M P @P< #10->9

online [online] <jh> <np-close> **ADJ** M P @N< #11->10

. #12->0

</s>

Para [para] **PRP** @ADVL> #1->6

cada [cada] <quant> **DET** M S @>N #2->3

artigo [artigo] <sem-r> **N** M S @P< #3->1

leiloado [leiloar] <vH> <np-close> **V** PCP M S @N< #4->3

deve [dever] <fmc> <aux> **V** PR 3S IND VFIN @FS-STA #5->0

existir [existir] <mv> **V** INF @ICL-AUX< #6->5

a [o] <artd> **DET** F S @>N #7->9

sua [seu] <poss 3S> **DET** F S @>N #8->9

descrição [descrição] <act> <sem-r> **N** F S @<SUBJ #9->5

e [e] KC@NPHR #10->0

categoria [categoria] <ac> **N** F S @<ACC #11->6

(#12->0

música [música] <domain> <sem-l> <cc-r> <np-close> **N** F S @N<PRED #13->11

, #14->0

mobiliário [mobiliário] <nh> <np-close> **ADJ** M S @N<PRED #15->13

, #16->0

decoração [decoração] <act> <ac> <np-close> **N** F S @N<PRED #17->15

, #18->0

etc [etc] **ADV** @<ADVL #19->6

) #20->0

. #21->0

</s>

Para [para] **PRP** @ADVL> #1->4

cada [cada] <quant> **DET** M S @>N #2->3

artigo [artigo] <sem-r> **N** M S @P< #3->1

é [ser] <vK> <fmc> <mv> **V** PR 3S IND VFIN @FS-STA #4->0

também [também] **ADV** @>A #5->6

necessário [necessário] <nh> **ADJ** M S @<SC #6->4

conhecer [conhecer] <vH> <mv> **V** INF @ICL-UTT #7->0

a [o] <artd> **DET** F S @>N #8->9

base [base] <Labs> **N** F S @<ACC #9->7

de [de] <np-close> **PRP** @N< #10->9

licitação [licitação] <occ> **N** F S @P< #11->10

, #12->0

o [o] <artd> **DET** M S @>N #13->15

seu [seu] <si> <poss 3S> **DET** M S @>N #14->15

valor [valor] <am> <ac> <unit> **N** M S @<ACC #15->9

de [de] <np-close> **PRP** @N< #16->15

venda [venda] <act> <inst> <cloH> **N** F S @P< #17->16

e [e] KC@NPHR #18->0

a [o] <artd> **DET** F S @>N #19->20

data [data] <temp> <Ltop> <amount> **N** F S @<ACC #20->9

e [e] KC@NPHR #21->0

a [o] <artd> **DET** F S @>N #22->23

hora [hora] <dur> <temp> <unit> **N** F S @<ACC #23->9

a=partir=de [a=partir=de] <clb> <sam-> **PRP** @ADVL> #24->27

a=qual [o=qual] <clb-fs> <-sam> <rel> **SPEC** F S @P< #25->24

não [não] **ADV** @ADVL> #26->27

são [ser] <vK> <mv> <np-close> **V** PR 3P IND VFIN @FS-N< #27->23

possíveis [possível] <nh> **ADJ** F P @<SC #28->27

mais [muito] <quant> <KOMP> **DET** F P @>N #29->30

licitações [licitação] <occ> **N** F P @<SUBJ #30->27

. #31->0

</s>

Sobre [sobre] **PRP** @ADVL> #1->4

as [o] <artd> **DET** F P @>N #2->3

licitações [licitação] <occ> **N** F P @P< #3->1

é [ser] <vK> <fmc> <mv> **V** PR 3S IND VFIN @FS-STA #4->0

necessário [necessário] <nh> **ADJ** M S @<SC #5->4

conhecer [conhecer] <vH> <mv> **V** INF @ICL-UTT #6->0

o [o] <artd> **DET** M S @>N #7->8

artigo [artigo] <sem-r> **N** M S @<ACC #8->6

, #9->0

a [o] <artd> **DET** F S @>N #10->11

data [data] <temp> <Ltop> <amount> **N** F S @<ACC #11->8

, #12->0

a [o] <artd> **DET** F S @>N #13->14

hora [hora] <dur> <temp> <unit> **N** F S @<ACC #14->8

, #15->0

o [o] <artd> **DET** M S @>N #16->17

valor [valor] <am> <ac> <unit> **N** M S @<ACC #17->8

licitado [licitar] <vH> <np-close> **V** PCP M S @N< #18->17

e [e] KC@NPHR #19->0

a [o] <artd> **DET** F S @>N #20->21

identificação [identificação] <act> **N** F S @<ACC #21->8

de [de] <sam-> <np-close> **PRP** @N< #22->21

o [o] <-sam> <artd> **DET** M S @>N #23->24

licitador [licitador] <Hprof> **N** M S @P< #24->22

. #25->0

</s>

Anexo IV – Resultado da análise sintática feita ao terceiro texto utilizando o "Dependency Links"

