FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Processamento de Linguagem Natural para Produtos de Seguros

Mário Jorge Silveira Pereira

PREPARAÇÃO DA DISSERTAÇÃO

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof. João Pascoal Faria

28 de Julho de 2014

Processamento de Linguagem Natural para Produtos de Seguros

Mário Jorge Silveira Pereira

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Prof. Dr. Nuno Flores
Arguente: Prof. Dr. Alberto Silva
Vogal: Prof. Dr. João Pascoal Faria
28 de Julho de 2014

Resumo

A Gestão do Ciclo de vida de um produto (*Product Lifecycle management*) é uma metodologia cuja existência é suportada pelo desempenho dos recursos humanos. As pessoas expressam-se da melhor forma através da linguagem natural. Neste sentido e focando-nos, é necessário que o produto seja receptivo e pronto para a entrada de dados inseridos por pessoas através de linguagem natural. A criação de um produto de seguro em XML a partir da introdução da sua descrição usando Linguagem Natural, assume-se como a fase de criação de produto, e a primeira no âmbito da Gestão do Ciclo de Vida do Produto.

Esta aplicação vem da necessidade das pessoas terem uma interação intuitiva e eficaz para com o sistema o que proporciona vantagens em termos de negócio. O objectivo desta dissertação foca-se em conseguir transmitir, armazenar e formatar informação introduzida em linguagem natural de maneira a conseguir obter um produto de seguro viável. Este é visualizável através de um formulário gerado a partir de um XML que está em concordância com os standards ACORD normalmente associados a produtos de seguro.

Todo o projecto é desenvolvido usando tecnologias Java (tirando o formulário final gerado em HTML) direcionadas aos diferentes aspectos do tratamento de dados, como por exemplo, a interpretação de linguagem natural ou o tratamento de XML.

Abstract

The PLM (Product Lifecycle Management) is a methodology whose existence is supported by the performance of human resources. People express themselves the best way through natural language. In this sense and in the focusing, it is necessary that the product is receptive and ready to input data entered by people using natural language. The creation of an insurance product to XML from entering its description using Natural Language, is assumed as the design phase of product, and the first under the Management Life Cycle Product.

This application comes from people's necessity of having an intuitive and useful for interaction with the system which provides benefits in terms of business. The purpose of this dissertation focuses on achieving transmit, store and format information entered in natural language in order to achieve a product viewable insurance through a form generated from an XML which is in agreement with ACORD standards normally associated with insurance products.

The entire project is developed in Java technologies (taking the final form generated HTML) directed to different aspects of data processing, such as the interpretation of natural language or the treatment of XML.

Agradecimentos

Agradeço ao meu orientador Prof. João Pascoal Faria por todas as indicações e ajudas dadas no decorrer do desenvolvimento da tese. Por causa da sua ajuda consegui fazer uma tese melhor.

Além do meu orientador quero agradecer à minha Mãe por me perguntar todos os dias como vai a tese e dar sempre os seus conselhos para que as coisas corram melhor.

Finalmente, quero agradecer ao João Anes, João Carvalho, João Afonso, Jorge Silva, Francisco Bernardo, Joel Ramos e Wilson Pimentel, por todas as dicas, ajudas e encorajamento dadas durante os últimos meses.

Mário Pereira

"When you want to succeed as bad as you want to breathe,
then you'll be successful"

Eric Thomas

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

PLN	Processamento de Linguagem Natural
PLM	Product Lifecycle Management
XML	eXtensible Markup Language
NLTK	Natural Language Toolkit
RSL-IL	Requisit Specification Language - Intermediate Languagem
RSL-PL	Requisite Specification Language - Pattern Language
CRUD	Create, Read, Update, Delete

Capítulo 1

Introdução

Actualmente, devido à cada vez maior competitividade entre empresas, a necessidade de colocar produtos no mercado à medida do consumidor e o mais rapidamente possível torna-se cada vez maior. Para fazer frente à cada vez maior luta por cotas de mercado nas diferentes áreas, as metodologias de desenvolvimento de produto evoluiram rapidamente. Neste momento, a abordagem Product Lifecycle Management(PLM) é apresentada como uma das mais capazes e úteis a fornecer uma curta e rápida introdução dos produtos no mercado e um rápido desenvolvimento. Desta forma, esta abordagem(PLM) é dada como uma das melhores abordagens tecnológicas a fornecer às empresas maneiras mais rápidas, fáceis e inteligentes de ultrapassar obstáculos que surgem desde o desenho/criação de um produto até este ser retirado do mercado, isto feito através de novas maneiras de planear, organizar, gerir, analisar e distribuir novos produtos ou serviços mais rapidamente, melhor e com menos custo. [Min05] Surge portanto, neste contexto, a oportunidade de desenvolvimento de diferentes produtos de software capazes de agilizar e suportar as diferentes fases de desenvolvimento do produto para as diferentes indústrias.

Neste sentido, a pertinência da criação de software que permita e suporte a gestão do ciclo de vida de um produto no sector financeiro, torna-se maior. Neste momento, algumas companhias financeiras utilizam parcialmente conceitos PLM. No entanto também é considerado que é possível melhorar ainda mais o seu desempenho. Frequentemente, não existem frameworks para guiar, monitorizar ou coordenar projectos em desenvolvimento, o que leva a que a concepção de um produto não seja devidamente estruturada e coordenada, e o processo seja custoso, causando problemas ao nível do processo de negócio. O principal problema deriva do facto de o desenvolvimento de um produto envolver vários departamentos e necessitar do *input* de várias equipas e/ou indivíduos que não estão na maior parte das vezes coordenados e cientes do fluxo do trabalho no seu todo. [Ada]

Tratando de um caso em concreto, e olhando para um produto financeiro especificamente como uma apólice de seguro, numa primeira fase de criação do produto, e se estivermos inseridos numa cultura PLM, o objectivo será que após a sua criação o mesmo esteja disponível para

ser processado por diferentes pessoas e/ou departamentos. Naturalmente, para a criação do produto no sistema, o criador do produto necessita de o descrever através da sua própria linguagem - através de linguagem natural. Portanto, a partir de um texto não estruturado, é necessário estabelecer um produto bem definido, estruturado e parametrizado, que seja passível de ser armazenado, consultado e trasmitido.

Esta interpretação da linguagem natural é feita através de uma abordagem em que são estabelecidos padrões linguísticos tendo em conta os produtos financeiros a inserir e a partir do momento em que é obtida informação formalmente estruturada, esta já é passível de ser posteriormente passada a XML e partir daí construir *WebServices* especificados à medida. No limite existirá outra forma de obter *inputs* , mas será um utilizador denominado de cliente que receberá a informção inserida a partir de linguagem natural e que a partir de um formulário poderá inserir informações que dizem respeito à sua interacção com o sistema.

No que diz respeito aos pontos a cobrir no decorrer deste capítulo introdutório, será feito um enquadramento mais específico acerca desta dissertação. Será especificada a sua motivação, os seus objectivos e por fim a estrutura da dissertação em si.

1.1 Contexto/Enquadramento

Esta dissertação é supervisionada pelo Prof. Dr. João Pascoal Faria e vai ser desenvolvida como tese de mestrado proposta pela Altran Portugal. A Altran Portugal é uma empresa de consultoria em projectos de IT, actualmente com mais de 500 colaboradores. Está presente em vários sectores de actividade que vão desde o sector financeiro até sectores como o da Administração Pública. Da parte da empresa, a supervisão é efectuada pelo Eng. Luís Alves. Esta dissertação tem como objectivo fornecer à Altran Portugal uma versão *beta* para verificar se o conceito em si tem o potencial necessário para evoluir para uma aplicação mais complexa e passível de ser utilizada em contextos mais práticos e reais.

Hoje em dia, tendo em conta o quão competitiva e agressiva pode ser a economia global , o sucesso de um produto depende em muito da capacidade da empresa para criar produtos que cativem os consumidores. A criação de produtos que safistaçam plenamente o utilizador está geralmente associada a factores como o tempo, a funcionalidade, performance, imagem e preço. [GCA+10] Dado serem indicadores constantemente em mudança, as empresas têm que utilizar métodos que permitam adaptarem-se rapidamente às diversas mudanças. Actualmente a direcção predominante no mercado de aplicações de software converge no sentido de fornecer soluções tecnológicas que suportem a metodologia PLM. Esta, fornece aos clientes, criadores, frabricantes e fornecedores os meios que permitem gerir a actividade de negócio de maneira mais eficiente e rápida. [Min05] As respostas às necessidades do mercado são feitas através de uma estratégia de negócio que usa uma framework computacional que permite a captura, representação e reutilização de conhecimento acerca de um determinado produto. [AD05]

Fundamentalmmente, PLM é uma estratégia de negócio que aplica um conjunto específico de soluções de modo a suporta uma criação, gestão, disseminação e uso da informação do produto

de forma colaborativa através de toda a empresa, desde a concepção até ao fim de vida do produto - integrando pessoas, processos, informação e sistemas de negócio. [GCA+10] Os sistemas PLM suportam a gestão de um portfólio de produtos, processos e serviços desde a concepção inicial, passando por todas as fazes como desing, fabrico e lançamento e acabando no fim de vida do produto. Existe uma coordenação e colaboração em termos de produto, projecto e processos que atravessa a cadeira de valor do produto dentro de uma empresa. [Min05]

No que diz respeito ao conteúdo presente nesta secção, será feita uma abordagem sobre a situação actual em termos de aplicação do PLM através da utilização de software que suporta este processo e as implicações que este processo traz sobre a cultura de uma empresa.Por fim, é explicado como é que este processo pode gerar valor para uma empresa e justificar o seu uso.

1.1.1 Aplicação de PLM

Relativamente à utilização deste processo é preciso delinear quais as fases a serem implementadas : Concepção, Design , Realização e Serviço. Nesta dissertação, o principal foco vai ser sobre a fase de Concepção, visto ser esta a fase que vai ser implementada durante o processo de desenvolvimento.

A metodologia PLM é geralmente vista como um grande *bundle* de complexas ferramentas de software e aplicações que suportem o Design e manufactura de produtos em várias fases. No entanto, esta conceptualização esquece-se que o conceito base do PLM é a gestão de conhecimento. As tecnologias que compõe o PLM permitem a criação de conhecimento, transformação e partilha ao longo de todo o ciclo de vida. [AD05] Este conceito é fundamental para a aplicação a desenvolver no âmbito desta dissertação, visto que as principais funcionalidades a conceber têm como objectivo permitir ao utilizador inserir e estruturar conhecimento acerca de apólices de seguro e pemitir a consulta dos seus dados através de *WebServices*. Para que exista gestão de conhecimento, em primeiro lugar e logicamente é necessário inseri-lo no sistema, ou seja, de alguma forma, criá-lo. Feita esta pequena análise dos aspectos mais conceptuais do conceito PLM, existe também um ponto de vista que analise o PLM do ponto de vista cultural, explicitanto este processo como uma cultura empresarial.

Para qualquer empresa, a chave para o sucesso a longo prazo é consistemente fazer determinadas coisas melhor que a sua competição e estar alinhada de acordo com uma estratégia corporativa. Grandes estratégias de produção são aquelas que desenvolvem uma cultura dinâmica e sustentável na organização, isto porque a cultura é algo intrínseco a uma organização e não é facilmente copiável. [AD05]

O primeiro passo para estabelecer a cultura PLM seria entender e analisar a maneira como a empresa funciona, a sua estrutura organizacional, papéis e responsabilidades dentro da organização. Cada parte do sistema operacional PLM deve ser definida de modo a saber-se quem contribuiu exactamente, qual o seu contributo, como a informação é partilhada e os responsáveis por cada parte do sistema. Não é necessário que todos os sistemas de operações estejam integrados numa única ferramenta de software, e normalmente, para pequenas e médias empresas isso não acontece ao contrário do verificado nas restantes. [GCA+10]

A aplicação do PLM em médias e grandes empresas geralmente vem com a oferta de soluções que ajudam as empresas a enfrentar e resolver os seus maiores desafios e criar uma vantagem competitiva. Estas alterações verificam-se em áreas como: gestão da introdução de novos produtos no mercado, maior velocidade de introdução dos produtos, redução de custo e velocidade de customização do produto e por fim, gestão da complexidade de produção e suporte de produtes em uso. [GCA+10]

Relativamente à aplicação de PLM em pequenas e médias empresas, estas têm necessidades especiais e recursos limitados. Este conceito traz soluções completas desenhadas especificamente para este tipo de empresas; soluções que ajudam as empresas a responder melhor às necessidades dos clientes. Os pequenos negócios necessitam de soluções PLM totalmente integradas de modo a maximizar a sua inovação, estratégia e facilmente escalarem a sua produção para irem de encontro às necessidades de amanhã. Um dos softwares que actualmente suporta PLM para este tipo de sempresas e o Siemens PLM Software. Este *bundle* de ferramentas ajuda as pequenas e médias empresas a inovarem nos seus processos, aplicando as melhores práticas no dia-a-dia. Beneficiam de maior segurança no armazenamento dos seus dados, mais eficiência e flexibilidade em termos de produção, baixo custo de *ownership* e finalmente redução da probabilidade de erro através de uma maior colaboração entre departamentos. [GCA+10]

1.1.2 Valor de Negócio

Para a implementação do PLM ser efectuada, existe uma necessidade clara de este gerar valor no negócio e criar uma marca diferenciadora de modo a gerar uma vantagem competitiva no produto e assim ultrapassar a concorrência que a empresa enfrenta. Quando a empresa implementa o conceito PLM, aí pode avançar estrategicamente enquanto atinge resultados a curto prazo e estabelece uma plataforma para inovação. Enquanto a empresa constrói e desenvolve fundações para sucessos futuros através da cultura PLM, vai ser possível medir a influência da inovação tanto imediatamente como ao longo do tempo. Estas melhorias são visíveis graficamente na Figura 3.2 . [Min05]

Tradicionalmente, as empresas levam os seus produtos para o mercado através de processos em série que consomem bastante tempo, atrasando assim a participação de contribuidores como fornecedores e serviços de manutenção. Ao permitir que as empresas executem várias tarefas do ciclo de vida em paralelo, o PLM faz com que exista uma *streamline* a nível de etapas críticas no ciclo de vida do produto. PLM fornece conhecimento de produto alinhado, preciso e altamente sincronizado a vários departamentos da empresa numa das primeiras etapas de desenvolvimento do produto - evitando assim o impacto do custo e calendarização que acontece quando sugestões e preocupações tardias aparecem. Portanto, PLM faz com que as empresas possam ultrapassar a concorrência com produtos inovadores que atingem rapidamente o mercado. [GCA+10]

PLM permite às empresas criar, capturar e partilhar os requisitos do produto, expectativas e preferências de determinado segmento de mercado, alinhando assim esses requisitos com conteúdo que os clientes querem por um preço que podem pagar em tempo útil. Existe portanto, uma grande **Redução de custos de produção** que dá às empresas a oportunidade de reduzir gastos

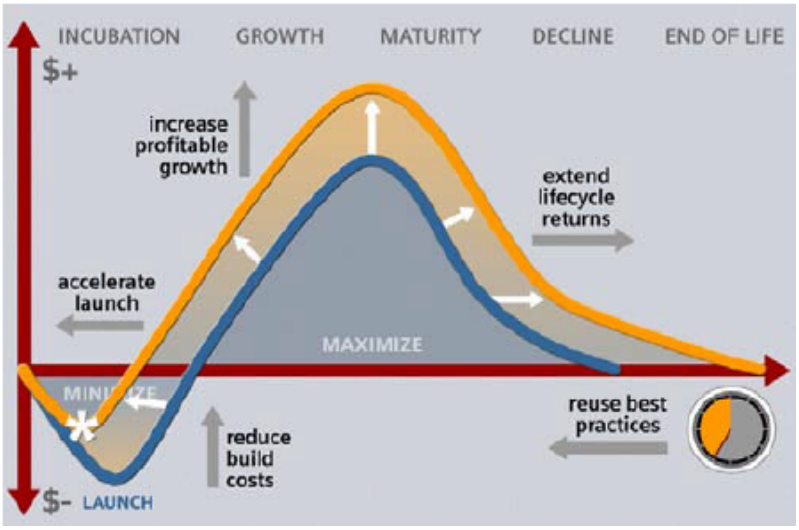


Figura 1.1: Valor de negócio do PLM
[GCA+10]

em todas as etapas do ciclo de vida do produto - o que por sua vez, permite minimizar os custos de oferta. Por exemplo, ao detectar falhas de design cedo no processo de desenvolvimento do produto, a empresa pode evitar custos desnecessários a refazer trabalho. A empresa pode usar PLM para a criação virtual de protótipos que permitem uma redução de custos associada à criação de protótipos fisicos. A implementação da metodologia PLM numa empresa facilita a extensão do período durante o qual um determinado produto pode gerar lucro através da criação de plataformas que aceleram processos, minimizam o custo de colocar um produto no mercado e maximizam o *revenue* gerado pelo produto como é possível observar na figura 1.1 PLM permite maximizar a re-utilização de processos, capitual intelectual, recursos humanos e cadeia valor através dos processos que colocam um produto no mercado. [AD05]

Por fim, provou ser bastante útil em todos os níveis de uma empresa, quer a nível vertical como horizontal. Traz assim uma proposta de valor bastante interessante e com muito potencial, sendo uma abordagem de negócio baseada em métodos e que existe uma constante análise e partilha de informação, que através de ferramentas de software consegue criar processos bastante eficientes e rentáveis.

