Processamento de Linguagem Natural Aplicada à Inspeção de Documentos de Especificação de Requisitos de Software

Luís Felipe Santos Silva

Wladmir Araújo Chapetta

Universidade Federal do Rio de Janeiro

COPPE / Engenharia de Sistemas de Computação

Junho / 2003

RESUMO

Na grande maioria dos processos de desenvolvimento de software, o documento de especificação de requisitos é o primeiro artefato tangível a ser produzido. Sua função principal é descrever todas as características e funções que o software a ser desenvolvido deve possuir, atuando como um contrato entre o cliente e o desenvolvedor. Realizar inspeções neste documento é uma alternativa muito interessante para detectar seus defeitos, aumentando a qualidade final do sistema a ser desenvolvido.

O apoio ferramental à inspeção seria bastante enriquecido com o uso de técnicas de processamento de linguagem natural, como a sumarização automática e a extração de indicadores de qualidade.

I - Introdução

I.1 Caracterização do Problema:

Na grande maioria dos processos de desenvolvimento de software, o documento de especificação de requisitos é o primeiro artefato tangível a ser produzido. Sua função principal é descrever todas as características e funções que o software a ser desenvolvido deve possuir, atuando como um contrato entre o cliente e o desenvolvedor.

Este documento servirá de base para todas as etapas subseqüentes do desenvolvimento, sendo utilizado pelos diversos participantes do processo (patrocinadores, usuários, projetistas, programadores, testadores, etc). De forma a facilitar seu entendimento por todos estes participantes, ele normalmente é escrito em linguagem natural. É sabido que isto pode trazer uma série de ambigüidades ao seu conteúdo. Permitindo múltiplas interpretações, os modelos produzidos a partir dele podem não contemplar as características que o software deve possuir. Além disso, a própria identificação dos requisitos pode não ser feita de forma correta. Requisitos podem ser omitidos ou, ainda, não expressar corretamente o conhecimento que se tem sobre o domínio da aplicação. Há também a possibilidade de determinados requisitos contradizerem outros, deixando o documento inconsistente.

Na tabela abaixo, temos os tipos de defeitos mais cometidos na identificação e documentação dos requisitos de software.

Tipo de Defeito	Percentagem (%)
Fato Incorreto	40
Omissão	31
Inconsistência	13
Ambigüidade	5
Localização Incorreta	2

Tabela 1: Tipos de defeitos encontrados em requisitos - Adaptado de (DAVIS,1990).

A ocorrência destes defeitos pode ter diversas conseqüências para um projeto:

- As estimativas de esforço e prazo podem deixar de fazer sentido já que seriam feitas com base em requisitos que não retratam as características desejadas para o software;
- Desperdício de recursos ocorreria na medida em que seriam dedicados esforços para implementação de funcionalidades incorretas ou desnecessárias;
- O produto final pode não atender as necessidades do usuário.

Assim, identificar e documentar corretamente os requisitos de um software é um fator crítico de sucesso para um projeto.

Encontramos na literatura diversas diretrizes para atingir este objetivo. O padrão IEEE 830-1998 recomenda diversos preceitos neste sentido. Ele define características de qualidade para um documento de especificação de requisitos bem como recomendações de como atingi-las. Estas características de qualidade, com uma breve definição, estão listadas na tabela abaixo:

Característica de Qualidade	Definição
Correto	Reflete somente a necessidade do usuário.
Não Ambíguo	Passível de apenas uma interpretação.
Completo	Reflete todas as necessidades do usuário.
Consistente	Não possui requisitos conflitantes.
Classificável	Cada requisito possui um indicativo de seu grau de importância ou estabilidade.
Verificável	Usa termos concretos e quantidades mensuráveis.
Modificável	Sua estrutura e estilo permitem que mudanças sejam feitas de forma consistente.
Rastreável	A origem dos requisitos é clara e é possível referenciá-los.

Tabela 2: Características de qualidade para documentos de especificação de requisitos – Adaptado de (IEEE Std 830-1998).

Ainda que diversas diretrizes sejam seguidas, seria no mínimo arriscado assumir que o documento não conterá defeitos. BOEHM e BASILI (2001) realçam que corrigir defeitos após a entrega do produto pode ser até cem vezes mais caro que corrigi-los nas primeiras fases do desenvolvimento (em projetos menores esta relação parece ser de 5:1). Além disso, em projetos recentes de software, teríamos um esforço de retrabalho entre 40% e 50% do esforço total.

Neste contexto, realizar inspeções no documento de especificações de requisitos é uma alternativa muito interessante para detectar seus defeitos e assim minimizar o retrabalho e aumentar a qualidade final do sistema a ser desenvolvido.

Método de Detecção	Erros Encontrados(%)
Inspeção	65
Teste de Unidade	10
Avaliação	10
Integração	5
Outros	10

Tabela 3: Métodos de Detecção de Defeitos - Adaptado de (DAVIS,1990)

I.2 Inspeções de Software:

O termo "Inspeção de Software" foi introduzido por Michael Fagan, na década de 70, em seu artigo "Design and Code Inspections to Reduce Errors in Program Development". Desde então, este tem sido o tipo de revisão de software mais estudado e utilizado (CHRISTIENSEN *et al.*,2001).

Seus principais objetivos são:

- Identificar erros específicos num documento ou sistema;
- Identificar erros sistemáticos no processo de desenvolvimento;
- Identificar desvios em relação a especificações e padrões;

O uso correto do processo de inspeção de software provê ganhos significativos em relação a prazos e custos (GRADY *et al.*,1994). Inspeções de software também tenderiam a achar mais defeitos do que qualquer outro processo, e a um custo menor (GLASS,1999).

I.3 Apoio Ferramental:

Pelo fato da maior parte dos documentos de especificação de requisitos de software ser escrita em linguagem natural, a automatização completa da atividade de inspeção torna-se uma tarefa extremamente complexa, já que muito dos defeitos são encontrados pelo inspetor através do seu conhecimento do domínio.

Entretanto, esta atividade pode ter um apoio ferramental. Uma ferramenta poderia auxiliar o inspetor no entendimento e na aplicação de uma técnica de inspeção. Na verdade, este é o foco de trabalho dos autores desta monografia, especificamente para aplicação da técnica de leitura baseada em perspectiva – PBR.