pretende-se [pretende-pretender] <fmc> <mv> **V** UK VFIN @FS-STA #1->0

informatizar [informatizar] <mv> **V** INF @ICL-<SUBJ #2->1

uma [um] <arti> **DET** F S @>N #3->4

tesouraria [tesouraria] <Lh> **N** F S @<ACC #4->2

de [de] <np-close> **PRP** @N< #5->4

uma [um] <arti> **DET** F S @>N #6->7

instituição [instituição] <inst> <act> **N** F S @P< #7->5

. #8->0

</s>

a [a] <sam-> **PRP** @PIV> #1->4

a [o] <-sam> <artd> **DET** F S @>N #2->3

tesouraria [tesouraria] <Lh> **N** F S @P< #3->1

dirigem- [dirigir] <hyfen> <vH> <fmc> <mv> **V** PR 3P IND VFIN @FS-STA #4->0

se [se] <refl> **PERS** M/F 3P ACC @<ACC-PASS #5->4

pessoas [pessoa] <H> **N** F P @<SUBJ #6->4

que [que] <clb> <clb-fs> <rel> **SPEC** M S @SUBJ> #7->8

pagam [pagar] <vH> <mv> <np-close> **V** PR 3P IND VFIN @FS-N< #8->6

facturas [factura] <cc-r> **N** F P @<ACC #9->8

. #10->0

</s>

as [o] <artd> **DET** F P @>N #1->2

facturas [factura] <cc-r> **N** F P @SUBJ> #2->3

têm [ter] <fmc> <mv> **V** PR 3P IND VFIN @FS-STA #3->0

um [um] <arti> **DET** M S @>N #4->5

número [número] <ac> <ac-sign> <sem> <ac-cat> **N** M S @<ACC #5->3

único [único] <np-close> **ADJ** M S @N< #6->5

. #7->0

</s>

as [o] <artd> **DET** F P @>N #1->2

pessoas [pessoa] <H> **N** F P @SUBJ> #2->3

podem [poder] <fmc> <aux> **V** PR 3P IND VFIN @FS-STA #3->0

pagar [pagar] <vH> <mv> **V** INF @ICL-AUX< #4->3

facturas [factura] <cc-r> **N** F P @<ACC #5->4

em [em] **PRP** @<ADVL #6->4

numerário [numerário] <mon> **N** M S @P< #7->6

, #8->0

ou [ou] KC@NPHR #9->0

por [por] **PRP** @<ADVL #10->4

ponto [ponto] <Labs> <temp> <Lh> <sem-c> <unit> **N** M S @P< #11->10

de [de] <np-close> **PRP** @N< #12->11

serviço [serviço] <act-d> **N** M S @P< #13->12

, #14->0

ou [ou] KC@NPHR #15->0

cheque [cheque] <cc-r> **N** M S @<ACC @P< #16->5

. #17->0

</s>

Os [o] <artd> **DET** M P @>N #1->2

meios [meio] <Labs> <tool> <amount> **N** M P @SUBJ> #2->5

de [de] <np-close> **PRP** @N< #3->2

pagamento [pagamento] <act> **N** M S @P< #4->3

podem [poder] <fmc> <aux> **V** PR 3P IND VFIN @FS-STA #5->0

ser [ser] <vK> <mv> **V** INF @ICL-AUX< #6->5

por [por] **PRP** @<SC #7->6

numerário [numerário] <mon> **N** M S @P< #8->7

, #9->0

ponto [ponto] <Labs> <temp> <Lh> <sem-c> <unit> **N** M S @<SC #10->7

de [de] <np-close> **PRP** @N< #11->10

serviço [serviço] <act-d> **N** M S @P< #12->11

ou [ou] KC@NPHR #13->0

cheque [cheque] <cc-r> **N** M S @P< #14->12

. #15->0

</s>

Anexo V – Resultado da análise sintática feita ao quarto texto utilizando o "Dependency Links"

A [o] <artd> **DET** F S @>N #1->2

história [história] <per> <domain> <sem-r> **N** F S @SUBJ> #2->7

de [de] <sam-> <np-close> **PRP** @N< #3->2

o [o] <-sam> <artd> **DET** M S @>N #4->5

Antigo=Egipto [Antigo=Egipto] <inst> **PROP** M S @P< #5->3

, #6->0

inclui [incluir] <fmc> <mv> **V** PR 3S IND VFIN @FS-STA #7->0

um [um] <arti> **DET** M S @>N #8->9

conjunto [conjunto] <coll> <HH> <amount> **N** M S @<ACC #9->7

de [de] <np-close> **PRP** @N< #10->9

períodos [período] <per> **N** M P @P< #11->10

históricos [histórico] <np-close> **ADJ** M P @N< #12->11

que [que] <clb> <clb-fs> <rel> **SPEC** M S @SUBJ> #13->14

vão [ir] <mv> <np-close> **V** PR 3P IND VFIN @FS-N< #14->11

desde [desde] **PRP** @<ADVL #15->14

2920 [2920] <card> <date> <year> **NUM** M P @>N #16->17

AC [AC] <inst> **PROP** M S @P< #17->15

até [até] **PRP** @<ADVL #18->14

395 [395] <card> **NUM** M/F P @>N #19->20

DC [DC] <party> **PROP** M/F P @P< #20->18

. #21->0

</s>

Cada [cada] <quant> **DET** M S @>N #1->2

período [período] <per> **N** M S @SUBJ> #2->3

pode [poder] <fmc> <aux> **V** PR 3S IND VFIN @FS-STA #3->0

incluir [incluir] <mv> **V** INF @ICL-AUX< #4->3

um [um] <arti> **DET** M S @>N #5->6

conjunto [conjunto] <coll> <HH> <amount> **N** M S @<ACC #6->4

de [de] <np-close> **PRP** @N< #7->6

dinastias [dinastia] <HH> **N** F P @P< #8->7

. #9->0

</s>

Cada [cada] <quant> **DET** F S @>N #1->2

dinastia [dinastia] <HH> **N** F S @SUBJ> #2->3

é [ser] <fmc> <aux> **V** PR 3S IND VFIN @FS-STA #3->0

numerada [numerar] <vH> <mv> **V** PCP F S @ICL-AUX< #4->3

(#5->0

1 [1] <card> **NUM** M/F S @>N #6->12

até [até] <np-close> **PRP** @N< #7->2

30 [30] <card> **NUM** M/F P @P< #8->7

) #9->0

e [e] **KC** @CO #10->8

corresponde- [corresponder] <hyfen> <fmc> <mv> **V** PR 3S IND VFIN @FS-STA #11->3