1.2 Objetivos e Questões a ter em conta

O trabalho a desenvolver nesta dissertação consiste numa ferramenta que faça interpretação linguística da informação proveniente de um artigo financeiro em concreto: apólices de seguro. Através da inserção do produto na aplicação utilizando linguagem natural, a ferramenta a desenvolver utiliza técnicas de processamento de linguagem natural para transformar parcialmente a informação inserida - inicialmente em linguagem natural sem restrições - num formato XML em que todo o conteúdo textual está dividido e organizado de acordo com o indicado no anexo B . Esta informação irá posteriormente poder ser obtida e/ou consultada através de um formulário

gerado e apresentado ao utilizador final.No limite, a partir desta descrição obter dados formalmente estruturados que sejam passíveis de serem transmitidos e no limite, armazenados e sujeitos a consulta por parte de diferentes utilizadores.

No que diz respeito a implementação existem um conjunto de questões que servirão como orientação para o desenvolvimento do trabalho, nomeadamente:

- Qual a melhor maneira de inserir através de linguagem natural com restrições apólices de seguro?
    1. Deverão existir sugestões que guiem o utilizador durante o processo de descrição do seguro?
- Como deve ser interpretado o texto inserido através de linguagem natural de modo a obter um XML?
    1. Além dos padrões de linguagem, deverá ser criada uma linguagem intermédia a partir do qual os dados possam ser extraídos para XML ?
    2. Qual a melhor forma de tratar dados obtidos a partir do tratamento de linguagem natural com restrições?
- Qual deve ser o critério escolhido para a especificação de *WebServices* ?

Estas questões apresentam-se como prioritárias para a aplicação ficar dotada das capacidades necessárias para o processamento e armazenamento de linguagem natural.

1.3 Contribuições

O trabalho a desenvolver no contexto desta dissertação irá trazer algo de novo para duas áreas em específico:

Relativamente à Altran Portugal, esta tese fornecerá uma compreensão acerca da forma de como a utilização de metodologias PLM poderão influenciar e actuar no contexto do sector financeiro. Podendo servir como base a desenvolvimentos futuros no sector empresarial de um novo produto que será integrado na cultura PLM.

E no âmbito científico em que é feita uma experimentação relativamemnte a um método de obtenção, formatação e transmissão de linguagem natural. Também é feita a verificação da viabilidade do uso de determinadas tecnologias ligadas ao processamento de linguagem natural de modo a serem futuramente integrada em sistemas maiores.

1.4 Estrutura da Dissertação

Tendo em conta a introdução que finda neste ponto, existem nesta dissertação mais 6 capítulos. No capítulo 1, é feita uma abordagem ao funcionamento e mencionados os principais conceitos

do PLM que dão um conhecimento geral e suficiente sobre o processo no qual se insere o trabalho a desenvolver. Relativamente ao capítulo 2, são explicados os conceitos fundamentais relativos ao processamento de linguagem natural, falando-se das soluções existentes que permitem o processamento de linguagem natural. É feito também uma abordagem à linguagem necessária para utilização em apólices de seguro. O capítulo 3 abrange todas as decisões tomadas durante o desenvolvimento da aplicação, explicando através da arquitectura até ao tipo de *input* e *output* obtido. Olhando para o capítulo 4 são demonstrados os resultados obtidos em termos de eficiência e de capacidade de processamento de conteúdo. Tirando-se por fim algumas conclusões sobre os mesmos. Por fim, no capítulo 5 é feita uma abordagem sobre o sucesso dos objectivos atingidos e sobre o trabalho futuro a realizar sobre esta dissertação..

Capítulo 2

Processamento de Linguagem Natural

O Processamento de Linguagem Natural é um campo que apareceu perto de 1950 e está ligado à área de ciências da computação, inteligência artificial e linguística cuja principal preocupação é estabelecer a ligação entre computadores e a linguagem natural falada pelas pessoas. Proveniente de um ramo da inteligência artificial, actualmente, o Processamento de Linguagem Natural assume-se como uma área de pesquisa independente. Apesar disto, em problemas mais práticos continua a ser complicado aplicar o processamento de linguagem natural para a sua resolução. Existem ainda assim algumas técnicas, algoritmos e recursos linguísticos que podem ser bastantes úteis para resolver problemas em específico. [Lis13]

Com base nas tecnologias actualmente disponíveis para aplicação em processamento de linguagem natural no contexto do desenvolvimento de um produto financeiro, uma apólice de seguro concretamente, o objectivo será aplicar o processamento de linguagem natural à introdução da descrição da apólice de seguro no sistema. Isto acontece, visto que a linguagem natural é a ponte entre os utilizadores e a aplicação com que vão trabalhar. Portanto, considerando o tipo de produto a ser descrito, o objectivo será o processamento textual da descrição tendo em conta várias restrições.

De modo a poder ser feito um processamento do texto inserido pelo utilizador, é feita uma pesquisa e estudo acerca das técnicas e algoritmos a utilizar numa abordagem de PLN. Nesta pesquisa são abordadas situações que apliquem o PLN em situações relativamente semelhantes que possam ser adaptadas de modo a atingir os objectivos estipulados para esta dissertação.

Nos restantes tópicos deste capítulo será feita uma análise das várias componentes que se inserem no PLN e que servirão como possíveis soluções para o problema em causa. Serão avaliadas possíveis algoritmos e recursos lexicais que permitirão fazer um processamento textual eficiente. Também serão analisadas *frameworks* relevantes para o desenvolvimento da ferramenta de análise linguística na qual se baseia esta dissertação. E por fim, será feita uma análise geral de ferramentas que utilizam conceitos semelhantes ao deste trabalho em termos de PLN.

2.1 Visão geral do PLN

O PLN consiste no uso e capacidade de sistemas para o processamento de frases em linguagem natural como o Inglês ou o Português, em vez de linguagens computacionais especializadas como o c++ ou o java.[1] Neste capítulo são observados os principais aspectos e técnicas que vão permitir a interpretação linguística do texto inserido pelo utilizador na aplicação em causa na dissertação.

2.1.1 Níveis Linguísticos

Existem diversas ténicas que nos permitem resolver problemas em específico que surgem para a interpretação da linguagem natural. Estas técnicas estão dividas em vários níveis dependendo do nível do fluxo de NLP que processam. O processo de interpretação mapeia as frases da linguagem natural para uma linguagem formal, mas existem diferentes tipos de processos de interpretação dependendo do nível linguístico que estiver a ser interpretado.[2] Existem portanto diferentes níveis linguísticos a considerar [ddS12]:

- **Léxico**: O léxico indica o tipo de discurso de palavras que vai ser utilizado. Ou seja, foca-se na estrutura e forma das palavras, nomeadamente nas regras através das quais as *palavras-raíz* (palavras base a partir das quais se podem gerar várias palavras semanticamente relacionadas) podem ser transformadas noutras palavras, gerando desta maneira o vocabulário da linguagem natural dada.

- **Sintaxe**: O principal foco relativamente à sintaxe é a ordem das palavras e a maneira como estas estruturam a frase baseado na sua função. Podendo ser um adjectivo, verbo, substantivo, etc. O *parsing*, isto é, a análise automática da sintaxe é feita através de uma *parse tree* constituida pelos elementos das frases com funções específicas atribuídas. [Lis13] Esta abordagem acontece assumindo que o utilizador constrói as frases palavra a palavra.

- **Semântica**: Este nível fala da atribuição de significado às palavras. Estas podem referir e ter conceitos associados. Entendendo o significado de cada palavra, identifica o significado da frase como um todo. Por exemplo, através da representação das frases como fórmulas matemáticas.

- **Pragmática**: Como as frases são utilizadas em diferentes situações e como isso afecta a interpretação das mesmas. Isto é, a variação da interpretação das frases nos diferentes contextos.

É de notar que a aplicação de técnicas de PLN em cada um destes níveis não deve ser feita sem antes serem aplicadas restrições de linguagem natural na linguagem inserida.

[1] http://www.mind.ilstu.edu/curriculum/protothinker/natural_language_processing.php

[2] http://www.mind.ilstu.edu/curriculum/protothinker/natural_language_processing.php(consultado a 10 de Fevereiro de 2014)

2.1.2 *Tokens* da Linguagem Natural

A utilização mais vulgar e utilizada para a definição de *tokens* em linguagem natural é através de **palavras**. Isto porque uma palavra é a unidade mais pequena que pode ser identificada dentro do contexto da linguagem natural. [Lis13]

A **Tokenization** é a conversão do *input* em partes (*tokens*) de modo a serem processados pelas técnicas de *parsing* de linguagem natural. Este processo é geralmente feito antes da análise sintáctica.

Em termos de classificação, uma determinada palavra pode ser inserida numa de duas classes: *open word* e *close word* definindo assim se o seu significado é mutável ou não. Por outro lado, através da verificação do seu conteúdo semântico podem ser classificadas como *lexical words*(nomes, verbos, adjectivos ou advérbios) e *functional words* (artigos, pronomes, conjunções e preposições). Apesar de classes independentes, estas estão ligadas pois a maior parte das *functional words* pertencem a classe das *closed words*, enquanto que a maioria das *lexical words* pertencem à classe das *open words*. [Lis13]

Adicionalmente, uma palavra pode ser classificada em termos de ***Part-Of-Speech***, o que indica a sua categoria lexical - estando desta forma relacionada com o seu comportamento sintáctico numa frase, bem como o campo semântico em que a frase se insere. O processo que permite a classificação de palavras em termos de *part-of-speech* é denominado de *tagging*. Portanto, o *tagging* é a técnica de PLN que aplica a tag *part-of-speech* mais provável a uma palavra. [Lis13]

Após serem aplicadas as *tags* a todas as palavras de uma frase, é criago um *tagset* que vai contar todas as *tags* de uma determinada tarefa de PLN. Este *tagset* deve estar de acordo com um *language model*(por exemplo uma máquina de estados) de modo a ser feita uma classificação adequada das palavras processadas.

2.1.3 Técnicas de Parsing

Tendo em conta a necessidade de analisar e compreender linguagem natural no seu todo existem duas técnicas principais para o *parsing* de textos em linguagem natural: *full parsing* e *shallow parsing*. A principal diferença entre si é o detalhe e profundidade que apresentam as *parse trees* que obtêm como resultado bem como a sua complexidade processual. [Lis13]

A abordagem mais tradicional para fazer *parsing* de um texto em linguagem natural é o **Full Parsing** que consiste na utilização de uma *feature-based grammar* (gramática com base em recursos), em que estas gramáticas validam formalmente as regras gramaticais da linguagem. Através do uso de uma gramática com um *char parser*(um tipo de parser dirigido a gramáticas ambíguas), é possível obter uma *parse tree* profunda, que é uma árvore que inclui determinados recursos da linguagem. [Lis13]

O **Shallow parsing** é uma abordagem mais recente também connhecida como *light parsing* que faz uma análise de uma frase em que identifica os seus constituintes (nomes ou verbos) mas não se especifica a sua estrutura interna. Esta abordagem é mais flexível e praticável no geral pois aplica técnicas relativamente leves de PLN para cumprir tarefas específicas que podem ser

aplicadas em casos em que não é necessário obter um entendimento geral sobre toda o texto de linguagem natural nem a sua total correcção. Uma técnica aplicável nesta abordagem é a de *chunk parsing*. Esta técnica procura padrões específicos entre pares de palavras com *part-of-speech tags* específicas que ocorrem perto uma da outra no texto, e usa esses padrões para construir tuplos armazenando assim a relação entre essas palavras. Como as estruturas construídas são relativamente planas, é possível recorrer a uma gramática de *chunks* variadas vezes para permitir a criação de uma estrutura de *chunks* com uma profundidade arbitrária. Desta maneira, é possível fazer *nesting* de *chunks*(*chunk cascading*) para a criação de *parse trees*. [Lis13] Além de demonstrar uma maior eficiência *chunk parsing* é mais flexível e permite um foco maior sobre os pedaços de informação realmente relevantes para o processamento do texto. [BKL09]

O **Alinhamento de Padrões** apresenta-se também como uma metodologia relevante para análise no estado da arte. O conceito de alinhamento é tipicamente empregado para a comparação de *strings* diferentes. O conceito base relevante neste tipo de algoritmos é a parecença verificada entre duas *strings*, sendo possível a verificação da semelhança entre duas *strings* ao serem comparadas. Geralmente este tipo de algoritmos é utilizado na comparação de *character-based strings* que podem representar sequências biológicas. Tendo no entanto potencial para aplicação na interpretação de linguagem natural. [CHL07].

Por fim, a **Extracção de Informação** tem como função extrair informação de textos em linguagem natural e armazenar essa informação de forma estruturada. [GW98] Este processo tem como objectivo obter e formatar informação relevante a partir de linguagem natural não estruturada de modo a facilitar aos utilizadores analisar problemas e obter respostas específicas e com informação concreta e relevante. Este tipo de técnica é tipicamente associado à extracção de *eventos* bastante específicos a partir de textos em linguagem natural através de técnicas de *matching* de padrões de modo a preencher estruturas baseadas em *templates* previamente definidos. [Lis13] Esta extracção é feita com a definição de regras de extracção concebidas através de um domínio específico. [GW98] Durante o processo, as técnicas de extração de informação (1) isolam as partes do texto relevantes, (2) extraem a informação relevante desses fragmentos e (3) integrar numa framework coerente de acordo com o *schema* bem definido. Qualquer técnica que usa *matching* de padrões de linguagem para armazenar dados de forma estruturada pode ser considerada como de extracção de informação. [Lis13]

2.2 Recursos Lexicais

O **WordNet** é um dicionário de Inglês orientado semanticamente com uma estrutura bastante rica. O NLTK inclui o WordNet inglês que inclui nomes, verbos, adjectivos e advérbios que estão agrupados em *sets* de sinónimos - *synsets*. Ao estabelecer bastantes relações semânticas entre *synsets*, a WordNet acumula o montante significativo de conhecimento processado através de uma maneira processável apenas por computador. [Lis13]

Relativamente ao **VerbNet** consiste numa base de dados lexical que agrupa verbos de acordo com a sua ligação sintaxe-semantica. Os seus verbos estão organizados por classes de verbos,

em que estão todos coerentemente organizados sintáctica e semanticamente de acordo com um schema. [Lis13] Desta maneira é observável a formação das estruturas para a construção de frases transitivas, intransitivas e proposicionais.[3]

Finalmente, o **PropBank** ou projecto **Proposition Bank** , criou um repositório de texto anotado com informação sobre proposições semânticas.[4] Este projecto é baseado na assumpção de que a maior parte de anotações semânticas exequíveis são anotações com a estrutura *predicate-argument* para verbos, modificadores no particípio e nominalizações. Assim, este projecto é baseado na consistência da estrutura dos verbos. Na prática, esta base de dados é uma outra anotação no *Penn Treebank* fornecendo exemplos de frases com *tags* e de verbos. Actualmente, esta base de dados está integrada noutros recursos linguísticos com o *VerbNet*. [Lis13]

2.3 *Frameworks* de PLN

Neste ponto, além dos algoritmos de PLN e recursos linguísticos fornecidos anteriormente na dissertação, são faladas as **frameworks**, visto que estas fornecem as utilidades necessárias para a implementação dos algoritmos para a execução de tarefas como a *tokenization*, o *tagging* e o *chunking*.