Acreditamos que a utilização de técnicas de Inteligência Artificial, particularmente Processamento de Linguagem Natural, auxiliariam os inspetores na execução de determinadas atividades de PBR.

I.4 Técnica de Leitura Baseada em Perspectiva – PBR :

SHULL *et al.* (2000) destacam que os desenvolvedores são treinados para escrever artefatos de software, mas raramente possuem habilidades para revisá-lo, quando confiam em técnicas *ad-hoc* e não seguem um procedimento bem definido. Obviamente esta deficiência pode prejudicar a fase mais importante do processo de inspeção de software: a fase de detecção de defeitos.

Uma técnica de leitura de software deve fornecer um conjunto de instruções a serem seguidas pelos revisores para que os defeitos sejam detectados. Este formalismo garante que este procedimento possa ser seguido e repetido, possibilitando a melhoria contínua do processo de revisão.

As técnicas de leitura baseadas em perspectiva (PBR) se baseiam no fato de existirem diferentes papéis durante o processo de desenvolvimento do software. Assim, PBR provê um conjunto de instruções de acordo com o papel do "consumidor" de um artefato (usuário, projetista ou testador). Estas instruções tentam destacar os pontos mais importantes para uma utilização particular do artefato, fornecendo indicações de como identificar defeitos nestes pontos. Tendo um foco mais específico, um inspetor seria muito mais eficiente na busca por defeitos, levando-se em conta a perspectiva adotada. A combinação de diferentes perspectivas promoveria uma maior cobertura de defeitos em comparação com um processo de inspeção *ad-hoc*.

Este é o principal diferencial de PBR. Enquanto as outras técnicas tratam todos os requisitos como sendo igualmente importantes, PBR assume que o grau de importância de um determinado requisito é determinado pela sua perspectiva de utilização.

Num processo de inspeção que utilize PBR como técnica de identificação de defeitos, cada inspetor assume uma perspectiva específica. Sua primeira tarefa é, então, criar um modelo de alto nível de acordo com a perspectiva assumida. Assim, um modelo de casos de uso seria produzido pelo inspetor que assumisse a perspectiva do usuário. Para as perspectivas do projetista e do testador seriam criados, respectivamente, um projeto de alto nível ou um plano com os possíveis casos de teste.

O objetivo da criação destes artefatos de alto nível é permitir que os inspetores possam analisar se o documento de especificação de requisitos atenderá às necessidades de seus futuros "consumidores". Vale também destacar que não é intenção da técnica duplicar o esforço na criação dos artefatos, já que estes modelos de alto nível devem servir de base para a criação de modelos mais detalhados nas fases de desenvolvimento subseqüentes.

Ao criar estes modelos, os inspetores são guiados a encontrar defeitos no documento de especificação de requisitos ao responderem a uma série de perguntas. O objetivo desse procedimento é identificar se as informações contidas no documento permitem que uma representação correta do sistema seja feita. Os requisitos que não possibilitam que estas perguntas sejam respondidas também não conterão as informações esperadas pelo seu futuro "consumidor". Quando isto acontece, uma discrepância é identificada e deve, então, ser categorizada.

Baseado nos atributos de qualidade definidos pelo padrão IEEE 830-1984, uma taxonomia para os defeitos foi definida.

Tipo de Defeito	Definição
Omissão	Informação necessária não incluída.
Ambigüidade	Informação passível de ter múltiplas interpretações.
Inconsistência	Informações conflitantes.
Fato Incorreto	Informação que não é verdadeira para as condições especificadas.
Informação Estranha	Informação desnecessária.

Tabela 4: Tipos de Defeito em Requisitos - Adaptado de (SHULL *et al.*,2000).

Assim, para cada discrepância identificada será associado um ou mais tipos de defeitos. Entretanto, esta taxonomia não pretende ser definitiva e cada organização pode acrescentar mais tipos de acordo com suas necessidades.

I.5 Propostas de Aplicações de PNL para Inspeção:

Algumas técnicas de processamento de linguagem natural poderiam enriquecer muito uma ferramenta como a proposta no item I.3.

Uma possibilidade é a utilização da sumarização de textos para a busca de possíveis atores para os modelos de casos de uso. Este método poderia ainda sugerir o conteúdo que deve ser considerado na descrição destes casos de uso (ver Figura 1). Em PBR, estes devem ser criados como parte da atividade de identificação de discrepâncias, quando a perspectiva do usuário é utilizada.

Uma outra possibilidade seria a análise automática de indicadores de qualidade do documento de especificação de requisitos. Esta tarefa foi feita para documentos especificados na língua inglesa, a partir da análise de uma série de documentos contidos numa base histórica do Goddard Space Flight Center da NASA (Wilson *et. al.*, 1997). O que faremos aqui é uma adaptação deste trabalho para a análise de documentos escritos na língua portuguesa. Seria interessante então comparar os tipos de discrepâncias obtidos como resultado do processo de inspeção com os indicadores de qualidade obtidos a partir do documento inspecionado e tentar encontrar algum relacionamento entre eles. A repetição desse processo nos permitiria a construção de uma base de conhecimento. Usaríamos tal base para, a partir dos indicadores de qualidade de um documento a ser inspecionado, sugerir os tipos de discrepâncias mais prováveis de serem encontrados.