lhe [ele] **PERS** M/F 3S DAT @<DAT #12->11

um [um] <arti> **DET** M S @>N #13->15

determinado [determinar] <vH> **V** PCP M S @>N #14->15

número [número] <ac> <ac-sign> <sem> <ac-cat> **N** M S @<ACC #15->11

de [de] <np-close> **PRP** @N< #16->15

governantes [governante] <Hprof> **N** M/F P @P< #17->16

, #18->0

cujo [cujo] <clb> <clb-fs> <rel> **DET** M S @>N #19->20

nome [nome] <percep-f> <ac-cat> **N** M S @SUBJ> #20->42

e [e] **KC** @CO #21->20

tipo [tipo] <meta> <H> <ac-sign> **N** M S @SUBJ> #22->20

de [de] <np-close> **PRP** @N< #23->22

cargo [cargo] <ac> **N** M S @P< #24->23

(#25->0

rei [rei] <Htit> <np-long> **N** M S @N<PRED #26->22

, #27->0

faraó [faraó] <Htit> <np-close> **N** M S @N<PRED #28->26

, #29->0

regente [regente] <Hprof> <np-close> **N** M/F S @N<PRED #30->28

) #31->0

, #32->0

a [o] <artd> **DET** F S @>N #33->34

data [data] <temp> <Ltop> <amount> **N** F S @SUBJ> #34->22

inicial [inicial] <nh> <np-close> **ADJ** F S @N< #35->34

e [e] **KC** @CO #36->37

final [final] <nh> <np-close> **ADJ** M/F S @N< #37->34

de [de] <sam-> <np-close> **PRP** @N< #38->34

o [o] <-sam> <artd> **DET** M S @>N #39->41

seu [seu] <poss 3S> **DET** M S @>N #40->41

reinado [reinado] <per> **N** M S @P< #41->38

interessa [interessar] <mv> <np-close> **V** PR 3S IND VFIN @FS-N< #42->17

saber [saber] <mv> **V** INF @ICL-<ACC #43->42

. #44->0

</s>

Para [para] **PRP** @ADVL> #1->4

cada [cada] <quant> **DET** F S @>N #2->3

dinastia [dinastia] <HH> **N** F S @P< #3->1

interessa [interessar] <fmc> <mv> **V** PR 3S IND VFIN @FS-STA #4->0

saber [saber] <mv> **V** INF @ICL-<ACC #5->4

uma [um] <arti> **DET** F S @>N #6->7

descrição [descrição] <act> <sem-r> **N** F S @<ACC #7->5

de [de] <sam-> <np-close> **PRP** @N< #8->7

a [o] <-sam> <artd> **DET** F S @>N #9->10

arte [arte] <domain> **N** F S @P< #10->8

de [de] <np-close> **PRP** @N< #11->10

cada [cada] <quant> **DET** F S @>N #12->13

dinastia [dinastia] <HH> **N** F S @P< #13->11

(#14->0

texto [texto] <sem-r> <np-close> **N** M S @N<PRED #15->13

curto [curto] <nh> <np-close> **ADJ** M S @N< #16->15

) #17->0

, #18->0

bem=como [bem=como] <clb> <rel> **ADV** @ADVL #19->21

quem [quem] <clb> <clb-fs> <interr> **SPEC** M/F S/P @SC> [quem] <clb> <rel> SPEC M/F S/P @SC> #20->21