O **Antelope**, que significa *Advanced Natural Language Object-Oriented Processing Environment*. Esta é uma framework de PLN para plataformas Microsoft .NET que fornece uma interface a um conjunto de algoritmos PLN e recursos linguísticos, assim suportando as tarefas que lidam com o análise léxica, semântica e sintáctica.No entanto é uma tentativa de integrar processos dispersos e não permite um melhor entendimento dos algoritmos, nem permite a sua alteração. [Lis13]

Outra framework é o **NLTK** (Natural Language Toolkit) que fornece uma biblioteca *open-source* em python de módulos para PLN, que contém vários recursos linguísticos. Permite a utilização de algoritmos de PLN como as de (1) *word tokenizer*, (2) *tokenizer* de frases, (3) *stemmers*, (4) classificadores, (5) *part-of-speech taggers* e (6) *chunkers*. Esta *framework* apresenta-se bastante bem documentado e relativamente fácil de aprender e utilizar, no entanto apresenta problemas de re-usabilidade. [Lis13]

Por fim, o Apache **OpenNLP** apresenta-se como uma *framework* que fornece fundações para suportar funções de *machine learning*. Esta framework junta numa biblioteca de Java as tarefas mais típicas de PLN nomeadamente: (1) *word tokenizer*, (2) *tokenizer* de frases, (3) *stemmers*, (4) classificadores, (5) *part-of-speech taggers* ,(6) *chunkers* e resolução *coreference*. Apesar de tudo este *toolkit* ainda é bastante recente e passível de erros.

[3] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html(acedido dia 10/02/2014)
[4] http://verbs.colorado.edu/~mpalmer/projects/ace.html(acedido dia 10/02/2014)

2.4 Aplicação de interpretação de Linguagem Natural para levantamento de Requisitos

Neste ponto da dissertação será abordado um projecto que na pesquisa se destacou mais como semelhante ao projecto a ser desenvolvido nesta dissertação. Esta solução utiliza técnica úteis e relevantes no contexto do esta da arte desta dissertação.

O projecto mencionado é o **RSLingo: a Formal Requirements Specification Approach based on Linguistic Patterns**. O RSLingo é um *toolkit* que fornece uma aproximação linguística que permite um melhoramento de qualidade na área da especificação de requisitos, baseando-se em duas linguagens e no mapeamento entre as duas: **RSL-PL**, uma linguagem extensível para lidar com a extracção dos requisitos escritos em linguagem natural; e RSL-IL, uma linguagem formal com um *set* fixo de estruturas para representar os requisitos especificados. [ddS12]

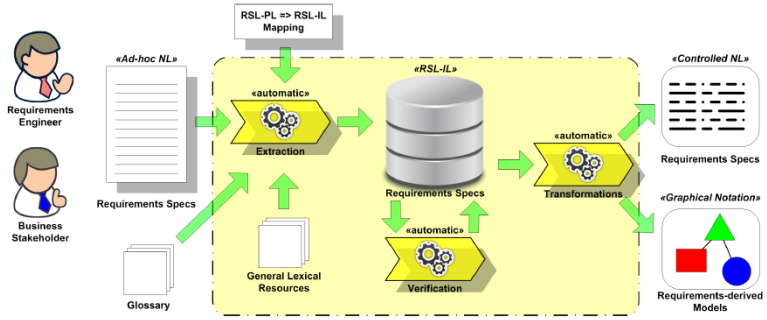


Figura 2.1: Visão geral do RSLingo ao nível de projecto [ddS12]

Nesta abordagem, é considerado que o primeiro passo para a automação e tratamento da representação textual de requisitos, é uma representação *white-box* para extrarir e analisar o significado dos requisitos introduzidos. [ddS12]

Estando este tipo de abordagem inserida numa classe de técnicas de *shallow parsing*, fazer *chunk parsing* com técnicas de PLN (geralmente referidas como técnicas de *full parsing*).Uma *parse tree* de vários níveis e completamente anotada é produzida, considerando a maior parte das restrições linguísticas de modo a verificar se árvore é válida. Contrariamente, o objectivo do *chunk parsing* não é forçar correcção gramatical, mas sim explorar um alinhamento entre as estruturas das frases e as suas semânticas. A estrutura de cada frase (padrões de linguagem) é baseada na posição relativa e *part-of-speech* de cada palavra que contém. [ddS12]

Assim, é possível ganhar vantagens neste alinhamento de modo a extrair o significado dos requisitos ou, pelo menos, extrair informação útil obtendo requisitos mais exactos. Além de utilizar menos poder computacional, a utilização de *chunk parsing* é mais flexível e robusta ao extrair informação. [ddS12]

Como é ilustrado na figura 2.1, o nível de projecto consiste em aplicar os seus conceitos através do *toolset* do RSLingo durante a criação de uma especificação de software.

Esta ferramenta faz uma abordagem avançada em termos de distinção conceptual de dois factores: (1) a definição de padrões linguísticos e a (2) especificação formal de requisitos. Esta separação clara diz respeito às duas linguagens utilizadas: RSL-PL e RSL-IL respectivamente. Por fim, os algoritmos de *parsing* utilizados são baseados em princípios de programação dinâmica, em vez de utilizar recursividade e ser *model-parametrized* como aqueles utilizados pelos *toolset* da CIRCE. [ddS12]

Capítulo 3

Concepção e Implementação

Neste capítulo é feita a descrição e especificação de todo o processo de implementação, através da explicação da linha de raciocínio usada para atingir os objectivos pré-determinados e para a tomada de decisões ao longo do desenvolvimento. Esta descrição é feita ao longo de 3 sub-secções.

Na primeira sub-secção denominada de **Arquitectura** é feita uma análise de alto nível da constituição da aplicação e da maneira como interagem os componentes entre si. É também feita uma explicação acerca do fluxo de dados ao longo de toda a aplicação e das tecnologias a serem utilizadas para o tratamento de dados.

Relativamente à sub-secção, **Estrutura de classes**, a estrutura da aplicação é analisada ao pormenor no que diz respeito à interacção das classes, funções e *packages* entre si que permitem o funcionamento da mesma. Isto facilita um entendimento da estrutura funcional da aplicação.

Na terceira sub-secção, ***Inputs* e *outputs***, é feita uma abordagem mais directa da maneira como é tratado o produto financeiro, nomeadamente a nível de estrutura textual. É feita também uma análise do estado de entrada e saída dos dados a serem tratados de modo a corresponderem às necessidades da aplicação em termos de procesamento de linguagem natural e tratamento de dados.

Por fim, na quarta sub-secção - **Método de processamento** - é explicada a aplicação das *frameworks* utilizadas no contexto da aplicação e a maneira como são processados os dados à medida que se vai progredindo no *flow chart*.

3.1 Arquitectura

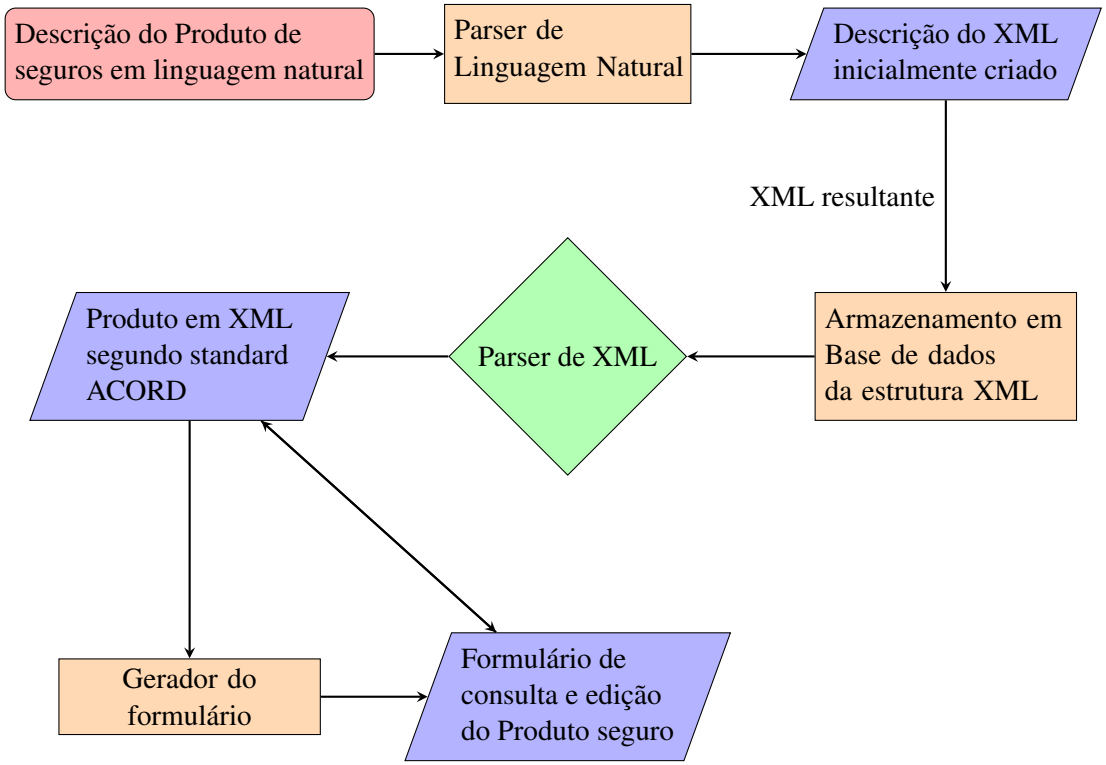
Neste capítulo é feita uma análise de alto-nível do funcionamento da aplicação. Como é observável na figura 3.1, o utilizador inicialmente apresenta uma descrição textual em linguagem natural do produto financeiro a ser tratado. Esta descrição textual vai ser numa primeira fase tratado por um *parser* de linguagem natural e transformado numa estrutura XML. Para efeitos de tratamento de linguagem natural é utilizada uma *framework* denominada de **Stanford CoreNLP**

que fornece as ferramentas necessárias ao tratamento de linguagem natural e que permitem a criação de uma estrutura XML com todos os conteúdos que estavam previamente em linguagem natural não-estruturada.

De seguida, as estruturas resultantes do *parsing* são armazenadas numa base de dados **BaseX**. Esta tecnologia fornece uma base de dados *light-weight*,escalável e nativa para XML que dá grande suporte ao utilizador através de uma GUI e acesso através de XPath/XQuery.

As estruturas XML são manipuladas através da utilização de **XML DOM**. Através do uso de *DOM (Document Object Model)*, é possível aceder e manipular os ficheiros XML retirados da base de dados. Sendo assim possível retirar o conteúdo textual necessário e manipulá-lo de modo a, através de funções de processamento alocar o conteúdo necessário a um ficheiro XML construído de acordo com os standards **ACORD**, sendo que se adequam à especificação e estruturação de apólices de seguro.

Figura 3.1: Diagrama de fluxo de dados da solução a desenvolver



Por conseguinte, o xml devidamente estruturado e preenchido serve como base para a construção de um formulário que vai ser apresentado ao utilizador, podendo este proceder à sua consulta e edição, ficando estas armazenadas no ficheiro XML.

3.2 Estrutura de classes

Esta aplicação é desenvolvida maioritariamente em Java, através do uso de *frameworks* como a *Stanford Core-NLP* para o processamento de linguagem natural, *swing* e *cookSwing* para assegurar

uma interface para interacção com o utilizador. O armazenamento é também assegurado por uma base de dados nativa para XML com o nome de *BaseX*. A partir do código Java foi gerado usando o *Enterprise Architect* um diagrama de estrutura de classes.

Como é visível na figura 3.2 , a aplicação tem como principal classe, a *Main*, a partir do qual correm os principais processos para *parsing* e tratamento de dados. A classe *NaturalLanguageParser* contem as funções necessárias para o processamento da linguagem natural. Sendo o responsável pela criação da primeira estrutura XML a ser armazenada na base de dados. Todo o tratamento posterior à criação do XML dá-se na classe *XMLParser*.

Nesta classe é criada uma base de dados *BaseX* direcionada unicamente ao armazenamento de XML, que guarda todos os ficheiros XML criados. A partir desta base de dados são retirados os xml que são utilizados em funções de *parsing* de modo a obter um xml estruturado de acordo com normas ACORD. Após estar criada a nova estrutura, apenas uma parte dos dados processados será colocada nos devidos campos do XML.

Por fim, existem interfaces criadas a partir de *swing* que permitem a visualização das diferentes partes do processo.

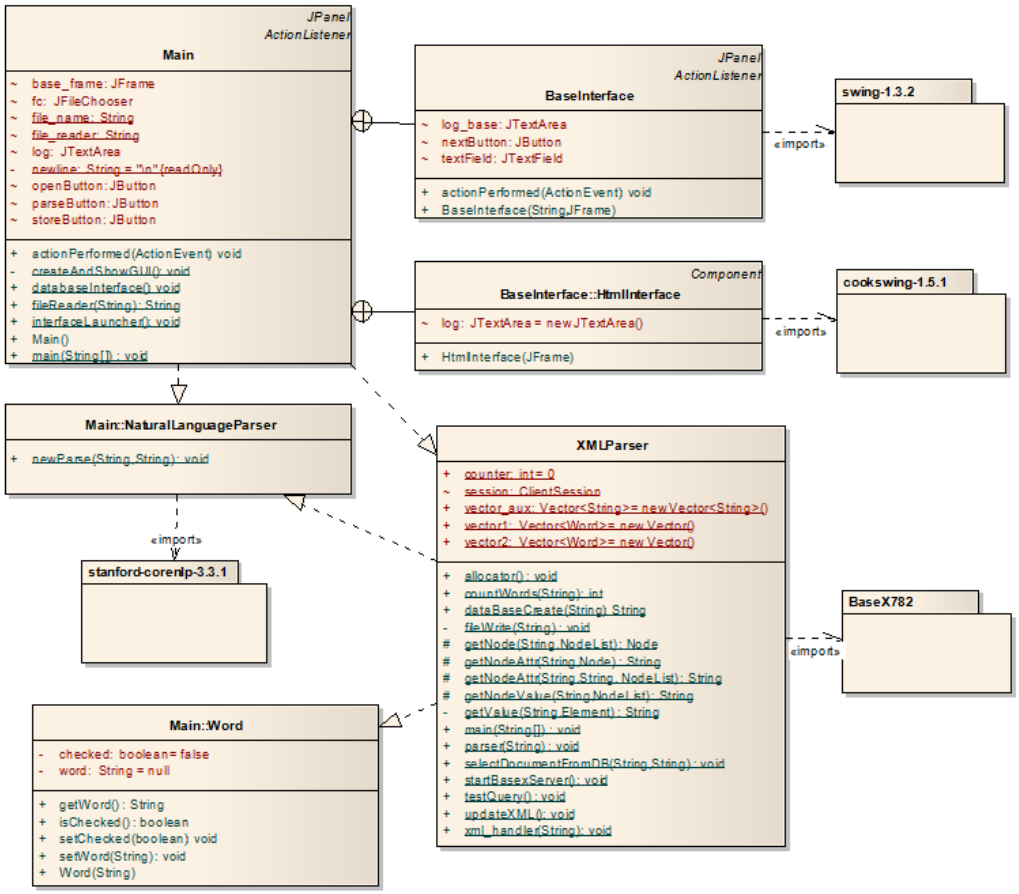


Figura 3.2: Arquitectura da aplicação

3.3 *Inputs* e *outputs*

A especificação do produto financeiro a utilizar inicialmente nesta dissertação é restringido a apólices de seguro. Este tipo de texto é demonstrado nos Casos 1 e 2 , servindo para propósitos de teste e de definição de padrões de linguagem visto que inicialmente a aplicação vai ler este tipo de documento para se atingir o fluxo completo de criação do produto. A criação de apólices de seguro implica a criação de um texto em linguagem com o objectivo de estipular as características textuais importantes a inserir no sistema para serem analisadas e estruturadas.