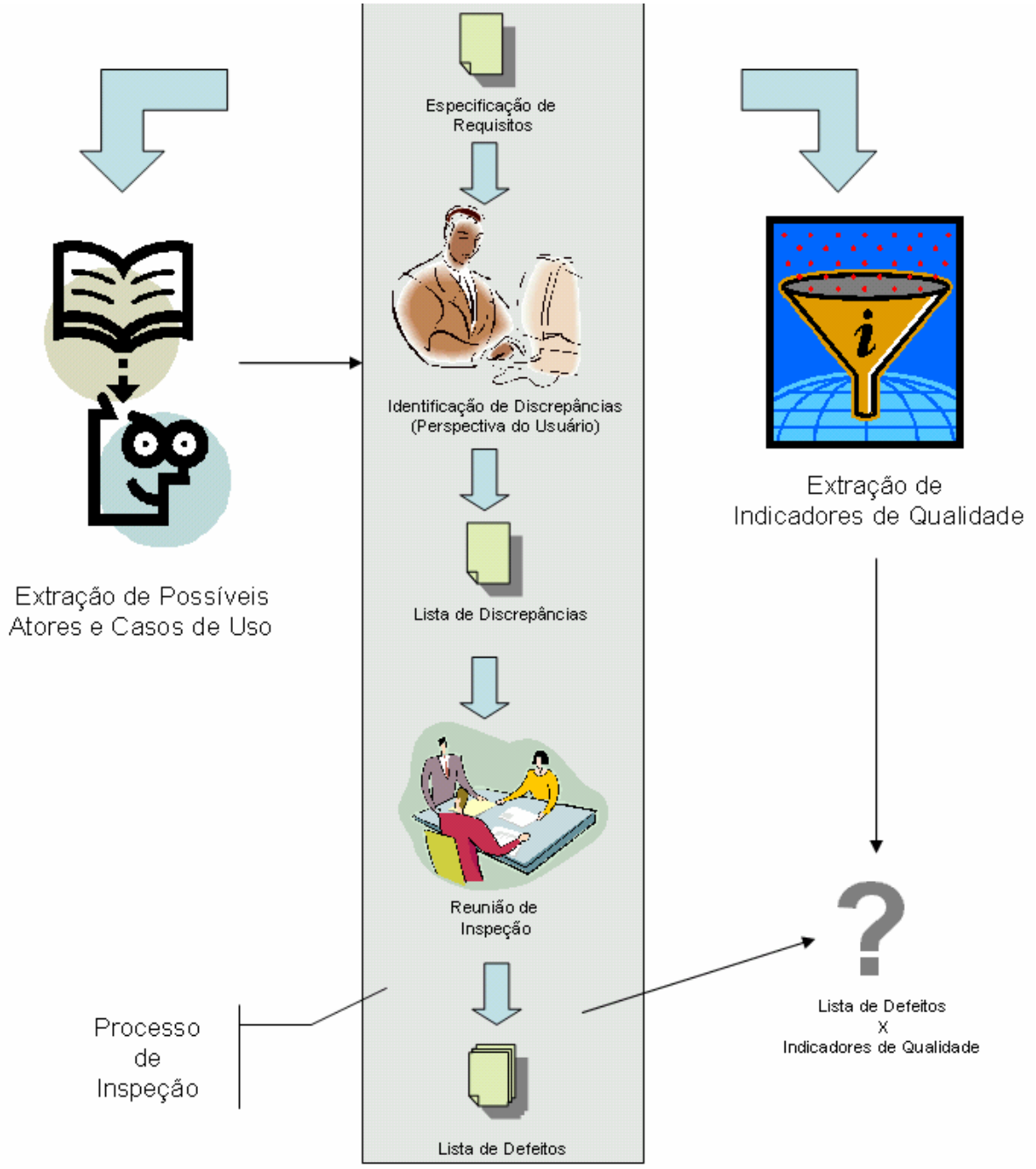

Figura 1: Processamento de Linguagem Natural aplicado à Inspeção de Requisitos de Software

II- Princípios de Processamento de Linguagem Natural (COLE,1995)

II.1 Gramática:

A nossa compreensão de unidades textuais extensas é feita através da compreensão das unidades textuais menores combinadas. O principal objetivo da teoria lingüística é mostrar como essas unidades maiores de compreensão são formadas pelas menores. Isto é modelado através de uma gramática.

A lingüística computacional tenta implementar este processo de uma forma eficiente. Tradicionalmente divide-se esta tarefa em sintaxe e em semântica, onde a sintaxe descreve como podem ser combinados os diferentes elementos de uma unidade textual (uma sentença) e a semântica descreve como a interpretação será feita.

Na maior parte das aplicações, a gramática é isolada dos componentes do software responsáveis pelo processamento. Uma gramática consiste de regras que sintática e semanticamente combinam palavras e frases em sentenças maiores. Tais gramáticas podem atingir um tamanho tal que seja muito difícil mantê-la, estendê-la ou reutilizá-la. Os sistemas resultantes poderiam ser suficientemente eficientes para algumas aplicações, mas perderiam velocidade de processamento. Essa é uma característica essencial para sistemas interativos ou para sistemas que processam grandes volumes de texto.

Nas pesquisas atuais há uma espécie de polarização. Num extremo, modelos de gramática bastante simples são utilizados em conjunto com métodos estatísticos para a busca de padrões lingüísticos. No outro, uma variedade de formalismos altamente sofisticados são utilizados.

II.2 Análise Morfológica:

Para construir uma representação sintática de uma determinada sentença, um *parser* deve mapear cada uma das palavras do texto para uma representação canônica e reconhecer suas propriedades morfológicas. A combinação da forma original com sua análise como uma forma canônica é denominada lema.

Os principais problemas nesta tarefa são:

- Alterações morfológicas: um mesmo morfema pode ser percebido de diferentes formas, dependendo do contexto;
- Radicais, prefixos e sufixos não podem ser combinados livremente, portanto um analisador morfológico precisa saber quais combinações são válidas.

II.3 *Parsers* Superficiais:

O termo sintaxe superficial é usado genericamente para análises menos completas que as análises feitas por um *parser* convencional. Um analisador superficial pode identificar alguns constituintes de uma frase, sem indicar a estruturação interna da mesma e a função que este constituinte exerce na frase.

Uma linha de pesquisa atual se concentra na integração de *parsers* superficiais com uma preocupação sintática maior. A idéia seria utilizar tais *parsers* com pré-processadores para *parsers* mais ousados.

II.4 Formalismos Gramaticais:

As chamadas gramáticas unificadas superam as representações das linguagens anteriores no sentido de ter uma semântica "limpa", que permite a codificação do conhecimento gramatical independente de qualquer algoritmo específico de processamento.

Uma característica desses formalismos é a complexa descrição formal das unidades gramaticais (palavras, frases, sentenças) através de conjuntos de pares atributo-valor, chamados de termos característicos (*feature terms*). Estes termos podem ser aninhados, ou seja, os valores podem conter símbolos atômicos ou outros termos característicos. Estes formalismos compartilham uma operação uniforme para a checagem das informações gramaticais. Este processo é comumente chamado de unificação.