foi [ser] <mv> <np-close> **V** PS 3S IND VFIN @FS-N< #21->15

o [o] <artd> **DET** M S @>N #22->24

seu [seu] <poss 3S> **DET** M S @>N #23->24

governante [governante] <Hprof> **N** M S @<SUBJ #24->21

mais [mais] <quant> <KOMP> **ADV** @>A #25->26

proeminente [proeminente] <nh> <np-close> **ADJ** M/F S @N< #26->24

. #27->0

</s>

Anexo VI – XML resultante da análise do primeiro texto de sistemas de informação

```
<?xml version="1.0" ?>
<XMI timestamp="Mon Apr  2 21:12:22 2012" xmi.version="1.2"
xmlns:UML="org.omg.xmi.namespace.UML">
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>
        ArgoUML (using Netbeans XMI Writer version 1.0)
      </XMI.exporter>
      <XMI.exporterVersion>
        0.32.2(6) revised on $Date: 2010-01-11 22:20:14 +0100 (Mon, 11 Jan
2010) $
      </XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.4"/>
  </XMI.header>
  <XMI.content>
    <UML:Model isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="modeloSemTitulo" xmi.id="1333397542.949">
      <UML:Namespace.ownedElement>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="filme" visibility="public"
xmi.id="filme_1333397542.593">
          <UML:Classifier.feature>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="titulo" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="titulo_1333397542.647">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_titulo_1333397542.978">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_titulo_1333397542.982"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="genero" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="genero_1333397542.667">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_genero_1333397542.996">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_genero_1333397542.999"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="pais_de_origem" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="pais_de_origem_1333397542.691">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_pais_de_origem_1333397543.013">
```

```
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_pais_de_origem_1333397543.017"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="ano_de_realizacao" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="ano_de_realizacao_1333397542.715">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_ano_de_realizacao_1333397543.031">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_ano_de_realizacao_1333397543.035"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="duracao_em_minutos" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="duracao_em_minutos_1333397542.807">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_duracao_em_minutos_1333397543.12">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_duracao_em_minutos_1333397543.123"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
          </UML:Classifier.feature>
        </UML:Class>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="realizador"
xmi.id="realizador_realizador_1333397542.906">
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397543.049_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="filme_1333397542.593"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397543.061_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="realizador_1333397542.602"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="actores" xmi.id="actores_actor_1333397542.908">
          <UML:Association.connection>
```

```
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397543.083_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="filme_1333397542.593"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397543.096_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="actor_1333397542.606"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="realizador" visibility="public"
xmi.id="realizador_1333397542.602">
          <UML:Classifier.feature/>
        </UML:Class>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="endereco_em_Internet"
xmi.id="endereco_em_Internet_endereco_1333397542.912">
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397543.146_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="realizador_1333397542.602"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397543.159_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="endereco_1333397542.612"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="actor" visibility="public"
xmi.id="actor_1333397542.606">
          <UML:Classifier.feature/>
        </UML:Class>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="endereco_em_Internet"
xmi.id="endereco_em_Internet_endereco_1333397542.914">
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397543.191_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="actor_1333397542.606"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
```

```
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397543.203_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="endereco_1333397542.612"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="endereco" visibility="public"
xmi.id="endereco_1333397542.612">
          <UML:Classifier.feature>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="link" ownerScope="instance" targetScope="instance" visibility="public"
xmi.id="link_1333397542.866">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity xmi.id="multiplicity_link_1333397543.237">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_link_1333397543.241"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="descricao" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="descricao_1333397542.9">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_descricao_1333397543.255">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_descricao_1333397543.258"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
          </UML:Classifier.feature>
        </UML:Class>
      </UML:Namespace.ownedElement>
    </UML:Model>
  </XMI.content>
</XMI>
```