Inicialmente, e de acordo com o enviado pelo orientador na empresa, as duas representações textuais enviadas, para efeitos de teste e identificação de restrições a serem processadas pela aplicação, estavam em português. No entanto , a *framework* escolhida para utilizar no processamento da linguagem natural presente nesses textos, provou ser mais capaz e eficiente se estes textos estivessem em inglês. Portanto, e após aprovação do orientador supracitado, os textos de teste foram traduzidos, correspondendo aos textos do **Caso 1** e do **Caso 2**.

> **Caso 1**: The product Car Insurance is composed by the Liability Converage and self damage. The insured must provide the name, age and car value. To hire an insurance, the insured must be over 21 years old. The own damage award is equal to 0.02% multiplied by the car's value. The award for the civil liability is equal to €110.

No que diz respeito ao **Caso 1**, é feita uma verificação do conteúdo textual relevante a ser alocado no *output* final retirado das indicações dadas na norma ACORD indicada em anexo. Esta filtragem de conteúdos é feita mediante a necessidade dos campos a preencher no XML do formulário de saída. Esta selecção é visível no *listing* 3.1, em que a informação referente à idade do segurado é alocada ao campo *SupplementaryNameInfo*.

Relativamente ao XML de saída, este foi construído com base nas normas ACORD a aplicar em apólices de seguro de acordo com o que está indicado no anexo D.2 . Baseado no *schema* aí indicado, através da criação do XSD cuja esquemática está indicada na figura em anexo(D), é gerada a esquemática XML que proporciona a criação do XML, que por sua vez dá lugar ao *output* com que o *end user* tem contacto.

```
1 <NameInfo>
2    <CommlName>CommlName0</CommlName>
3    <PersonName>PersonName0</PersonName>
4    <TaxIdentity>0</TaxIdentity>
5    <SupplementaryNameInfo> insured to be older than 21 years old</
        SupplementaryNameInfo>
6 </NameInfo>
```

Listing 3.1: Excerto nº1 do XML de saída

Numa segunda fase, para o povoamento do xml que permite a criação do formulário de saída é feita uma alocação de conteúdo a campos pertencentes a *AccidentViolation*. Aqui é visível a alocação de conteúdo nos campos de *DamageTotalAmt* e *Coverage*, sendo que este pertence ao campo *LossPayment*. Toda a informação aque é inserida no xml, é proveniente dos casos de teste, sendo que dependendo do texto colocado para inserir, isto é, o texto de *input*, terá como consequência um preenchimento do *output*.

É de notar que os campos não mencionados, estão sujeitos a edição através de formulário. Qualquer campo do ficheiro XML pode ser alterado mediante as opções do utilizador que interagir com o *output*. Relativamente ao ficheiro XML que é visualizado em formato de formulário no *output* final, este, encontra-se no anexo **??**.

```
1   <AccidentViolation>
2      <DamageTotalAmt>
3   own damage award is equal to 0.02% multiplied by the car's value
4      </DamageTotalAmt>
5      <LossPayment>
6         <Coverage>
7     award for the civil liability is \texteuro{110}
8         </Coverage>
9         <LossPaymentAmt>
10        </LossPaymentAmt>
11     </LossPayment>
12     <ExcessSpeed>
13     </ExcessSpeed>
14     <ConvictionsDuration>
15     </ConvictionsDuration>
16  </AccidentViolation>
```

Listing 3.2: Excerto nº2 do XML de saída com dois campos com informação alocada

O **Caso 2** apresenta-se como o segundo caso de teste para a aplicação de tratamento de dados. Neste caso existe uma quantidade maior de texto não estruturado a ser tratado, apresentado-se com conteúdo o suficiente para preencher outros campos presentes no *output* final.

> **Caso 2**: The car insurance product is composed by Liability coverages, own damage and theft. The insured must provide the name, age and number of years of license. To have an insurance is required to the insured to be older than 21 years old and driving license for over 2 years The own damage premium is equal to 0.02% multiplied by the value of the car with a 0.1% discount multiplied by the number of years of license with. The premium liability is equal to €110 if the number of years of license is less than 5 years, €90 otherwise. The prize

theft is equal to 0.01% multiplied by the value of the car. If all covers are hired is applied a discount of 0.005% global coverage to all.

Relativamente ao **Caso 2** existe mais conteúdo a ser analisado devido à óbvia maior complexidade lexical e gramatical do texto presente no caso. Como tal, no *listing* 3.3 existem outros campos alocados como o *Coverage* e *PaymentOption* em que é observável a alocação de pequenas partes provenientes inicialmente do texto de linguagem natural.

```
1     <PersPolicy>
2       <ContractTerm>
3       </ContractTerm>
4       <TotalPaidLossAmt>
5       </TotalPaidLossAmt>
6       <OtherOrPriorPolicy>
7          <ContractTerm>
8          </ContractTerm>
9          <LengthTimeWithPreviousInsurer>
10         </LengthTimeWithPreviousInsurer>
11         <Coverage> discount of 0.005% global coverage to all </
               Coverage>
12      </OtherOrPriorPolicy>
13      <PaymentOption> 0.0001% discount multiplied by the number of
           years of license </PaymentOption>
```

Listing 3.3: Excerto nº3 do XML de saída

No exemplo apontado na figura 3.4, são as *tags DamageTotalAmt* e *Coverage* cujo conteúdo é alterado. Aqui o critério para o algoritmo de alocação de dados foi a ligação entre as palavras *Damage* e no caso da *Coverage*, foram as palavras *premium liability* que serviram para a alocação das frases correctas aos campos correctos.

```
1      <AccidentViolation>
2       <DamageTotalAmt> own damage premium is equal to 0.02%
           multiplied by the value of the car </DamageTotalAmt>
3       <LossPayment>
4          <Coverage> premium liability is equal to 110 if the number
              of years of license is less than 5 years,
5      90 otherwise prize theft is equal to 0.01\% multiplied by the
          value of the car </Coverage>
```

```
6           <LossPaymentAmt>
7           </LossPaymentAmt>
8        </LossPayment>
9        <ExcessSpeed>
10       </ExcessSpeed>
11       <ConvictionsDuration>
12       </ConvictionsDuration>
13    </AccidentViolation>
14    <License>
15       <LicenseTerm> driving license for over 2 years </LicenseTerm>
```

Listing 3.4: Excerto nº4 do XML de saída em que já é especificado o *LicenseTerm*

Estes *outputs* demonstrados são o exemplo mais objectivo e directo dos resultados finais que vão permitir a geração do formulário a demonstrar ao utilizador. É utilizada como referência para alocar o conteúdo associado a cada *tag*, as indicações dadas pela referência ACORD (*ACORD Property and Casualty Transaction Message Specifications*). Através desta referência é possível analisar e saber o conteúdo que cada *tag* deve indicar. No caso da *Coverage* que descende de *AccidentViolation* e *LossPayment*, campos estes que especificam o tipo de incidente ocorrido e no casso do *LossPayment*, é indicada a informação que diz respeito à cobertura dos danos cobertos num determinado caso. Olhando então para a *Coverage* propriamente dita, após consulta da referência ACORD verifica-se que o conteúdo indicado para este campo diz respeito ao agregado de informação dirigida à cobertura do incidente em causa. [SBW+]

O campo *PaymentOption* encontra-se como descendente do *PersPolicy* que faz referência à *Personal Policy*. O conteúdo aqui é dirigido a condições de pagamento ao cliente em caso de incidente. [SBW+] Estes são alguns exemplos da verificação da informação a insderir em campos verificada através das normas. A informação referente a cada campo deve ser sempre consultado nas normas ACORD. Isto porque existe uma grande quantidade de campos com informação muito específica e nestes casos de teste nem toda a informação que estava na forma de linguagem natural tem lugar no formulário final.

De ressalvar, que neste ponto, a interacção do utilizador para com o formulário pode ser feita de duas maneiras: por edição dos campos com texto já alocado e por adição de conteúdo nos campos sem qualquer conteúdo.

3.4 *Parser* de Linguagem Natural

Tendo em conta a necessidade de tratamento de dados que existe patente a todo o fluxo de dados da aplicação, na sua arquitectura existem dois pontos fundamentais de passagem que permitem a existência de uma diferenciação relevante entre os dados inseridos e os dados à saída. Diferença esta observável na secção referente aos *inputs* e *outputs* da aplicação. Nesta secção é feita uma análise de funcionamento, explicado o raciocínio concebido para a criação do *parser*

de Linguagem Natural e explicada a maneira como foram aplicadas as tecnologias necessárias em contexto para a crição do mesmo.

Para a criação deste *parser* foi escolhida uma *framework* criada especificamente para o processamento de linguagem natural. Este *parser* foi estritamente criado para criar uma estrutura passível de ser armazenada a partir de linguagem natural fornecida pelo utilizador.

A escolha desta *framework* deveu-se ao facto de se ter mostrado a ferramenta mais capaz e eficiente para atingir o objectivo deste *parser*. Um parser de linguagem natural é um aplicação que deduz a estrutura gramatical das frases em análise, como por exemplo, que grupos de palavras devem ser agrupadas e dadas como frases e os diferentes papéis gramaticais das mesmas no texto.

```
<token id="4">
      <word>Insurance</word>
      <lemma>insurance</lemma>
      <CharacterOffsetBegin>18</CharacterOffsetBegin>
      <CharacterOffsetEnd>27</CharacterOffsetEnd>
      <POS>NN</POS>
      <NER>O</NER>
    </token>
```

Listing 3.5: Exemplo de um *token*

Como é observável no *listing* 3.4, a criação de *tokens* permite manipular cada palavra individualmente e obter uma série de informações de carácter sintáctico e gramatical.

Numa primeira fase, é utilizado um *tokenizer* que divide o texto numa sequência de *tokens*, que basicamente correspondem a cada palavra do texto. Existe uma classe expecífica para a sua aplicação direcionada à língua inglesa. A classe *PTBTokenizer* é um *tokenizer* eficiente, rápido e determinístico que consegue processar texto até 200,000 *tokens* por segundo. Através do uso de determinadas heurísticas consegue decidir que palavras pertencem a determinadas frases e quando estas terminam ou não.

```
<parse>(ROOT
    (S
    (NP (DT The) (NN product) (NN Car) (NN Insurance))
    (VP (VBZ is)
    (VP (VBN composed)
    (PP (IN by)
    (NP
    (NP (DT the) (NNP Liability) (NNP Coverage))
    (CC and)
    (NP (NN self) (NN damage))))))
```

```
11          (. .)))
12
13          </parse>
```

Listing 3.6: Exemplo de uma *parse tree*

A *framework* manipula o texto de linguagem natural criando diversas estruturas. Nomeadamente, uma *parse tree* por cada frase como é exemplificado no 3.6. Esta árvore resulta das relações gramaticais existentes entre os constituintes da frase, originando assim uma distinção principal entre nós terminais e nós não-terminais.

No que diz respeito à árvore, o S (*sentence*) representa a estrutura que está no nível mais elevado, indicando assim o início de uma árvore a representar uma frase. De seguida o nó NP (*noun phrase*) indica um sintagma nominal onde está contido o sujeito da frase. Relativamente ao nó VP (*verb phrase*) é onde armazenado o predicado da frase em questão. Estes nós são os mais relevantes para partir uma frase em duas partes relevantes. Descendo desse nível existem nós para indicação dos verbos, artigos (definidos ou indefinidos), determinantes, entre outras características gramaticais relevantes no contexto da linguagem natural.

Estes dois processos anteriormente falados, são os mais relevantes de modo a ser criada uma estrutura coerente e pronta a passar ao próximo passo do processo. Os resultados em concreto relativos aos casos de teste já apresentados podem ser consultados em anexo.

3.5 *Parser* de XML

Após a criação do XML resultante do processamento de linguagem natural, este é armazenado na base de dados dedicada para XML. Nesta base de dados que é criada assim que é gerado o primeiro ficheiro XML, são armazenados todos os ficheiros de maneira estruturada. É possível executar operações CRUD sobre a base de dados, apesar de esta não ser relacional pois é nativa para XML. Posteriormente esta base de dados é consultada sendo feita uma seleção do XML desejado . É neste ponto que é necessário um processamento direccionado ao conteúdo do XML de modo a poder ser feita uma manipulação e posterior alocação de dados ao XML final de *output* que estará de acordo com as normas ACORD.

Para a manipulação de XML, a tecnologia utilizada, como foi anteriormente mencionado, foi a DOM (*Document Object Model*). Isto permitiu manipular o conteúdo de todos os nós do DOM resultante do XML. Indo buscar a cada nó da árvore o conteúdo textual, é criada uma estrutura para armazenar todas as palavras, de forma ordenada, sobre a forma de *string*. De seguida, a partir do XML ACORD (o XML de saída), é criado outro objecto DOM. Desse objecto são extraídas toda as *tags*. Ao serem analisado o conteúdo das duas estruturas, existe a verificação de um *match* entre *tags* e palavras provenientes do texto de entrada. Esta verificação é feita segundo regras inicialmente estipuladas que estão direcionadas aos textos de entrada do género dos casos de teste. Quando acontece uma verificação positiva, as palavras que se encontram na mesma frase que

aquelas em que foi assinalado um *match*positivo vão ser alocadas ao campo no XML de saída com a respectiva *tag*.

Neste processo, foram completos os campos do XML de saída (normas ACORD). Este XML serve para gerar um formulário HTML, que através do uso de uma *framework* de Java chamada *CookSwing* gera um formulário XML visualizável através do *browser* no qual se podem fazer alterações que mexem com o conteúdo do ficheiro XML selecionado anteriormente da base de dados e disponível como conteúdo da aplicação.

Apesar de a aplicação estar divida em dois grandes blocos (o *parser* de linguagem natural e o *parser* de XML), as componentes de integração que são a base de dados e as operações sobre ela feitas, assim como o tratamento do XML final ligado ás normas ACORD até à interacção com o utilizador, são blocos de ligação bastante relevantes e que permitem atingir um dos mais importantes objectivos que é o armazenamento, estruturação e transmissão de linguagem natural.

Capítulo 4

Experimentação

Neste capítulo é analisada a utilização da aplicação e os resultados que daí advêm. Existem portanto duas sub-secções relevantes a explorar: a **Taxa de Execução e Processamento** e **Interacção com o utilizador**. Na primeira, são dispostos e analisados os resultados fornecidos pela utilização da *framework* de processamento de linguagem natural nos casos de teste fornecidos e explicados previamente. Na secção relativa à interacção com o utilizador são explicados os passos base para se consultar a informação que vai sendo processada e a maneira como deve ser interpretada.