A experiência tem mostrado que gramáticas extensas podem ser especificadas, mas o seu desenvolvimento demanda um esforço muito grande. Algumas experiências alimentam a esperança que estes formalismos possam acelerar este desenvolvimento. Um outro aspecto importante é a reusabilidade de uma gramática. Quanto mais dependente de um modelo de processamento estiver a gramática, menores serão suas chances de adaptação aos novos modelos de processamento ou às novas áreas de aplicação. Ainda que uma representação uniforme esteja bem distante, uma gramática formal facilita a tradução de um modelo para outro.

II.5 Explorando Sentenças:

A complexidade estrutural das sentenças em linguagem natural se manifesta de duas formas. Na primeira, cada constituinte da oração (uma palavra, por exemplo), não ocorre com igual probabilidade em todos os contextos. Além dessa, a informação contida numa sentença depende dos relacionamentos existentes entre os constituintes da sentença.

Todo *parser* tem que incluir uma gramática que especifique como as sentenças serão interpretadas. Podemos pensar esta gramática como um espaço de configurações, onde cada configuração representa estágios da combinação dos constituintes da sentença sendo analisada. Transições entre configurações descrevem como os constituintes são combinados para derivar outros constituintes mais extensos.

Um *parser* seria então representado por um modelo consistindo de uma gramática e um procedimento de busca. As propriedades computacionais dos *parsers* dependem de dois fatores: a estrutura do espaço de busca e a exaustividade do procedimento de busca.

II.5.1 Estrutura do Espaço de Busca:

As transições entre uma configuração e outra teria que levar em conta todas as configurações. Entretanto, a maior parte das classes de gramática possui um certo grau de localidade, e cada transação envolve apenas uma parte da configuração. Assim, derivações podem ser fatoradas em sub-derivações a partir de partes independentes da configuração. O compartilhamento destas sub-derivações potencialmente permite reduções exponenciais no tamanho do espaço de busca.

Um algoritmo de busca pode tabular cada sub-derivação e reutilizá-la na construção de derivações que compartilhem estas sub-derivações. Gramáticas livres de contexto são um exemplo onde estas derivações são tabuladas.

II.5.2 Exaustividade do Procedimento de Busca:

Ainda que a gramática permita a busca tabular, pode não ser computacionalmente viável explorar todo o espaço de busca em razão do efeito do tamanho da gramática no tamanho do espaço de busca. Os modelos finitos para o reconhecimento da fala são um exemplo. Já que cada estado é potencialmente considerado, a computação por palavra reconhecida é pesada demais para a performance necessária em aplicações de tempo-real.

Uma alternativa é explorar técnicas que evitam explorar todo o espaço de busca da gramática. Estas técnicas são divididas em duas classes: poda (*pruning*) e busca admissível.

Nas técnicas de poda, uma função de avaliação aplicada às configurações determina se elas serão expandidas. Já que esta função não pode prever o futuro (para fazer isto deveria explorar todo o espaço de busca), a técnica de poda pode bloquear a derivação correta. A escolha desta função é uma troca entre a redução do espaço de busca (e então das complexidades de espaço e tempo) e o risco de não se realizar a interpretação correta.

Os procedimentos de busca admissíveis ordenam as sub-derivações de forma que as mais prováveis de gerar as derivações completas são consideradas primeiro. A dificuldade passa a ser então definir o critério de ordenação com maior probabilidade de se alcançar as melhores derivações antes de se explorar um conjunto desnecessário de configurações. Neste contexto, os algoritmos do tipo A* podem ser usados já que expandem as configurações de menor custo estimado primeiro. Sendo esta estimativa um limite inferior do custo real, a derivação completa alcançada com um algoritmo do tipo A* garantidamente será achada no menor custo.

III - Sumarização de Textos (MARTINS *et al*, 2001)

III-1 Introdução

A sumarização, em geral, é uma atividade bastante comum. Quando se narra um evento a uma pessoa, costuma-se fazer um resumo do que aconteceu e não uma narração completa e detalhada. Inconscientemente, as pessoas estão sempre sumarizando. Exemplos de sumários escritos incluem notícias de jornais, artigos de revistas, resumo de textos científicos, entre muitos outros.

Por sua utilidade e freqüência, há um grande interesse em automatizar esse processo.

Os sumários produzidos a partir de textos são particularmente úteis, podendo servir como indexadores ou ser autocontidos. No primeiro caso, os sumários são lidos para descobrir qual é o assunto do texto-fonte correspondente e, caso seja de interesse, o leitor remete ao texto completo, para informar-se. No segundo caso, os sumários já são considerados informativos o suficiente e, portanto, o leitor pode dispensar o texto de origem e, ainda assim, situar-se de suas informações principais.

As duas abordagens principais do PLN são a superficial e a profunda, as quais caracterizam métodos distintos de sumarização automática. A abordagem superficial utiliza, sobretudo, métodos experimentais e estatísticos, enquanto a profunda está relacionada a teorias formais e lingüísticas.

Nesta monografia, veremos apenas as principais características a serem consideradas para a sumarização automática superficial.

Quando se fala de sumarização, é necessário referir-se ao que se entende por um sumário. Há três características possíveis na sumarização humana: a variação de conteúdo informativo e a multiplicidade sentencial ou estrutural dos sumários. Logo há a possibilidade de se produzir mais de um sumário para um mesmo texto-fonte. Tais possibilidades levam ao problema de modelá-las de modo adequado, para que os resultados automáticos não percam sua interdependência com os textos-fonte correspondentes.

Além dessas características, várias outras, também apreendidas mediante análise do processo humano de sumarização, irão interferir no projeto e desenvolvimento de sumarizadores automáticos. O importante é notar que:

a) sumários remetem, necessariamente, a eventos ou textos originários dos mesmos e;

b) sumários devem ser construídos tendo em mente que não pode haver perda do significado original, muito embora contenham menos informações e possam apresentar diferentes estruturas, em relação a suas fontes.

Por essas razões, é necessário, primeiramente, que se definam claramente os conceitos básicos relativos ao que se consideram sumários para, em seguida, analisar os fundamentos que permitirão a modelagem.