Anexo VII – XML resultante da análise do segundo texto de sistemas de informação

```
<?xml version="1.0" ?>
<XMI timestamp="Mon Apr  2 21:15:26 2012" xmi.version="1.2"
xmlns:UML="org.omg.xmi.namespace.UML">
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>
        ArgoUML (using Netbeans XMI Writer version 1.0)
      </XMI.exporter>
      <XMI.exporterVersion>
        0.32.2(6) revised on $Date: 2010-01-11 22:20:14 +0100 (Mon, 11 Jan
2010) $
      </XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.4"/>
  </XMI.header>
  <XMI.content>
    <UML:Model isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="modeloSemTitulo" xmi.id="1333397726.617">
      <UML:Namespace.ownedElement>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="artigo" visibility="public"
xmi.id="artigo_1333397726.006">
          <UML:Classifier.feature>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="descricao" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="descricao_1333397726.159">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_descricao_1333397726.65">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_descricao_1333397726.654"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="categoria" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="categoria_1333397726.199">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_categoria_1333397726.67">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_categoria_1333397726.673"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="valor_de_venda" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="valor_de_venda_1333397726.367">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_valor_de_venda_1333397726.726">
```

```
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_valor_de_venda_1333397726.729"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="data" ownerScope="instance" targetScope="instance" visibility="public"
xmi.id="data_1333397726.394">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity xmi.id="multiplicity_data_1333397726.746">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_data_1333397726.749"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="hora_a_partir_de_o_qual_nao_ser_possiveis_licitaces"
ownerScope="instance" targetScope="instance" visibility="public"
xmi.id="hora_a_partir_de_o_qual_nao_ser_possiveis_licitaces_1333397726.441">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_hora_a_partir_de_o_qual_nao_ser_possiveis_licitaces_1333
397726.765">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_hora_a_partir_de_o_qual_nao_ser_possiveis_licitaces
_1333397726.769"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
          </UML:Classifier.feature>
        </UML:Class>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="base_de_licitacao"
xmi.id="base_de_licitacao_licitacao_1333397726.579">
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397726.688_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="artigo_1333397726.006"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397726.701_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="licitacao_1333397726.065"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
```

```
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="licitacao" visibility="public"
xmi.id="licitacao_1333397726.065">
          <UML:Classifier.feature>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="data" ownerScope="instance" targetScope="instance" visibility="public"
xmi.id="data_1333397726.498">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity xmi.id="multiplicity_data_1333397726.835">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_data_1333397726.839"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="hora" ownerScope="instance" targetScope="instance" visibility="public"
xmi.id="hora_1333397726.51">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity xmi.id="multiplicity_hora_1333397726.855">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_hora_1333397726.859"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="valor_licitado" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="valor_licitado_1333397726.525">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_valor_licitado_1333397726.875">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_valor_licitado_1333397726.879"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="identificacao_de_licitador" ownerScope="instance"
targetScope="instance" visibility="public"
xmi.id="identificacao_de_licitador_1333397726.575">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_identificacao_de_licitador_1333397726.895">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_identificacao_de_licitador_1333397726.899"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
          </UML:Classifier.feature>
        </UML:Class>
```

```
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="artigo" xmi.id="artigo_artigo_1333397726.584">
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397726.795_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="licitacao_1333397726.065"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397726.81_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="artigo_1333397726.006"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
      </UML:Namespace.ownedElement>
    </UML:Model>
  </XMI.content>
</XMI>
```