4.1 Taxa de execução e processamento

Esta sub-secção é dedicada à recolha, exploração e interpretação dos dados relativos à eficiência do processamento de linguagem natural. Estes resultados foram obtidos através do processamento dos dois casos de teste fornecidos e já descritos no capítulo 3.3. Após ser iniciado o *parser* existem várias fases a ter em atenção que demoram vários segundos cada um. A duração do processamento de cada texto varia a cada utilização, pelo que foi feita uma média da duração de cada caso a partir de três experimentações, em que em todas as utilizações a capacidade de processamento disponível para a aplicação era sempre semelhante.

Após terem sido feitas as experimentações, as médias de tempos obtidos para cada caso foram expostas numa tabela, sendo diferenciados os casos de utilização e as diferentes fase pelas quais passa o *parser* de modo a poder ser feita uma análise mais pormenorizada dos métodos utilizados para o processamento de linguagem natural.

O primeiro processo a ser necessário neste caso é a Leitura do POS (*part-of-speech*). Este processo consiste no *marking up* das palavras no texto de linguagem natural, identificando-as como parte de um contexto fazendo associações entre elas a nível gramatical. Basicamente, associa uma palavra com aquelas que estão adjacentes ou próximas. Neste caso, é feita apenas uma leitura de um modelo do POS pelo que o tempo de execução é bastante semelhante entre os dois casos, não sendo algo que interage no imediato com o texto escrito.

Processamento	**Caso 1**	**Caso 2**
Leitura do POS	3.667s	3.233s
Loading Classifier		
LC1	7.5s	8.2s
LC2	8.533s	4.567s
LC3	4.467s	8.467s
Loading Parser from serialized	1.367s	1.667s
Escrita do XML	2.607s	8.825s
Tokenization	2.467s (67 tokens a 23.0 tokens/s)	8.267s (139 tokens a 9.9 tokens/s)
Pipeline Setup	30.367s	34.333s
Tempo total	33.067s	43.1s

Tabela 4.1: Tabela com tempos de execução

De seguida é feito o *Loading* dos *classifiers*, provenientes das bibliotecas NER (*Name Entity Recognizer*). Esta entidade tem como objectivo fazer uma diferenciação das palavras através da identificação das mesmas como sendo de diferentes géneros, ou se são nomes de objectos ou de pessoas. Aqui, existem 3 bibliotecas diferentes a serem utilizadas, daí a existência de três tempos de execução diferentes que como no processo anterior acabam por sofrer variações que estão apenas relacionadas com a velocidade de processamento da ferramenta utilizada na altura da utilização da mesma.

Ao ser feito o *Loading* do *parser*, significa que as ferramentas estão todas prontas a processar o texto que é fornecido. Novamente, aqui o tempo acaba por ser desprezível, havendo uma média de tempos bastante parecida em ambos os casos.

Agora que o ficheiro é processado, após o devido número de experimentações, verificou-se que a geração do ficheiro XML demora em média cerca de quatro vezes mais no **Caso 2**. Isto deve-se ao facto de o segundo caso apresentar um maior número de palavras e consequentemente uma maior complexidade.

A *Tokenization* consiste no processo de dividir um bloco de texto em frases, palavras, símbolos e outros elementos relevantes no contexto que são considerados *tokens*. Estes, são relevantes para numa fase posterior ser efectuado *parsing* sobre os elementos recolhidos de modo a analisar a sua contituição gramatical e lexical. A eficiência deste parte do processo depende também da complexidade do texto, tanto a nível de número de palavras, como nas relaxões entre elas. Observa-se portanto que demora quatro vezes mais no **Caso 2** do que no outro caso, havendo também um maior ritmo de *tokenization* no segundo caso.

No total, a *pipeline* através do qual acontece todo o processo demoara em média entre 30 e 34 segundos a ser preparada. Desta maneira, verifica-se que o *parser* de linguagem natural processa o **Caso 1** em média em 33 segundos e o **Caso 2** em 43.1s. As principais razões para a diferença de tempo durante o processamento devem-se ao número de palavras no texto e principalmente, porque o **Caso 2** apresenta uma maior complexidade sintática, gramatical e lexical, provocando desta maneira uma maior demora nos processoes que constituem o processamento textual.

Esta é a primeira fase e a mais relevante, no processamento de linguagem natural. A partir dos resultados obtidos no processamento de linguagem natural, é possível moldar o conteúdo obtido incialmente por texto e armazená-lo e alocá-lo noutra estruturas que permitam um *output* mais intuituivo, resumido e facilmente editável.

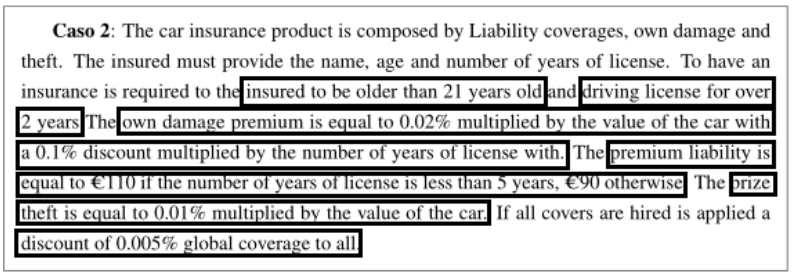

Caso 2: The car insurance product is composed by Liability coverages, own damage and theft. The insured must provide the name, age and number of years of license. To have an insurance is required to the insured to be older than 21 years old and driving license for over 2 years. The own damage premium is equal to 0.02% multiplied by the value of the car with a 0.1% discount multiplied by the number of years of license with. The premium liability is equal to €110 if the number of years of license is less than 5 years, €90 otherwise. The prize theft is equal to 0.01% multiplied by the value of the car. If all covers are hired is applied a discount of 0.005% global coverage to all.

Figura 4.1: As frases assinaladas são as selecionadas para serem alocadas no formulário final no Caso 2

Verificando agora a eficiência em termos de aproveitamento textual, de acordo com a figura 4.1, existe um grande aproveitamento do texto inserido. O mesmo acontece no Caso 1 como assinalado na figura 4.2, apesar de haver menos conteúdo e por isso menos conteúdo selecionado . Na figura 4.1, cerca de 80% do texto é aproveitado para demonstração final no formulário, e portanto estão sujeitos a edição. Alterações no conteúdo final serão reflectidas no ficheiro XML e nunca chegam as textos de linguagem natural inicialmente inseridos.

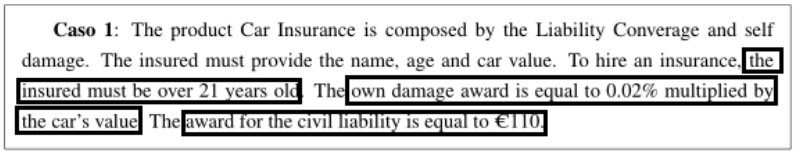

Caso 1: The product Car Insurance is composed by the Liability Converage and self damage. The insured must provide the name, age and car value. To hire an insurance, the insured must be over 21 years old. The own damage award is equal to 0.02% multiplied by the car's value. The award for the civil liability is equal to €110.

Figura 4.2: As frases assinaladas são as selecionadas para serem alocadas no formulário final no Caso 1

Como consequência da existência de menos conteúdo da figura 4.2, apenas cerca de 50 % do texto é selecionado. No entanto, o facto de haver muito conteúdo selecionado em ambos os caos, isso revela uma certa ambiguidade que pode resultar da interpretação das normas ACORD. O conteúdo selecionado também depende da qualidade (e do contexto relativo ao formulário ACORD) do texto em linguagem natural criado para descrever o produto de seguro.

4.2 Interacção com o utilizador

Relativamente à primeira fase do processo, a introdução do ficheiro que contém o texto a ser processado é feita como é demonstrado na figura 4.3. Após a criação da estrutura xml proveniente do *parser* de linguagem natural, é feito um armazenamento da estrutura resultante do *parser* ou seja, o ficheiro xml, numa base de dados BaseX. Uma base de dados especificamente dirigida para

guardar ficheiros no formato XML. Esses resultados, aparecem numa janela, de modo a dar ao utilizador uma noção do estado do processamento de dados(4.4).

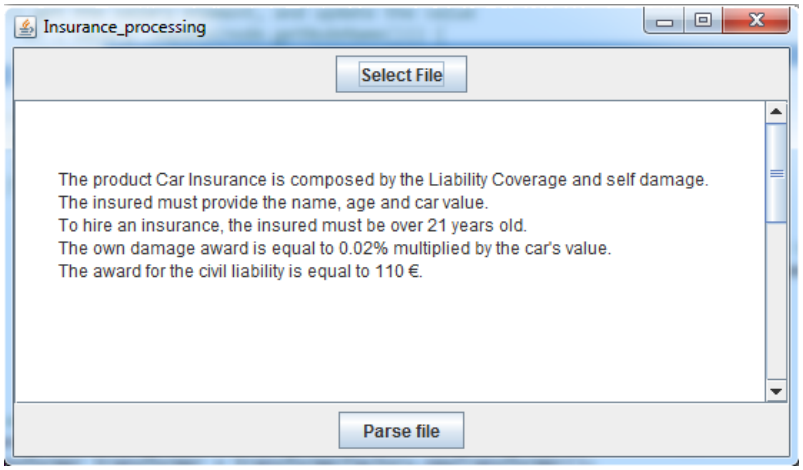


Figura 4.3: Linguagem Natural

Depois de criada a estrutura XML é posível ao utilizador a percepção do estado do XML resultante no decorrer do processo, sendo por isso visualizavel através de dados como o nome da base de dados, o local a partir do qual foi armazenado o XML e o tamanho do mesmo. Após esta fase pode ser escolhido o XML a ser consultado sobre a forma de formulário.

As figuras 4.5 e 4.6 dizem respeito a passagens recolhidas do XML de saída. A primeira figura é referente ao *listing* 3.1.

Este é outro excerto do formulário de interacção com o utilizador originário do listing 3.4.

Estas interfaces para interacção com o utilizador destinam-se principalmente a dar ao utilizador percepção sobre o que está a acontecer durante o processamento do texto que inseriu e não são uma verdadeira interface para o utilizador tirar o maior partido de todo o conceito envolvido nesta dissertação.

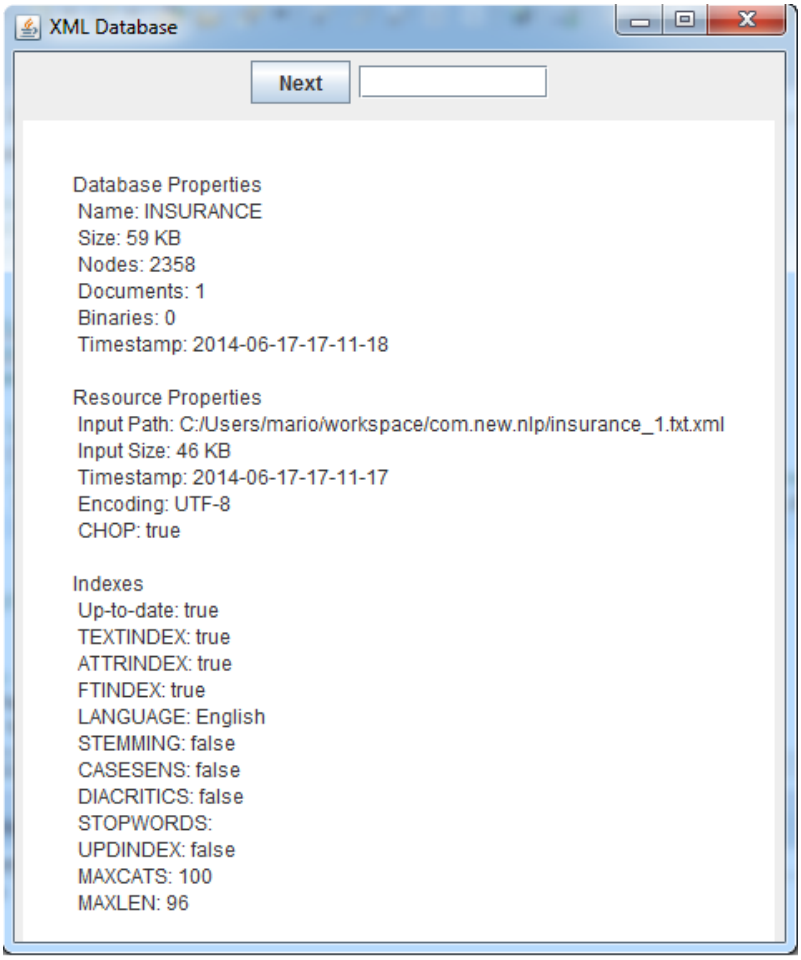


Figura 4.4: Resultados da base de dados após ser inserido um ficheiro XML

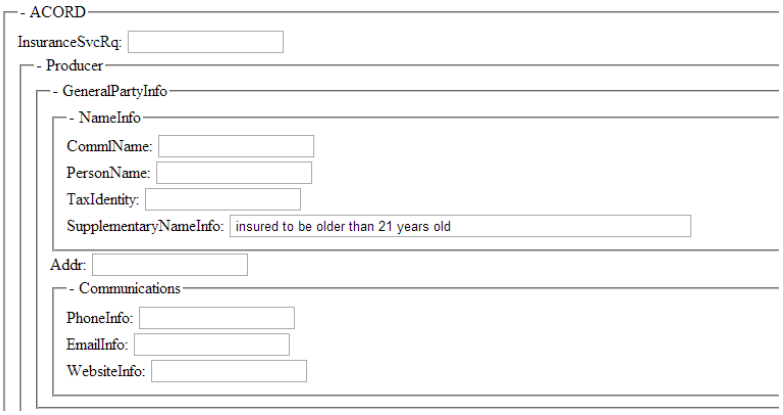


Figura 4.5: Excerto do formulário para edição e consulta do XML

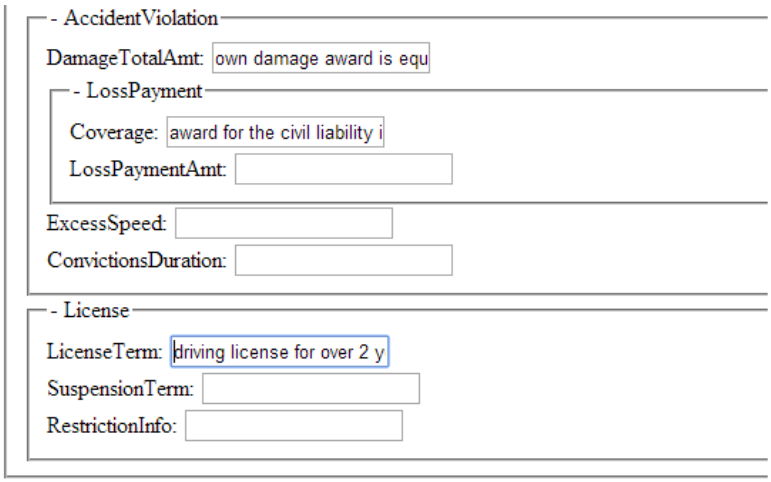


Figura 4.6: Excerto nº2 do formulário para edição e consulta do XML

Capítulo 5

Conclusões e trabalho futuro

O processamento de linguagem natural assume um papel relevante na interacção entre o utilizador e os sistemas informáticos. No caso da existência de processos de negócio implementados como o de gestão de ciclo de vida do produto, a linguagem natural devidamente restringida e posteriormente processada revela grande importância na primeira fase deste processo ao permitir a criação do produto e a sua alocação ao sistema.