III -2 Sumários, Índices e Extratos

Sumários científicos podem geralmente ser classificados em três tipos: indicativos, informativos e sumários de crítica (*evaluative*). Sumários indicativos contêm apenas os tópicos essenciais de um texto, não necessariamente contendo detalhes de resultados, argumentos e conclusões. Sumários informativos, por sua vez, são considerados substitutos do texto, devendo conter todos os seus aspectos principais. Assim, se o texto-fonte correspondente for organizado em função de, por exemplo, dados, métodos, hipóteses e conclusões, um sumário informativo deveria conter as informações principais de cada um desses tópicos. Sumários de crítica assumem a função de avaliadores que, por exemplo, apresentam uma análise comparativa do conteúdo do texto-fonte com o contexto de outros trabalhos relacionados a esse conteúdo, na área específica em foco. Seria mais simples produzir automaticamente sumários indicativos, devido à complexidade de se modelar adequadamente a sumarização humana para os demais tipos de sumários. Mais recentemente, procurou-se esclarecer a distinção entre sumários e índices de modo a permitir uma melhor forma de proceder à modelagem lingüístico-computacional. Sumários são também textos, podendo ser substitutos para o documento original. Seriam equivalentes, portanto, aos sumários informativos. No entanto, considera-se que um sumário também pode ser utilizado como índice. Sumários indexadores, no entanto, não poderiam ser utilizados como substitutos para seus textos-fonte, pois não necessariamente estariam preservando o que os originais têm de mais importante, em termos de conteúdo e estrutura. Ao contrário, estariam transmitindo somente uma vaga idéia do texto correspondente, podendo, inclusive, apresentar uma forma não textual (uma lista de itens, por exemplo). Outra diferença entre sumários e índices, reside na avaliação de sua funcionalidade e qualidade. Índices, por exemplo, são utilizados na classificação de documentos bibliográficos, de um modo geral,

indicando seu conteúdo e agilizando o acesso às suas informações relevantes. Através deles, o leitor de uma enciclopédia poderia ir direto ao volume ou página de determinado assunto de interesse. Dessa forma, a utilidade de índices é mais clara e sua função mais limitada do que as de sumários, permitindo uma avaliação mais robusta de sua funcionalidade e qualidade. Do seu objetivo dependerá bastante a avaliação sobre o quanto ele atende às necessidades do usuário. Extratos são outras formas de sumários, podendo ser entendidos como uma composição de segmentos relevantes de um texto. Na sumarização automática, sentenças inteiras podem ser extraídas de um texto e justapostas, sem qualquer modificação, para compor um sumário. O problema, neste caso, é identificar, no texto-fonte, aquelas sentenças que sejam relevantes para transmitir a idéia principal do texto. Métodos estatísticos são explorados com esse fim, como veremos adiante.

As variações conceituais ilustradas acima, sobre o que se consideram sumários, embora compreensíveis do ponto de vista de um falante, são noções ainda obscuras para o projeto e desenvolvimento de sumarizadores automáticos. Além de métodos estatísticos, são explorados também métodos profundos, para se tentar obter modelos computacionais que reflitam as principais características da sumarização humana.

O ponto central da sumarização é reconhecer, em um texto, o que é relevante e o que pode ser descartado, para compor um sumário. Esse é também um dos pontos mais problemáticos e sujeito a controvérsias. A importância de uma sentença ou trecho de um texto pode depender de vários fatores:

a) dos objetivos do autor do sumário,

b) dos objetivos ou interesse de seus possíveis leitores,

c) da importância relativa (e subjetiva) que o próprio autor (ou leitor) atribui às informações textuais.

A análise do conteúdo de documentos é uma das atividades mais importantes de um sistema de informação e entender o modo como profissionais (indexadores ou sumarizadores humanos) produzem seus textos pode levar a avanços consideráveis para a automação do processo. No entanto, há uma grande dificuldade em saber exatamente como esses profissionais trabalham, devido à subjetividade da tarefa.

III-3 A Estrutura Textual

Para conseguir produzir um bom sumário, é necessário que o sumarizador consiga reconhecer e reproduzir as idéias principais do texto. De forma semelhante, um sistema de sumarização automática precisa, de alguma forma, entender o que está sendo apresentado. A

compreensão do discurso, por sua vez, envolve o reconhecimento das estruturas do texto. Assim, antes da sumarização propriamente dita, faz-se necessário investigar a estrutura do discurso.

III-4 O Conteúdo Textual

Além de determinar o conteúdo relevante de um texto-fonte, para compor um sumário, além dos sinais superficiais (sejam eles estruturais ou lingüísticos) aos quais recorre o autor do texto em sua escrita, outros fatores também devem ser considerados pelo sumarizador. Destacam-se, principalmente:

a) o domínio do assunto específico, que o sumarizador detém para entender e, portanto, abstrair ou generalizar as informações que ele lê no texto-fonte e;

b) o conhecimento prévio que ele possa ter.

São relatadas sobre o fato de sumários apresentarem informações que não necessariamente se encontram aparentes nos respectivos textos-fonte. Essa observação remete, muitas vezes, a evidências de que, ao contrário do que se espera normalmente, sumários são construídos de forma independente, ou até mesmo antes, de seus textos correspondentes, apresentando, inclusive, objetivos diversos dos mesmos.

Muito embora essas considerações sobre o conteúdo textual sejam altamente relevantes para o entendimento do processo de sumarização, elas são de difícil modelagem computacional, pois apresentam um alto grau de subjetividade e requerem uma representação bastante complexa do conhecimento de domínio, além de um modelo do escritor e/ou leitor também elaborado, para dar conta de decisões variáveis. Em geral, os modelos explorados para a sumarização automática se baseiam no conteúdo explícito dos textos-fonte e em suas características estruturais, quando se baseiam em metodologias profundas. Quando contemplam técnicas superficiais, baseiam-se também em suas características estruturais.

Tornaram-se freqüentes, ainda, as pesquisas sobre sumarização multi-documentos, cujo objeto de estudo é a produção automática de sumários de um assunto, explorando-se vários textos-fonte que discorrem sobre o mesmo.