Anexo VIII – XML resultante da análise do terceiro texto de sistemas de informação

```
<?xml version="1.0" ?>
<XMI timestamp="Mon Apr  2 21:17:32 2012" xmi.version="1.2"
xmlns:UML="org.omg.xmi.namespace.UML">
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>
        ArgoUML (using Netbeans XMI Writer version 1.0)
      </XMI.exporter>
      <XMI.exporterVersion>
        0.32.2(6) revised on $Date: 2010-01-11 22:20:14 +0100 (Mon, 11 Jan
2010) $
      </XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.4"/>
  </XMI.header>
  <XMI.content>
    <UML:Model isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="modeloSemTitulo" xmi.id="1333397852.191">
      <UML:Namespace.ownedElement>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="tesouraria" visibility="public"
xmi.id="tesouraria_1333397851.682">
          <UML:Classifier.feature/>
        </UML:Class>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="pessoas"
xmi.id="pessoas_pessoa_1333397852.156">
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397852.221_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="tesouraria_1333397851.682"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397852.235_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="pessoa_1333397851.687"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="pessoa" visibility="public"
xmi.id="pessoa_1333397851.687">
          <UML:Classifier.feature>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="ponto_de_servico" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="ponto_de_servico_1333397851.977">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_ponto_de_servico_1333397852.309">
                  <UML:Multiplicity.range>
```

```
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_ponto_de_servico_1333397852.313"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="cheque" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="cheque_1333397852.004">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity xmi.id="multiplicity_cheque_1333397852.33">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_cheque_1333397852.333"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
          </UML:Classifier.feature>
        </UML:Class>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="facturas_em_numerario"
xmi.id="facturas_em_numerario_factura_1333397852.159">
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397852.269_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="pessoa_1333397851.687"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397852.283_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="factura_1333397851.694"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="factura" visibility="public"
xmi.id="factura_1333397851.694">
          <UML:Classifier.feature>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="numero_unico" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="numero_unico_1333397852.041">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_numero_unico_1333397852.36">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_numero_unico_1333397852.364"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
          </UML:Classifier.feature>
```

```
        </UML:Class>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="meios_de_pagamento"
visibility="public" xmi.id="meios_de_pagamento_1333397851.716">
          <UML:Classifier.feature>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="numerario" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="numerario_1333397852.113">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_numerario_1333397852.392">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_numerario_1333397852.396"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="ponto_de_servico" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="ponto_de_servico_1333397852.131">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_ponto_de_servico_1333397852.412">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_ponto_de_servico_1333397852.416"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="cheque" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="cheque_1333397852.154">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_cheque_1333397852.432">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_cheque_1333397852.436"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
          </UML:Classifier.feature>
        </UML:Class>
      </UML:Namespace.ownedElement>
    </UML:Model>
  </XMI.content>
</XMI>
```

Anexo IX – XML resultante da análise do quarto texto de sistemas de informação

```
<?xml version="1.0" ?>
<XMI timestamp="Mon Apr  2 21:19:52 2012" xmi.version="1.2"
xmlns:UML="org.omg.xmi.namespace.UML">
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>
        ArgoUML (using Netbeans XMI Writer version 1.0)
      </XMI.exporter>
      <XMI.exporterVersion>
        0.32.2(6) revised on $Date: 2010-01-11 22:20:14 +0100 (Mon, 11 Jan
2010) $
      </XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.4"/>
  </XMI.header>
  <XMI.content>
    <UML:Model isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="modeloSemTitulo" xmi.id="1333397992.188">