Este dissertação tem como objectivo a demonstração da prova de conceito relativo à aplicação de processamento de linguagem natural e manipulação de conteúdos como parte de um processo de negócio: a gestão do ciclo de vida do produto. Esta prova de conceito em particular é demonstrado, estando o objectivo mais relevante inicialmente estipulado atingido. É feita uma demonstração da manipulação de dados através das diferentes partes do processo de linguagem natural e de XML em que o conteúdo inicial em linguagem natural é apresentado devidamente filtrado e alocado num XML final que é apresentado ao utilizador na forma de formulário.

Partindo deste objectivo, foram sendo delineados ao longo do processo de implementação objectivos relativamente secundários que são parte integrante do projecto como um todo. A parte que assume mais relevância é a do *parser* de linguagem natural. Este objectivo em particular, assume relevância em termos de eficiência e de qualidade de resultado, visto ser a primeira estrutura gerada a partir do *input* do utilizador. Relativamente à eficiência, apresentou resultados razoáveis, assim como na qualidade de resultados, mostrando ser uma metodologia exequível de ser aplicada em contextos mais práticos como casos reais de uso.

Outro objectivo que foi necessariamente estipulado, foi orientado ao processamento a ser feito ao XML resultante da fase anteriormente falada. Os objectivos principais a serem atingidos nesta fase focam-se na capacidade de filtrar as partes essenciais do conteúdo armazenado previamente e alocá-las com eficiência ao XML de saída. Este XML foi construído com base em standards ACORD fornecidos pela empresa, pelo que se encontra relativamente fidedigno e próximo de um caso real. Neste caso, não se verifica uma eficiência acima da média na selecção de texto e do seu posicionamente no momento de *output*. No entanto, os resultados foram satisfatórios na medida em que todo o conteúdo consegue ser armazenado, selecionado e demonstrado da melhor maneira.

Para a obtenção de resultados, tanto para verificação de eficiência como para avaliar a qualidade dos mesmos, foram usados os dois casos de teste fornecidos pela empresa. Este casos foram usados, visto serem para a empresa a simulação mais próxima que representa a real necessidade que deu origem a esta dissertação: facultar uma maneira de iniciar o ciclo de vida de um produto a partir de uma interacção exclusivamente dedicada a linguagem natural, sendo este o ponto em que reside o maior potencial da aplicação.

No que diz respeito ao trabalho futuro a ser feito sobre esta dissertação, além de um amadurecimento da sua arquitectura em termos de funcionalidades e complexidade de ficheiros a serem processados, falta ainda o planeamento e implementação de *WebServices* para a transmissão de conteúdo com vista a alargar a utilização da informação a um maior número de pessoas. É necessária uma fase de integração num sistema com maior complexidade e alcance em termos de gestão do ciclo de vida do protudo. Desta maneira é possível retirar valor das capacidades desenhada nesta dissertação.

Referências

[AD05] Farhad Ameri e Deba Dutta. Product Lifecycle Management : Closing the Knowledge Loops. *Computer Aided Design And Applications*, 2(5):577–590, 2005. URL: http://www.cadanda.com/V2N5_01.pdf.

[Ada] David Adams. How Financial Services can use Product Lifecycle Management to ensure regulatory compliance.

[BKL09] Steven Bird, Ewan Klein e Edward Loper. *Natural language processing with Python*. O'reilly, 2009.

[CHL07] Maxime Crochemore, Christophe Hancart e Thierry Lecroq. *Algorithms on strings*. Cambridge University Press, 2007.

[ddS12] David de Almeida Ferreira e Alberto Rodrigues da Silva. RSLingo: An information extraction approach toward formal requirements specifications. *2012 Second IEEE International Workshop on Model-Driven Requirements Engineering (MoDRE)*, pages 39–48, September 2012. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6360073, doi:10.1109/MoDRE.2012.6360073.

[GCA+10] Valentina Gecevska, Paolo Chiabert, Zoran Anisic, Franco Lombardi e Franc Cus. Product lifecycle management through innovative and competitive business environment. *Journal of Industrial Engineering and Management*, 3(2):323–336, October 2010. URL: http://jiem.org/index.php/jiem/article/view/266, doi:10.3926/jiem.2010.v3n2.p323-336.

[GW98] Robert Gaizauskas e Yorick Wilks. Information extraction: Beyond document retrieval. *Journal of documentation*, 54(1):70–105, 1998.

[Lis13] D E Lisboa. RSLingo : a Formal Requirements Specification Approach based on Linguistic Patterns Tese Provis . (March), 2013.

[Min05] X. G. Ming. Technology Solutions for Collaborative Product Lifecycle Management - Status Review and Future Trend. *Concurrent Engineering*, 13(4):311–319, December 2005. URL: http://cer.sagepub.com/cgi/doi/10.1177/1063293X05060135, doi:10.1177/1063293X05060135.

[SBW+] This Standard, I S Being, Offered Without, A N Y Warranty, Express O R Implied, I N Particular, A N Y Warranty, O F Merchantability, Particular Purpose, Neither Acord, N O R Any, O F Its, Participants O R Submitters, Shall Have, A N Y Liability, Whatsoever To, A N Y Implementer, O R Third, Party For, A N Y Damages, O F Any, Nature Whatsoever e Directly O R Indirectly. *ACORD Property and Casualty Transaction Message Specifications Quick Links (Aggregate / Complex Elements)*.

REFERÊNCIAS

Anexo A

Fase de Processamento de Linguagem Natural

A.1 Resultados da fase de processamento de linguagem natural

```
Reading POS tagger model from edu/stanford/nlp/models/pos-tagger/
    english-left3words/english-left3words-distsim.tagger ... done
    [4.1 sec].
Adding annotator lemma
Adding annotator ner
Loading classifier from edu/stanford/nlp/models/ner/english.all.3
    class.distsim.crf.ser.gz ... done [9.4 sec].
Loading classifier from edu/stanford/nlp/models/ner/english.muc.7
    class.distsim.crf.ser.gz ... done [11.9 sec].
Loading classifier from edu/stanford/nlp/models/ner/english.conll.4
    class.distsim.crf.ser.gz ... done [3.8 sec].
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/defs.
    sutime.txt
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/
    english.sutime.txt
Ignoring inactive rule: null
Ignoring inactive rule: temporal-composite-8:ranges
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/
    english.holidays.sutime.txt
Initializing JollyDayHoliday for sutime with classpath:edu/stanford/
    nlp/models/sutime/jollyday/Holidays_sutime.xml
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/defs.
    sutime.txt
```

```
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/english.sutime.txt
Ignoring inactive rule: null
Ignoring inactive rule: temporal-composite-8:ranges
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/english.holidays.sutime.txt
Adding annotator parse
Loading parser from serialized file edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz ... done [1.6 sec].

Ready to process: 1 files, skipped 0, total 1
Processing file C:\Users\mario\workspace\com.new.nlp\insurance_1.txt ... writing to C:\Users\mario\workspace\com.new.nlp\insurance_1.txt.xml {
 Annotating file C:\Users\mario\workspace\com.new.nlp\insurance_1.txt {
  Untokenizable: ? (U+FFFD, decimal: 65533)
 } [2.915 seconds]
} [3.102 seconds]
Processed 1 documents
Skipped 0 documents, error annotating 0 documents
Annotation pipeline timing information:
PTBTokenizerAnnotator: 0.1 sec.
WordsToSentencesAnnotator: 0.0 sec.
POSTaggerAnnotator: 0.1 sec.
MorphaAnnotator: 0.1 sec.
NERCombinerAnnotator: 1.2 sec.
ParserAnnotator: 1.5 sec.
TOTAL: 2.9 sec. for 67 tokens at 23.0 tokens/sec.
Pipeline setup: 37.8 sec.
Total time for StanfordCoreNLP pipeline: 41.1 sec.
```

Listing A.1: Resultados após processamento do **Caso 1**

```

Adding annotator tokenize
Adding annotator ssplit
Adding annotator pos
```

```
Reading POS tagger model from edu/stanford/nlp/models/pos-tagger/
    english-left3words/english-left3words-distsim.tagger ... done
    [3.4 sec].
Adding annotator lemma
Adding annotator ner
Loading classifier from edu/stanford/nlp/models/ner/english.all.3
    class.distsim.crf.ser.gz ... done [8.4 sec].
Loading classifier from edu/stanford/nlp/models/ner/english.muc.7
    class.distsim.crf.ser.gz ... done [5.9 sec].
Loading classifier from edu/stanford/nlp/models/ner/english.conll.4
    class.distsim.crf.ser.gz ... done [10.1 sec].
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/defs.
    sutime.txt
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/
    english.sutime.txt
Ignoring inactive rule: null
Ignoring inactive rule: temporal-composite-8:ranges
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/
    english.holidays.sutime.txt
Initializing JollyDayHoliday for sutime with classpath:edu/stanford/
    nlp/models/sutime/jollyday/Holidays_sutime.xml
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/defs.
    sutime.txt
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/
    english.sutime.txt
Ignoring inactive rule: null
Ignoring inactive rule: temporal-composite-8:ranges
Reading TokensRegex rules from edu/stanford/nlp/models/sutime/
    english.holidays.sutime.txt
Adding annotator parse
Loading parser from serialized file edu/stanford/nlp/models/
    lexparser/englishPCFG.ser.gz ... done [2.2 sec].

Ready to process: 1 files, skipped 0, total 1
Processing file C:\Users\mario\workspace\com.new.nlp\insurance_2.txt
     ... writing to C:\Users\mario\workspace\com.new.nlp\insurance_2.
    txt.xml {
 Annotating file C:\Users\mario\workspace\com.new.nlp\insurance_2.
     txt {
   Untokenizable: ? (U+FFFD, decimal: 65533)
```

```
  } [14.42 seconds]
} [14.547 seconds]
Processed 1 documents
Skipped 0 documents, error annotating 0 documents
Annotation pipeline timing information:
PTBTokenizerAnnotator: 0.1 sec.
WordsToSentencesAnnotator: 0.0 sec.
POSTaggerAnnotator: 0.1 sec.
MorphaAnnotator: 0.1 sec.
NERCombinerAnnotator: 10.9 sec.
ParserAnnotator: 2.8 sec.
TOTAL: 14.0 sec. for 139 tokens at 9.9 tokens/sec.
Pipeline setup: 51.3 sec.
Total time for StanfordCoreNLP pipeline: 66.0 sec.
```

Listing A.2: Resultados após processamento do **Caso 2**

Anexo B

Resultados em XML

B.1 XML resultante do processamento de linguagem natural do Caso 1

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="CoreNLP-to-HTML.xsl" type="text/xsl"?>
<root>
  <document>
    <sentences>
      <sentence id="1">
        <tokens>
          <token id="1">
            <word>The</word>
            <lemma>the</lemma>
            <CharacterOffsetBegin>2</CharacterOffsetBegin>
            <CharacterOffsetEnd>5</CharacterOffsetEnd>
            <POS>DT</POS>
            <NER>O</NER>
          </token>
          <token id="2">
            <word>product</word>
            <lemma>product</lemma>
            <CharacterOffsetBegin>6</CharacterOffsetBegin>
            <CharacterOffsetEnd>13</CharacterOffsetEnd>
            <POS>NN</POS>
            <NER>O</NER>
          </token>
          <token id="3">
```

```
25          <word>Car</word>
26          <lemma>car</lemma>
27          <CharacterOffsetBegin>14</CharacterOffsetBegin>
28          <CharacterOffsetEnd>17</CharacterOffsetEnd>
29          <POS>NN</POS>
30          <NER>O</NER>
31        </token>
32        <token id="4">
33          <word>Insurance</word>
34          <lemma>insurance</lemma>
35          <CharacterOffsetBegin>18</CharacterOffsetBegin>
36          <CharacterOffsetEnd>27</CharacterOffsetEnd>
37          <POS>NN</POS>
38          <NER>O</NER>
39        </token>
40        <token id="5">
41          <word>is</word>
42          <lemma>be</lemma>
43          <CharacterOffsetBegin>28</CharacterOffsetBegin>
44          <CharacterOffsetEnd>30</CharacterOffsetEnd>
45          <POS>VBZ</POS>
46          <NER>O</NER>
47        </token>
48        <token id="6">
49          <word>composed</word>
50          <lemma>compose</lemma>
51          <CharacterOffsetBegin>31</CharacterOffsetBegin>
52          <CharacterOffsetEnd>39</CharacterOffsetEnd>
53          <POS>VBN</POS>
54          <NER>O</NER>
55        </token>
56        <token id="7">
57          <word>by</word>
58          <lemma>by</lemma>
59          <CharacterOffsetBegin>40</CharacterOffsetBegin>
60          <CharacterOffsetEnd>42</CharacterOffsetEnd>
61          <POS>IN</POS>
62          <NER>O</NER>
63        </token>
64        <token id="8">
```

```
65          <word>the</word>
66          <lemma>the</lemma>
67          <CharacterOffsetBegin>43</CharacterOffsetBegin>
68          <CharacterOffsetEnd>46</CharacterOffsetEnd>
69          <POS>DT</POS>
70          <NER>O</NER>
71        </token>
72        <token id="9">
73          <word>Liability</word>
74          <lemma>Liability</lemma>
75          <CharacterOffsetBegin>47</CharacterOffsetBegin>
76          <CharacterOffsetEnd>56</CharacterOffsetEnd>
77          <POS>NNP</POS>
78          <NER>O</NER>
79        </token>
80        <token id="10">
81          <word>Coverage</word>
82          <lemma>Coverage</lemma>
83          <CharacterOffsetBegin>57</CharacterOffsetBegin>
84          <CharacterOffsetEnd>65</CharacterOffsetEnd>
85          <POS>NNP</POS>
86          <NER>O</NER>
87        </token>
88        <token id="11">
89          <word>and</word>
90          <lemma>and</lemma>
91          <CharacterOffsetBegin>66</CharacterOffsetBegin>
92          <CharacterOffsetEnd>69</CharacterOffsetEnd>
93          <POS>CC</POS>
94          <NER>O</NER>
95        </token>
96        <token id="12">
97          <word>self</word>
98          <lemma>self</lemma>
99          <CharacterOffsetBegin>70</CharacterOffsetBegin>
100         <CharacterOffsetEnd>74</CharacterOffsetEnd>
101         <POS>NN</POS>
102         <NER>O</NER>
103       </token>
104       <token id="13">
```

```
105           <word>damage</word>
106           <lemma>damage</lemma>
107           <CharacterOffsetBegin>75</CharacterOffsetBegin>
108           <CharacterOffsetEnd>81</CharacterOffsetEnd>
109           <POS>NN</POS>
110           <NER>O</NER>
111         </token>
112         <token id="14">
113           <word>.</word>
114           <lemma>.</lemma>
115           <CharacterOffsetBegin>81</CharacterOffsetBegin>
116           <CharacterOffsetEnd>82</CharacterOffsetEnd>
117           <POS>.</POS>
118           <NER>O</NER>
119         </token>
120       </tokens>
121       <parse>(ROOT
122         (S
123         (NP (DT The) (NN product) (NN Car) (NN Insurance))
124         (VP (VBZ is)
125         (VP (VBN composed)
126         (PP (IN by)
127         (NP
128         (NP (DT the) (NNP Liability) (NNP Coverage))
129         (CC and)
130         (NP (NN self) (NN damage))))))
131         (. .)))
132
133         </parse>
134       <dependencies type="basic-dependencies">
135         <dep type="root">
136           <governor idx="0">ROOT</governor>
137           <dependent idx="6">composed</dependent>
138         </dep>
139         <dep type="det">
140           <governor idx="4">Insurance</governor>
141           <dependent idx="1">The</dependent>
142         </dep>
143         <dep type="nn">
144           <governor idx="4">Insurance</governor>
```