III-5 Abordagem Superficial

Os principais métodos encontrados na literatura sobre a abordagem superficial são:

- Método das Palavras-Chave
- Métodos das Palavras-Chave do Título
- Método da Localização

- Método Relacional ou Adaptativo
- Método das Palavras Sinalizadoras (*Cue Phrases*) ou dos Marcadores Lingüísticos
- Método da Frase Auto-Indicativa
- Mineração de Textos (*Text Mining)*

Tratar de todos esses métodos foge ao escopo desta monografia, e portanto nos prenderemos somente a métodos que contribuirão com alguns de seus conceitos, para que possamos definir mais adiante nossas propostas (identificação de possíveis atores e descrição de casos de uso além da extração de indicadores de qualidade de um documento de requisitos).

Método das Palavras-Chave

Este método assume que as idéias principais de um texto podem ser expressas por algumas palavras-chave. Conforme as idéias vão sendo desenvolvidas no texto, os termos-chave aparecem com maior freqüência. A idéia é, então, determinar a distribuição estatística das palavras-chave do texto e, a partir de sua freqüência (considerando somente palavras que se encontram no conjunto de substantivos, verbos, advérbios e adjetivos), extrair as sentenças que as contenham, agrupando-as de forma a constituir um sumário, na ordem em que aparecem originalmente.

Método das Palavras Sinalizadoras (*Cue Phrases*) ou dos Marcadores Lingüísticos

Este método consiste em atribuir valores a sentenças de um texto, selecionando as de maiores pesos. O método faz uso de um dicionário composto por palavras consideradas relevantes no domínio do texto previamente construído, a partir do qual as sentenças consideradas importantes têm pesos associados positivos. Sentenças cujas palavras não constem do dicionário têm pesos negativos. Assim, o peso de uma sentença passa a ser uma média ponderada entre valores negativos e positivos, conforme a especificação dicionarizada. Por exemplo, em um texto científico as palavras *conclusões* e *resultados* serão altamente significativas, estando presentes no dicionário. Sua ocorrência em alguma sentença implicará o aumento da relevância da mesma, em relação à sua escolha para compor um sumário. Textos de gêneros distintos teriam outros dicionários e seus próprios marcadores da importância do conteúdo. No entanto, a correspondência entre os dicionários de marcadores e os diferentes gêneros textuais não é biunívoca e, assim, as intersecções podem levar a imprecisões na seleção dos componentes de um sumário (por exemplo, quando os marcadores dicionarizados sequer aparecem no texto-fonte). Embora se assemelhe ao método das palavras-chave, neste método, o dicionário não é composto

por palavras-chave ocorrentes no texto-fonte, mas por palavras consideradas significativas como marcas da relevância.

Método da Frase Auto-Indicativa

Este método faz uso de um indicativo explícito de quão significativa é a sentença a ser selecionada para compor o sumário. Uma frase auto-indicativa apresenta uma estrutura cuja ocorrência é freqüente no texto e indica explicitamente que a sentença se refere a algo importante sobre o assunto do texto.Exemplos de frases auto-indicativas são *O objetivo deste artigo é investigar...*; *Neste artigo é descrito um método para…*, etc. Essas frases auto-indicativas permitiriam somente a produção de sumários indicativos, ou seja, aqueles que ajudam a identificar o tópico de um texto, sem, no entanto, discorrer sobre o mesmo. Este método é similar ao método dos marcadores lingüísticos, pois também depende, de certa forma, do gênero textual. No entanto, não trata de palavras (unitárias) de conteúdo do domínio do texto-fonte, propriamente, mas de frases relativamente genéricas entre diversos gêneros textuais que irão indicar o conteúdo relevante.

Mineração de Textos (*Text Mining*)

Com a disponibilidade de grandes quantidades de dados e os avanços tecnológicos para sua manipulação eletrônica, houve novamente um interesse bastante grande pela aplicação de métodos estatísticos em larga escala. Esses métodos são comumente chamados métodos baseados em *corpora* e têm demonstrado que é possível, para alguns gêneros textuais ou domínios particulares do conhecimento, obter-se sumários bons e úteis para algum objetivo previamente estabelecido.

Esses *corpora*, também utilizados na área de *Data Mining*, motivaram o surgimento da área de *Text Mining,* ou *Text Data Mining* , área bastante interessante para Recuperação da Informação e Sumarização Automática. Enquanto em *Data Mining* contempla-se dados estruturados, em *Text Mining* busca-se relações existentes entre componentes de textos não estruturados. Esse inter-relacionamento pode ser interno, isto é, relativo a apenas um texto, ou externo, abrangendo vários textos. Para a recuperação da informação é importante buscar informações representativas em algum texto em particular. Portanto, para distingui-lo, é necessário investigar vários textos. Para a sumarização automática é importante identificar informações relevantes em um dado contexto textual contemplado em um único texto.

Estamos interessados em como *Text Mining* pode ser utilizada na sumarização automática. Remetendo à obtenção de palavras ou sentenças-chave de um texto para a composição de um

sumário. A seguir demonstraremos como essa questão é tratada no âmbito da mineração de dados textuais.

Um dos tratamentos para produção de sumários por meio da extração de sentenças é a utilização de freqüência de palavras para identificar palavras relativamente únicas ao documento e que tendem a indicar os tópicos ou carregar informações importantes. A partir de um conjunto de documentos, o sistema compara a freqüência das palavras de cada documento em relação a freqüência de palavras em um conjunto de treinamento. O sistema guarda seu número de ocorrências, a partir da qual a palavra recebe um peso, relativo também ao número de ocorrências no *corpus* de treinamento. Palavras que ocorrem mais freqüentemente no documento do que no conjunto de treinamento recebem um peso maior. Isso equivale à medida chamada TF-IDF (*Text Frequency-Inverse Document Frequency*), utilizada na Recuperação de Informações. A medida TF-IDF é derivada da estatística e se baseia na freqüência de termos. E parte do princípio de que uma palavra será representativa em um texto se ocorrer diversas vezes no texto em questão e for pouco freqüente em outros textos. Usando essa medida, palavras cujo peso indicam que carregam informações importantes são separadas em uma lista, chamada lista de *signature words*. São também adicionadas a esta lista palavras do título, ainda que pouco freqüentes. Compondo uma lista de palavras que são relativamente únicas ao documento. O peso das sentenças é determinado a partir do peso das palavras que a compõem, ou seja, o peso resultante será a soma dos pesos das palavras da sentença que também estiverem presentes na lista de *signature words*.