      <UML:Namespace.ownedElement>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="historia_de_Antigo_Egipto"
visibility="public" xmi.id="historia_de_Antigo_Egipto_1333397990.712">
          <UML:Classifier.feature/>
        </UML:Class>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="conjunto_de_periodos_historicos"
xmi.id="conjunto_de_periodos_historicos_periodo_1333397992.134">
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397992.217_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class
xmi.idref="historia_de_Antigo_Egipto_1333397990.712"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397992.231_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="periodo_1333397990.719"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="periodo" visibility="public"
xmi.id="periodo_1333397990.719">
          <UML:Classifier.feature/>
        </UML:Class>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="conjunto_de_dinastias"
xmi.id="conjunto_de_dinastias_dinastia_1333397992.136">
          <UML:Association.connection>
```

```
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397992.262_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="periodo_1333397990.719"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397992.275_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="dinastia_1333397990.732"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="dinastia" visibility="public"
xmi.id="dinastia_1333397990.732">
          <UML:Classifier.feature>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="descricao_de_arte" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="descricao_de_arte_1333397991.698">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_descricao_de_arte_1333397992.342">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_descricao_de_arte_1333397992.346"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
          </UML:Classifier.feature>
        </UML:Class>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="numero_de_governantes"
xmi.id="numero_de_governantes_governante_1333397992.139">
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397992.307_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="dinastia_1333397990.732"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397992.319_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="governante_1333397990.762"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
isSpecification="false" name="governante_mais_proeminente"
xmi.id="governante_mais_proeminente_governante_1333397992.143">
```

```
          <UML:Association.connection>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397992.358_inicio">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="dinastia_1333397990.732"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd aggregation="none" changeability="changeable"
isNavigable="true" isSpecification="false" ordering="unordered"
targetScope="instance" visibility="public" xmi.id="1333397992.371_fim">
              <UML:AssociationEnd.participant>
                <UML:Class xmi.idref="governante_1333397990.762"/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:Class isAbstract="false" isActive="false" isLeaf="false"
isRoot="false" isSpecification="false" name="governante" visibility="public"
xmi.id="governante_1333397990.762">
          <UML:Classifier.feature>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="nome" ownerScope="instance" targetScope="instance" visibility="public"
xmi.id="nome_1333397992.052">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity xmi.id="multiplicity_nome_1333397992.404">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_nome_1333397992.407"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="tipo_de_cargo" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="tipo_de_cargo_1333397992.079">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_tipo_de_cargo_1333397992.421">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_tipo_de_cargo_1333397992.424"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="data_inicial_reinado" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="data_inicial_reinado_1333397992.118">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_data_inicial_reinado_1333397992.438">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_data_inicial_reinado_1333397992.441"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
```

```
            </UML:Attribute>
            <UML:Attribute changeability="changeable" isSpecification="false"
name="data_final_reinado" ownerScope="instance" targetScope="instance"
visibility="public" xmi.id="data_final_reinado_1333397992.119">
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity
xmi.id="multiplicity_data_final_reinado_1333397992.455">
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange lower="1" upper="1"
xmi.id="multiplicityRange_data_final_reinado_1333397992.459"/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
            </UML:Attribute>
          </UML:Classifier.feature>
        </UML:Class>
      </UML:Namespace.ownedElement>
    </UML:Model>
  </XMI.content>
</XMI>
```