```
145           <dependent idx="2">product</dependent>
146         </dep>
147         <dep type="nn">
148           <governor idx="4">Insurance</governor>
149           <dependent idx="3">Car</dependent>
150         </dep>
151         <dep type="nsubjpass">
152           <governor idx="6">composed</governor>
153           <dependent idx="4">Insurance</dependent>
154         </dep>
155         <dep type="auxpass">
156           <governor idx="6">composed</governor>
157           <dependent idx="5">is</dependent>
158         </dep>
159         <dep type="prep">
160           <governor idx="6">composed</governor>
161           <dependent idx="7">by</dependent>
162         </dep>
163         <dep type="det">
164           <governor idx="10">Coverage</governor>
165           <dependent idx="8">the</dependent>
166         </dep>
167         <dep type="nn">
168           <governor idx="10">Coverage</governor>
169           <dependent idx="9">Liability</dependent>
170         </dep>
171         <dep type="pobj">
172           <governor idx="7">by</governor>
173           <dependent idx="10">Coverage</dependent>
174         </dep>
175         <dep type="cc">
176           <governor idx="10">Coverage</governor>
177           <dependent idx="11">and</dependent>
178         </dep>
179         <dep type="nn">
180           <governor idx="13">damage</governor>
181           <dependent idx="12">self</dependent>
182         </dep>
183         <dep type="conj">
184           <governor idx="10">Coverage</governor>
```

```
185           <dependent idx="13">damage</dependent>
186         </dep>
187       </dependencies>
188       <dependencies type="collapsed-dependencies">
189         <dep type="root">
190           <governor idx="0">ROOT</governor>
191           <dependent idx="6">composed</dependent>
192         </dep>
193         <dep type="det">
194           <governor idx="4">Insurance</governor>
195           <dependent idx="1">The</dependent>
196         </dep>
197         <dep type="nn">
198           <governor idx="4">Insurance</governor>
199           <dependent idx="2">product</dependent>
200         </dep>
201         <dep type="nn">
202           <governor idx="4">Insurance</governor>
203           <dependent idx="3">Car</dependent>
204         </dep>
205         <dep type="nsubjpass">
206           <governor idx="6">composed</governor>
207           <dependent idx="4">Insurance</dependent>
208         </dep>
209         <dep type="auxpass">
210           <governor idx="6">composed</governor>
211           <dependent idx="5">is</dependent>
212         </dep>
213         <dep type="det">
214           <governor idx="10">Coverage</governor>
215           <dependent idx="8">the</dependent>
216         </dep>
217         <dep type="nn">
218           <governor idx="10">Coverage</governor>
219           <dependent idx="9">Liability</dependent>
220         </dep>
221         <dep type="agent">
222           <governor idx="6">composed</governor>
223           <dependent idx="10">Coverage</dependent>
224         </dep>
```

```
      <dep type="nn">
        <governor idx="13">damage</governor>
        <dependent idx="12">self</dependent>
      </dep>
      <dep type="conj_and">
        <governor idx="10">Coverage</governor>
        <dependent idx="13">damage</dependent>
      </dep>
    </dependencies>
    <dependencies type="collapsed-ccprocessed-dependencies">
      <dep type="root">
        <governor idx="0">ROOT</governor>
        <dependent idx="6">composed</dependent>
      </dep>
      <dep type="det">
        <governor idx="4">Insurance</governor>
        <dependent idx="1">The</dependent>
      </dep>
      <dep type="nn">
        <governor idx="4">Insurance</governor>
        <dependent idx="2">product</dependent>
      </dep>
      <dep type="nn">
        <governor idx="4">Insurance</governor>
        <dependent idx="3">Car</dependent>
      </dep>
      <dep type="nsubjpass">
        <governor idx="6">composed</governor>
        <dependent idx="4">Insurance</dependent>
      </dep>
      <dep type="auxpass">
        <governor idx="6">composed</governor>
        <dependent idx="5">is</dependent>
      </dep>
      <dep type="det">
        <governor idx="10">Coverage</governor>
        <dependent idx="8">the</dependent>
      </dep>
      <dep type="nn">
        <governor idx="10">Coverage</governor>
```

```
265           <dependent idx="9">Liability</dependent>
266         </dep>
267         <dep type="agent">
268           <governor idx="6">composed</governor>
269           <dependent idx="10">Coverage</dependent>
270         </dep>
271         <dep type="nn">
272           <governor idx="13">damage</governor>
273           <dependent idx="12">self</dependent>
274         </dep>
275         <dep type="agent">
276           <governor idx="6">composed</governor>
277           <dependent idx="13">damage</dependent>
278         </dep>
279         <dep type="conj_and">
280           <governor idx="10">Coverage</governor>
281           <dependent idx="13">damage</dependent>
282         </dep>
283       </dependencies>
284     </sentence>
285
286   </sentences>
287  </document>
288 </root>
```

B.2 XML resultante do processamento de linguagem natural do Caso 2

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet href="CoreNLP-to-HTML.xsl" type="text/xsl"?>
3 <root>
4  <document>
5   <sentences>
6     <sentence id="1">
7       <tokens>
8         <token id="1">
9           <word>The</word>
```

```
10          <lemma>the</lemma>
11          <CharacterOffsetBegin>2</CharacterOffsetBegin>
12          <CharacterOffsetEnd>5</CharacterOffsetEnd>
13          <POS>DT</POS>
14          <NER>O</NER>
15        </token>
16        <token id="2">
17          <word>car</word>
18          <lemma>car</lemma>
19          <CharacterOffsetBegin>6</CharacterOffsetBegin>
20          <CharacterOffsetEnd>9</CharacterOffsetEnd>
21          <POS>NN</POS>
22          <NER>O</NER>
23        </token>
24        <token id="3">
25          <word>insurance</word>
26          <lemma>insurance</lemma>
27          <CharacterOffsetBegin>10</CharacterOffsetBegin>
28          <CharacterOffsetEnd>19</CharacterOffsetEnd>
29          <POS>NN</POS>
30          <NER>O</NER>
31        </token>
32        <token id="4">
33          <word>product</word>
34          <lemma>product</lemma>
35          <CharacterOffsetBegin>20</CharacterOffsetBegin>
36          <CharacterOffsetEnd>27</CharacterOffsetEnd>
37          <POS>NN</POS>
38          <NER>O</NER>
39        </token>
40        <token id="5">
41          <word>is</word>
42          <lemma>be</lemma>
43          <CharacterOffsetBegin>28</CharacterOffsetBegin>
44          <CharacterOffsetEnd>30</CharacterOffsetEnd>
45          <POS>VBZ</POS>
46          <NER>O</NER>
47        </token>
48        <token id="6">
49          <word>composed</word>
```

```
50          <lemma>compose</lemma>
51          <CharacterOffsetBegin>31</CharacterOffsetBegin>
52          <CharacterOffsetEnd>39</CharacterOffsetEnd>
53          <POS>VBN</POS>
54          <NER>O</NER>
55        </token>
56        <token id="7">
57          <word>by</word>
58          <lemma>by</lemma>
59          <CharacterOffsetBegin>40</CharacterOffsetBegin>
60          <CharacterOffsetEnd>42</CharacterOffsetEnd>
61          <POS>IN</POS>
62          <NER>O</NER>
63        </token>
64        <token id="8">
65          <word>Liability</word>
66          <lemma>liability</lemma>
67          <CharacterOffsetBegin>43</CharacterOffsetBegin>
68          <CharacterOffsetEnd>52</CharacterOffsetEnd>
69          <POS>NN</POS>
70          <NER>O</NER>
71        </token>
72        <token id="9">
73          <word>coverages</word>
74          <lemma>coverage</lemma>
75          <CharacterOffsetBegin>53</CharacterOffsetBegin>
76          <CharacterOffsetEnd>62</CharacterOffsetEnd>
77          <POS>NNS</POS>
78          <NER>O</NER>
79        </token>
80        <token id="10">
81          <word>,</word>
82          <lemma>,</lemma>
83          <CharacterOffsetBegin>62</CharacterOffsetBegin>
84          <CharacterOffsetEnd>63</CharacterOffsetEnd>
85          <POS>,</POS>
86          <NER>O</NER>
87        </token>
88        <token id="11">
89          <word>own</word>
```

```
90          <lemma>own</lemma>
91          <CharacterOffsetBegin>64</CharacterOffsetBegin>
92          <CharacterOffsetEnd>67</CharacterOffsetEnd>
93          <POS>JJ</POS>
94          <NER>O</NER>
95        </token>
96        <token id="12">
97          <word>damage</word>
98          <lemma>damage</lemma>
99          <CharacterOffsetBegin>68</CharacterOffsetBegin>
100         <CharacterOffsetEnd>74</CharacterOffsetEnd>
101         <POS>NN</POS>
102         <NER>O</NER>
103       </token>
104       <token id="13">
105         <word>and</word>
106         <lemma>and</lemma>
107         <CharacterOffsetBegin>75</CharacterOffsetBegin>
108         <CharacterOffsetEnd>78</CharacterOffsetEnd>
109         <POS>CC</POS>
110         <NER>O</NER>
111       </token>
112       <token id="14">
113         <word>theft</word>
114         <lemma>theft</lemma>
115         <CharacterOffsetBegin>79</CharacterOffsetBegin>
116         <CharacterOffsetEnd>84</CharacterOffsetEnd>
117         <POS>NN</POS>
118         <NER>O</NER>
119       </token>
120       <token id="15">
121         <word>.</word>
122         <lemma>.</lemma>
123         <CharacterOffsetBegin>84</CharacterOffsetBegin>
124         <CharacterOffsetEnd>85</CharacterOffsetEnd>
125         <POS>.</POS>
126         <NER>O</NER>
127       </token>
128     </tokens>
129     <parse>(ROOT
```

```
(S
(NP (DT The) (NN car) (NN insurance) (NN product))
(VP (VBZ is)
(VP (VBN composed)
(PP (IN by)
(NP
(NP (NN Liability) (NNS coverages))
(, ,)
(NP (JJ own) (NN damage))
(CC and)
(NP (NN theft))))))
(. .)))

</parse>
<dependencies type="basic-dependencies">
<dep type="root">
<governor idx="0">ROOT</governor>
<dependent idx="6">composed</dependent>
</dep>
<dep type="det">
<governor idx="4">product</governor>
<dependent idx="1">The</dependent>
</dep>
<dep type="nn">
<governor idx="4">product</governor>
<dependent idx="2">car</dependent>
</dep>
<dep type="nn">
<governor idx="4">product</governor>
<dependent idx="3">insurance</dependent>
</dep>
<dep type="nsubjpass">
<governor idx="6">composed</governor>
<dependent idx="4">product</dependent>
</dep>
<dep type="auxpass">
<governor idx="6">composed</governor>
<dependent idx="5">is</dependent>
</dep>
<dep type="prep">
```

```
170           <governor idx="6">composed</governor>
171           <dependent idx="7">by</dependent>
172         </dep>
173         <dep type="nn">
174           <governor idx="9">coverages</governor>
175           <dependent idx="8">Liability</dependent>
176         </dep>
177         <dep type="pobj">
178           <governor idx="7">by</governor>
179           <dependent idx="9">coverages</dependent>
180         </dep>
181         <dep type="amod">
182           <governor idx="12">damage</governor>
183           <dependent idx="11">own</dependent>
184         </dep>
185         <dep type="conj">
186           <governor idx="9">coverages</governor>
187           <dependent idx="12">damage</dependent>
188         </dep>
189         <dep type="cc">
190           <governor idx="9">coverages</governor>
191           <dependent idx="13">and</dependent>
192         </dep>
193         <dep type="conj">
194           <governor idx="9">coverages</governor>
195           <dependent idx="14">theft</dependent>
196         </dep>
197       </dependencies>
198       <dependencies type="collapsed-dependencies">
199         <dep type="root">
200           <governor idx="0">ROOT</governor>
201           <dependent idx="6">composed</dependent>
202         </dep>
203         <dep type="det">
204           <governor idx="4">product</governor>
205           <dependent idx="1">The</dependent>
206         </dep>
207         <dep type="nn">
208           <governor idx="4">product</governor>
209           <dependent idx="2">car</dependent>
```

```
      </dep>
      <dep type="nn">
        <governor idx="4">product</governor>
        <dependent idx="3">insurance</dependent>
      </dep>
      <dep type="nsubjpass">
        <governor idx="6">composed</governor>
        <dependent idx="4">product</dependent>
      </dep>
      <dep type="auxpass">
        <governor idx="6">composed</governor>
        <dependent idx="5">is</dependent>
      </dep>
      <dep type="nn">
        <governor idx="9">coverages</governor>
        <dependent idx="8">Liability</dependent>
      </dep>
      <dep type="agent">
        <governor idx="6">composed</governor>
        <dependent idx="9">coverages</dependent>
      </dep>
      <dep type="amod">
        <governor idx="12">damage</governor>
        <dependent idx="11">own</dependent>
      </dep>
      <dep type="conj_and">
        <governor idx="9">coverages</governor>
        <dependent idx="12">damage</dependent>
      </dep>
      <dep type="conj_and">
        <governor idx="9">coverages</governor>
        <dependent idx="14">theft</dependent>
      </dep>
    </dependencies>
    <dependencies type="collapsed-ccprocessed-dependencies">
      <dep type="root">
        <governor idx="0">ROOT</governor>
        <dependent idx="6">composed</dependent>
      </dep>
      <dep type="det">
```

```
250           <governor idx="4">product</governor>
251           <dependent idx="1">The</dependent>
252         </dep>
253         <dep type="nn">
254           <governor idx="4">product</governor>
255           <dependent idx="2">car</dependent>
256         </dep>
257         <dep type="nn">
258           <governor idx="4">product</governor>
259           <dependent idx="3">insurance</dependent>
260         </dep>
261         <dep type="nsubjpass">
262           <governor idx="6">composed</governor>
263           <dependent idx="4">product</dependent>
264         </dep>
265         <dep type="auxpass">
266           <governor idx="6">composed</governor>
267           <dependent idx="5">is</dependent>
268         </dep>
269         <dep type="nn">
270           <governor idx="9">coverages</governor>
271           <dependent idx="8">Liability</dependent>
272         </dep>
273         <dep type="agent">
274           <governor idx="6">composed</governor>
275           <dependent idx="9">coverages</dependent>
276         </dep>
277         <dep type="amod">
278           <governor idx="12">damage</governor>
279           <dependent idx="11">own</dependent>
280         </dep>
281         <dep type="agent">
282           <governor idx="6">composed</governor>
283           <dependent idx="12">damage</dependent>
284         </dep>
285         <dep type="conj_and">
286           <governor idx="9">coverages</governor>
287           <dependent idx="12">damage</dependent>
288         </dep>
289         <dep type="agent">
```

```
290           <governor idx="6">composed</governor>
291           <dependent idx="14">theft</dependent>
292         </dep>
293         <dep type="conj_and">
294           <governor idx="9">coverages</governor>
295           <dependent idx="14">theft</dependent>
296         </dep>
297       </dependencies>
298     </sentence>
299   </sentences>
300  </document>
301 </root>
```