Outra abordagem para *text mining* realiza as tarefas de *clustering* de documentos e sumarização textual conforme descrito. O algoritmo de sumarização é também baseado na freqüência inversa de palavras, tendo algumas modificações. A principal peculiaridade do método é que a sumarização pode ser aplicada a um texto específico, sem a necessidade de um grande *corpus* de textos. Nos casos cuja freqüência das palavras é primordial, é necessário formatar o texto para que palavras iguais com formatação diferente não sejam erroneamente consideradas palavras distintas. Do mesmo modo, várias designações de um mesmo verbo devem ser consideradas em conjunto. Portanto, inicialmente o texto de entrada sofre um pré-processamento, consistindo de três passos:

1) *Case Folding*

Consiste em converter todos os caracteres do documento para um mesmo formato, ou seja, letras maiúsculas ou minúsculas.

2) *Stemming*

Consiste em converter cada palavra à sua forma radical, ou seja, sua forma neutra sem inflexões verbais ou de número.

3) Remoção de *Stop Words*

Stop Words são palavras de classe fechada, ou seja, que não carregam significado, tal como artigos e pronomes. A remoção é feita verificando a presença em uma lista de *stop words*.

Os procedimentos 2) e 3) se aplicam para a língua inglesa. No caso de algoritmos específicos para línguas diferentes do inglês, é utilizado um processo baseado em n-gramas. Assim, alternativamente, o sistema aplica a representação *n-grama* em vez de executar esses passos. Um *n-grama* é uma parte de uma palavra consistindo de n caracteres. Por exemplo, a palavra DATA pode ser representada pelos trigramas *_DA, DAT, ATA, TA_* ou pelos bigramas _D, DA, TA, entre outros.

O algoritmo de sumarização representa cada sentença como um vetor de pesos TF-ISF (*Term Frequency - Inverse Sentence Frequency*), métrica similar à TF-IDF. A diferença é que a noção de documento é substituída pela noção de sentenças. Assim, a importância de uma palavra w em uma sentença s, denotada por TF-ISF(w,s), é calculada através da fórmula:

TF-ISF(w,s) = TF(w,s)*ISF(w) , onde:

TF(w,s): número de vezes que a palavra w ocorre na sentença s

ISF(w): freqüência inversa da sentença. Obtida através da fórmula:

ISF(w) = log(|S|/SF(w)) , onde:

SF(w): número de sentenças nas quais a palavra w ocorre.

Em seguida, para cada sentença s, o peso médio de cada sentença, denominado Avg-TF-ISF(s) é calculado através da média aritmética dos pesos TF-ISF(w,s) de todas as palavras w pertencentes à sentença, ou seja:

$$\text{Avg-TF-ISF(s)} = \sum_{i=1}^{W(s)} [\ \text{TF-ISF(i,s)}/\ W(s)\ ]$$

onde W(s) é o número de palavras na sentença s.

Finalmente, são selecionadas as sentenças com os maiores valores de Avg-TFISF(s), baseadas em um valor mínimo (*threshold*) definido pelo usuário.

IV – Extração de Possíveis Atores e Casos de Uso

Atores são entidades externas que interagem com um determinado sistema, sendo os mais comuns usuários, outros sistemas, outras organizações, e dispositivos externos. Sua identificação é parte do processo de análise de um sistema, e uma atividade na PBR. Atores iniciam eventos, ou interagem com respostas a eventos de sistema. E quando um ator é um usuário humano, ele representa um papel e não uma pessoa específica.

Casos de Uso são cenários de utilização de um sistema, enfocando o funcionamento do sistema através da idéia ***do que acontece*** sob o ponto de vista do usuário, não interessando em ***como*** o sistema funciona internamente sob determinado cenário. Auxiliam também na definição das interfaces e protocolos de comunicação do sistema. Casos de Uso são construídos para capturar, através de descrição textual estruturada e gráficos, os requisitos funcionais presentes num determinado cenário, podendo inclusive ser usado como veículo para que clientes e desenvolvedores discutam as funcionalidades do sistema, compondo assim uma base para gerenciamento, especificação, compreensão, teste e verificação de requisitos funcionais.

Para exemplificar os conceitos apresentados, imagine que se deseja desenvolver um sistema de reserva de vôos de uma companhia aérea. No documento de requisitos elaborado para este sistema está descrito como um dos requisitos funcionais a seguinte sentença: "O sistema deve efetuar a reserva de passageiros". Fica claro que teremos um caso de uso para fazer as reservas de vôos de passageiros. Onde, pela frase contida no documento de requisitos, é possível se identificar um ator, "Passageiro", e o Caso de Uso propriamente dito, "Fazer reserva". Essa situação pode ser facilmente representada em UML da seguinte maneira:

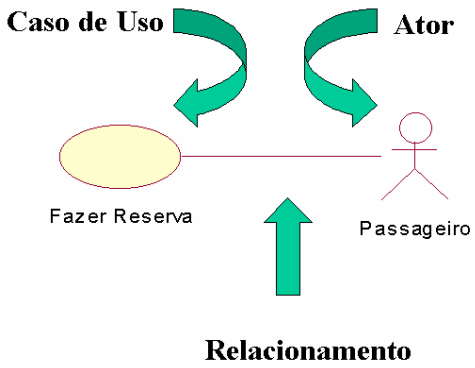

Figura 2: Exemplo de Caso de Uso

Como pode ser observado, o processo que aqui descrevemos foi o de resumir a sentença à idéia principal e assim obtermos um caso de uso e seu respectivo ator, e é assim que

inconscientemente desenvolvedores agem durante o processo de leitura do documento de requisitos em busca de casos de uso e atores.

IV.1 O Método de extração de atores e sentenças indicativas de casos de uso

Uma das estratégias de identificação de atores mais difundidas é a de, diante do documento de requisitos, marcar todos os sujeitos e complementos verbais, que na maioria dos casos são substantivos. A partir daí, de acordo com o domínio do problema, selecionar esses substantivos que representam entidades que interagem com um sistema, tomando-se cuidado com sinônimos e com descrições expressas em mais de uma palavra, como por exemplo "Gerente de Assuntos Comerciais" e "Gerente de Marketing".