Anexo X – Exemplo de código XML a importar para o ArgoUML

```
<?xml version = '1.0' encoding = 'UTF-8' ?>
<XMI xmi.version = '1.2' xmlns:UML = 'org.omg.xmi.namespace.UML' timestamp =
'Thu May 24 17:16:15 BST 2012'>
  <XMI.header>    <XMI.documentation>
      <XMI.exporter>ArgoUML (using Netbeans XMI Writer version
1.0)</XMI.exporter>
      <XMI.exporterVersion>0.32.2(6) revised on $Date: 2010-01-11 22:20:14
+0100 (Mon, 11 Jan 2010) $ </XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.4"/></XMI.header>
  <XMI.content>
    <UML:Model xmi.id = '10-20-2-114-57272062:1377fa0c0a6:-
8000:0000000000000865'
      name = 'modeloSemTitulo' isSpecification = 'false' isRoot = 'false'
isLeaf = 'false'
      isAbstract = 'false'>
      <UML:Namespace.ownedElement>
        <UML:Class xmi.id = '10-20-2-114-57272062:1377fa0c0a6:-
8000:0000000000000866'
          name = 'Pessoa' visibility = 'public' isSpecification = 'false'
isRoot = 'false'
          isLeaf = 'false' isAbstract = 'false' isActive = 'false'>
          <UML:Classifier.feature>
            <UML:Attribute xmi.id = '10-20-2-114-57272062:1377fa0c0a6:-
8000:0000000000000867'
              name = 'nome' visibility = 'public' isSpecification = 'false'
ownerScope = 'instance'
              changeability = 'changeable' targetScope = 'instance'>
              <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity xmi.id = '10-20-2-114-
57272062:1377fa0c0a6:-8000:0000000000000868'>
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange xmi.id = '10-20-2-114-
57272062:1377fa0c0a6:-8000:0000000000000869'
                      lower = '1' upper = '1'/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:StructuralFeature.multiplicity>
              <UML:StructuralFeature.type>
                <UML:DataType href =
'http://argouml.org/profiles/uml14/default-uml14.xmi#-84-17--56-5-
43645a83:11466542d86:-8000:000000000000087E'/>
              </UML:StructuralFeature.type>
            </UML:Attribute>
          </UML:Classifier.feature>
        </UML:Class>
      </UML:Namespace.ownedElement>
    </UML:Model>
  </XMI.content>
</XMI>
```