Anexo C

UML do *schema* de saída

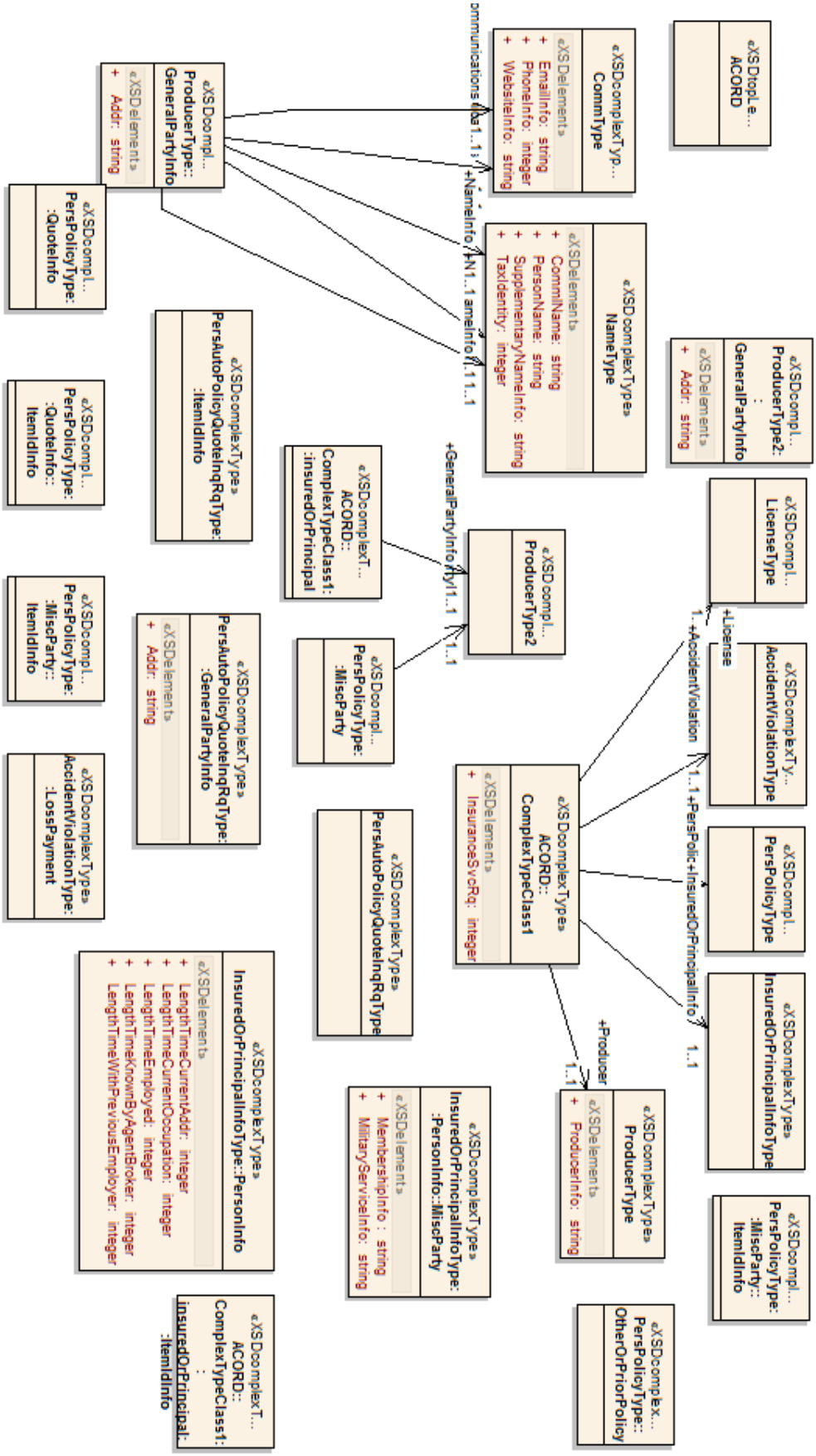


Figura C.1: UML do schema de saída

Anexo D

XML segundo standards ACORD

D.1 XML ACORD

```
1   <?xml version="1.0" encoding="UTF-8"?>
2  <ACORD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="file:/C:/Users/mario/Desktop/
        schema__.xsd">
4     <InsuranceSvcRq>0</InsuranceSvcRq>
5     <Producer>
6        <GeneralPartyInfo>
7           <NameInfo>
8              <CommlName>CommlName0</CommlName>
9              <PersonName>PersonName0</PersonName>
10             <TaxIdentity>0</TaxIdentity>
11             <SupplementaryNameInfo>SupplementaryNameInfo0</
                   SupplementaryNameInfo>
12          </NameInfo>
13          <Addr>Addr0</Addr>
14          <Communications>
15             <PhoneInfo>0</PhoneInfo>
16             <EmailInfo>EmailInfo0</EmailInfo>
17             <WebsiteInfo>WebsiteInfo0</WebsiteInfo>
18          </Communications>
19       </GeneralPartyInfo>
20       <ProducerInfo>ProducerInfo0</ProducerInfo>
21    </Producer>
22    <insuredOrPrincipal>
23       <ItemIdInfo>
```

```
24          <OtherIdentifier>
25          </OtherIdentifier>
26       </ItemIdInfo>
27       <GeneralPartyInfo>
28          <GeneralPartyInfo>
29             <NameInfo>
30                <CommlName>CommlName1</CommlName>
31                <PersonName>PersonName1</PersonName>
32                <TaxIdentity>0</TaxIdentity>
33                <SupplementaryNameInfo>SupplementaryNameInfo1</
                     SupplementaryNameInfo>
34             </NameInfo>
35             <Addr>Addr1</Addr>
36             <Communications>
37                <PhoneInfo>0</PhoneInfo>
38                <EmailInfo>EmailInfo1</EmailInfo>
39                <WebsiteInfo>WebsiteInfo1</WebsiteInfo>
40             </Communications>
41          </GeneralPartyInfo>
42       </GeneralPartyInfo>
43    </insuredOrPrincipal>
44    <InsuredOrPrincipalInfo>
45       <PersonInfo>
46          <LengthTimeEmployed>0</LengthTimeEmployed>
47          <LengthTimeCurrentOccupation>0</LengthTimeCurrentOccupation
               >
48          <LengthTimeWithPreviousEmployer>0</
               LengthTimeWithPreviousEmployer>
49          <LengthTimeCurrentAddr>0</LengthTimeCurrentAddr>
50          <MiscParty>
51             <ItemIdInfo>
52                <OtherIdentifier>
53                </OtherIdentifier>
54             </ItemIdInfo>
55             <GeneralPartyInfo>
56                <GeneralPartyInfo>
57                   <NameInfo>
58                      <CommlName>CommlName2</CommlName>
59                      <PersonName>PersonName2</PersonName>
60                      <TaxIdentity>0</TaxIdentity>
```

```
61                      <SupplementaryNameInfo>SupplementaryNameInfo2</
                           SupplementaryNameInfo>
62                   </NameInfo>
63                   <Addr>Addr2</Addr>
64                   <Communications>
65                      <PhoneInfo>0</PhoneInfo>
66                      <EmailInfo>EmailInfo2</EmailInfo>
67                      <WebsiteInfo>WebsiteInfo2</WebsiteInfo>
68                   </Communications>
69                </GeneralPartyInfo>
70             </GeneralPartyInfo>
71             <MiscPartyInfo>
72             </MiscPartyInfo>
73             <MembershipInfo>MembershipInfo0</MembershipInfo>
74             <MilitaryServiceInfo>MilitaryServiceInfo0</
                  MilitaryServiceInfo>
75          </MiscParty>
76          <LengthTimeKnownByAgentBroker>0</
               LengthTimeKnownByAgentBroker>
77       </PersonInfo>
78    </InsuredOrPrincipalInfo>
79    <PersPolicy>
80       <ContractTerm>
81       </ContractTerm>
82       <TotalPaidLossAmt>
83       </TotalPaidLossAmt>
84       <OtherOrPriorPolicy>
85          <ContractTerm>
86          </ContractTerm>
87          <LengthTimeWithPreviousInsurer>
88          </LengthTimeWithPreviousInsurer>
89          <Coverage>
90          </Coverage>
91       </OtherOrPriorPolicy>
92       <PaymentOption>
93       </PaymentOption>
94       <QuoteInfo>
95          <ItemIdInfo>
96             <OtherIdentifier>
97             </OtherIdentifier>
```

```
98          </ItemIdInfo>
99       </QuoteInfo>
100      <MiscParty>
101         <ItemIdInfo>
102            <OtherIdentifier>
103            </OtherIdentifier>
104         </ItemIdInfo>
105      </MiscParty>
106   </PersPolicy>
107   <AccidentViolation>
108      <DamageTotalAmt>
109      </DamageTotalAmt>
110      <LossPayment>
111         <Coverage>
112         </Coverage>
113         <LossPaymentAmt>
114         </LossPaymentAmt>
115      </LossPayment>
116      <ExcessSpeed>
117      </ExcessSpeed>
118      <ConvictionsDuration>
119      </ConvictionsDuration>
120   </AccidentViolation>
121   <License>
122      <LicenseTerm>
123      </LicenseTerm>
124      <SuspensionTerm>
125      </SuspensionTerm>
126      <RestrictionInfo>
127      </RestrictionInfo>
128   </License>
129</ACORD>
```

D.2 *Schema* ACORD

```
1
2 <?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

   <xs:element name="ACORD" >
      <xs:complexType>
      <xs:sequence>
         <xs:element name = "InsuranceSvcRq" type= "xs:integer"/>
         <xs:element name = "Producer" type = "ProducerType"/>
         <xs:element name = "insuredOrPrincipal" >
            <xs:complexType>
               <xs:sequence>
                  <xs:element name = "ItemIdInfo">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:element name = "OtherIdentifier"/>
                        </xs:sequence>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name = "GeneralPartyInfo" type = "
                     ProducerType2"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name = "InsuredOrPrincipalInfo" type = "
            InsuredOrPrincipalInfoType"/>
         <xs:element name = "PersPolicy" type = "PersPolicyType"/>
         <xs:element name = "AccidentViolation" type = "
            AccidentViolationType"/>
         <xs:element name = "License" type = "LicenseType"/>
      </xs:sequence>
      </xs:complexType>
   </xs:element>

      <xs:complexType name = "NameType">
         <xs:sequence>
            <xs:element name="CommlName" type="xs:string"/>
            <xs:element name="PersonName" type="xs:string"/>
            <xs:element name="TaxIdentity" type="xs:integer"/>
            <xs:element name="SupplementaryNameInfo" type="xs:string
               "/>
```

```
38          </xs:sequence>
39       </xs:complexType>
40
41
42
43       <xs:complexType name= "CommType">
44          <xs:sequence>
45             <xs:element name="PhoneInfo" type="xs:integer"/>
46             <xs:element name="EmailInfo" type="xs:string"/>
47             <xs:element name="WebsiteInfo" type="xs:string"/>
48          </xs:sequence>
49       </xs:complexType>
50
51
52    <xs:complexType name ="ProducerType">
53       <xs:sequence>
54          <xs:element name="GeneralPartyInfo" >
55             <xs:complexType>
56                <xs:sequence>
57                   <xs:element name="NameInfo" type = "NameType"/>
58                   <xs:element name="Addr" type="xs:string" />
59                   <xs:element name="Communications" type="CommType"/
                        >
60
61                </xs:sequence>
62             </xs:complexType>
63          </xs:element>
64          <xs:element name="ProducerInfo" type="xs:string"/>
65       </xs:sequence>
66
67    </xs:complexType>
68
69    <xs:complexType name="ProducerType2">
70       <xs:sequence>
71          <xs:element name="GeneralPartyInfo" >
72             <xs:complexType>
73                <xs:sequence>
74                   <xs:element name="NameInfo" type = "NameType"/>
75                   <xs:element name="Addr" type="xs:string" />
```

```
76                  <xs:element name="Communications" type="CommType"/
                        >
77
78               </xs:sequence>
79            </xs:complexType>
80         </xs:element>
81      </xs:sequence>
82   </xs:complexType>
83
84
85   <xs:complexType name="PersAutoPolicyQuoteInqRqType">
86      <xs:sequence>
87         <xs:element name = "ItemIdInfo">
88           <xs:complexType>
89              <xs:sequence>
90                 <xs:element name = "OtherIdentifier"/>
91              </xs:sequence>
92           </xs:complexType>
93         </xs:element>
94         <xs:element name="GeneralPartyInfo" >
95            <xs:complexType>
96               <xs:sequence>
97                  <xs:element name="NameInfo" type = "NameType"/>
98                  <xs:element name="Addr" type="xs:string" />
99                  <xs:element name="Communications" type="CommType"/
                        >
100              </xs:sequence>
101           </xs:complexType>
102        </xs:element>
103     </xs:sequence>
104  </xs:complexType>
105
106  <xs:complexType name="PersPolicyType">
107    <xs:sequence>
108       <xs:element name = "ContractTerm"/>
109       <xs:element name = "TotalPaidLossAmt"/>
110       <xs:element name = "OtherOrPriorPolicy">
111          <xs:complexType>
112             <xs:sequence>
113                <xs:element name = "ContractTerm"/>
```

```
114                  <xs:element name = "LengthTimeWithPreviousInsurer"/
                        >
115                  <xs:element name = "Coverage"/>
116               </xs:sequence>
117            </xs:complexType>
118         </xs:element>
119         <xs:element name = "PaymentOption"/>
120         <xs:element name = "QuoteInfo">
121            <xs:complexType>
122               <xs:sequence>
123                  <xs:element name = "ItemIdInfo">
124                     <xs:complexType>
125                        <xs:sequence>
126                           <xs:element name = "OtherIdentifier"/>
127                        </xs:sequence>
128                     </xs:complexType>
129                  </xs:element>
130               </xs:sequence>
131            </xs:complexType>
132         </xs:element>
133         <xs:element name = "MiscParty">
134            <xs:complexType>
135               <xs:sequence>
136                  <xs:element name = "ItemIdInfo">
137                     <xs:complexType>
138                        <xs:sequence>
139                           <xs:element name = "OtherIdentifier"/>
140                        </xs:sequence>
141                     </xs:complexType>
142                  </xs:element>
143               </xs:sequence>
144            </xs:complexType>
145         </xs:element>
146      </xs:sequence>
147   </xs:complexType>
148
149   <xs:complexType name="AccidentViolationType">
150      <xs:sequence>
151         <xs:element name = "DamageTotalAmt"/>
152         <xs:element name = "LossPayment">
```

```
153             <xs:complexType>
154                <xs:sequence>
155                   <xs:element name = "Coverage"/>
156                   <xs:element name = "LossPaymentAmt"/>
157                </xs:sequence>
158             </xs:complexType>
159          </xs:element>
160          <xs:element name = "ExcessSpeed"/>
161          <xs:element name = "ConvictionsDuration"/>
162       </xs:sequence>
163    </xs:complexType>
164
165    <xs:complexType name="LicenseType">
166       <xs:sequence>
167          <xs:element name = "LicenseTerm"/>
168          <xs:element name = "SuspensionTerm"/>
169          <xs:element name = "RestrictionInfo"/>
170       </xs:sequence>
171    </xs:complexType>
172
173    <xs:complexType name="InsuredOrPrincipalInfoType">
174       <xs:sequence>
175          <xs:element name = "PersonInfo">
176             <xs:complexType>
177                <xs:sequence>
178                   <xs:element name="LengthTimeEmployed" type="xs:
                         integer"/>
179                   <xs:element name="LengthTimeCurrentOccupation"
                         type="xs:integer"/>
180                   <xs:element name="LengthTimeWithPreviousEmployer"
                         type="xs:integer"/>
181                   <xs:element name="LengthTimeCurrentAddr" type="xs:
                         integer"/>
182                   <xs:element name="MiscParty">
183                      <xs:complexType>
184                         <xs:sequence>
185                            <xs:element name = "ItemIdInfo">
186                               <xs:complexType>
187                                  <xs:sequence>
```

```
188                                    <xs:element name = "
                                          OtherIdentifier"/>
189                                 </xs:sequence>
190                              </xs:complexType>
191                           </xs:element>
192                           <xs:element name = "GeneralPartyInfo"
                                 type = "ProducerType2"/>
193                           <xs:element name = "MiscPartyInfo"/>
194                            <xs:element name = "MembershipInfo "
                                  type = "xs:string"/>
195                           <xs:element name = "MilitaryServiceInfo"
                                 type = "xs:string"/>
196                        </xs:sequence>
197                     </xs:complexType>
198                  </xs:element>
199                  <xs:element name= "LengthTimeKnownByAgentBroker"
                        type = "xs:integer"/>
200               </xs:sequence>
201            </xs:complexType>
202         </xs:element>
203      </xs:sequence>
204   </xs:complexType>
205 </xs:schema>
```