Nosso objetivo não é simular ou aplicar tal estratégia, mas em cima dela definirmos uma nova estratégia que seja relativamente simples e que não dependa de análises gramatical e sintática formais da língua portuguesa. Isto custaria uma aquisição de conhecimento gramatical e sintático da língua portuguesa, tarefa muito complexa para nossos propósitos. Assim, dentre as técnicas de PLN, usaremos mais especificamente métodos de Sumarização de Textos, para identificarmos possíveis Atores e sentenças indicativas de Casos de Uso.

A estratégia consiste, como na sumarização de texto extrativa, fazer a contagem de palavras, mas somente sujeitos e complementos verbais, ou seja, especificamente substantivos, presentes em um documento. Devemos evitar que as palavras que diferem apenas no número (singular ou plural) sejam contadas como palavras diferentes. Assim, palavras presentes no singular ou no plural num documento serão contadas juntas. Por exemplo, casa e casas seriam contadas como duas ocorrências de um único símbolo, "cas". Papel e papéis seriam contados como duas ocorrências do símbolo "pap". Como não estamos definindo um símbolo apenas como uma representação de palavras que contêm os mesmos prefixos e os mesmos radicais, não cometeríamos o erro de, por exemplo, representar papelaria e papel sob o mesmo símbolo. Não corremos então o risco de concentrar apenas na contagem de ocorrências de radicais, evitando que conceitos diferentes sejam vistos da mesma maneira. Outro cuidado a ser tomado é considerar sob o mesmo símbolo uma mesma palavra escrita com letras maiúsculas ou minúsculas (lembrando que na língua portuguesa orações são iniciadas com a letra da primeira palavra em maiúscula).

A solução dada será reescrever com letras minúsculas, sem seus possíveis acentos, e sem seus indicadores que caracterizam o singular ou plurais. Na língua portuguesa são a(s), e(s), i(s), o(s), u(s), al(is), ão(ãos,ães,ões), el(is), il(s), z(es), m(ns), etc. A localização de substantivos se daria pela localização de palavras antecedidas por artigos, preposições e contrações de ambos. E

substituir pronomes relativos pelo símbolo antecendente encontrado pelo método, computando suas ocorrências no símbolo selecionado para substituí-los. Este último pode ajudar a encontrar requisitos ambíguos, já que o inspetor pode verificar se é possível determinar corretamente o substantivo ao qual se refere a sentença.

Depois definiríamos um *treshold* que seria um percentual que aplicado ao símbolo mais freqüentes daria o limite mínimo de ocorrências de um símbolo para que ele seja considerado relevante, todos os outros serão descartados. Poderia-se alegar que essa abordagem de localização de substantivos poderia selecionar verbos e adjetivos, mas levando-se em consideração que um documento de requisitos geralmente utiliza-se uma descrição formal e estruturada, tornando, se não impossível, muito baixa a ocorrência de locuções e de sujeitos compostos de muitas palavras sendo a primeira um adjetivo, problema esse contornado com o *treshold*. A próxima etapa seria exibir as palavras das quais derivaram-se os símbolos relevantes e apresentá-los ao inspetor, que selecionaria aqueles que dentro do seu conhecimento do domínio configuram atores de um sistema. E por fim, seria selecionar todas as sentenças onde essas palavras ocorrem, exibindo-as uma a uma para que o inspetor leia e identifique através de sua análise, casos de uso. Sendo possível também criar um relatório com as sentenças selecionadas relacionando-as com as palavras identificadas que a tornaram uma sentença selecionada. Poderia-se também exibir sentenças onde palavras como “deve” , “deveria”, “deverá”, “deverão”, “devem”, “deveriam” ocorram mesmo que não contenham um símbolo selecionado, pois é notório em grande parte dos documentos de requisitos que tais palavras estão relacionados a algo que deve ser provido por um sistema, um requisito funcional.

IV.2 Resumo do Método:

- Passo 1: Fazer a contagem de palavras antecedidas por artigos, preposições e contrações de ambos (supostos substantivos), presentes em um documento sem acentos e os seus sufixos que caracterizam o singular e o plural, que na língua portuguesa são a(s), e(s), i(s), o(s), u(s), al(is), ão(ãos,ães,ões), el(is), il(s), z(es), m(ns), etc. Ou substituir pronomes relativos pelo símbolo antecendente e computando suas ocorrências no símbolo usado para substituí-lo.
- Passo 2: Pegar o símbolo com maior freqüência e multiplicar seu número de ocorrência pelo *treshold*.
- Passo 3: Exibir as palavras das quais derivaram-se os símbolos relevantes e apresentá-los ao inspetor, que selecionaria aqueles que dentro do seu conhecimento do domínio configuram atores de um sistema.

- Passo 4: Selecionar todas as sentenças onde esses símbolos ocorrem, ou sentenças onde palavras como “deve” , “deveria”, “deverá”, “deverão”, “devem”, “deveriam” ocorram mesmo que não contenham um símbolo selecionado, para que o inspetor leia e identifique através de sua análise aqueles que são casos de uso.

IV.3 A Gramática do *parser*

A gramática que definimos, apesar de relativamente simples, tem um poder de expressão necessário para o nosso propósito, que como explicado é fazer a contagem de ocorrência de símbolos que representam possíveis substantivos. Além disso, esta é uma gramática recursiva e não ambígua, ou seja, uma sentença não pode ter mais de uma interpretação.

A seguir a gramática:

<S>→ S’ Pontuação S | ε

<S’> → I A I’

<A> → artigo substantivo C

<C> → preposição B C| pronome C | artigo pronome C

(para indentificar o substantivo a que se referente o pronome)

<B> → nome | pronome

<I> → ignorado I | preposição I | ε

<I’> → ignorado I’ | ε

<Pontuação> → . | ! | ? | ...

Onde:

1- <S> é o estado inicial.

2- <S’> é uma sentença separada pelos sinais de pontuação.

3- <A> é um estado onde encontro o primeiro artigo da sentença, sendo assim todos as palavras seguidas de preposição antes de ser encontrado algum artigo serão ignorados. Desta maneira eliminamos casos tais como: “De acordo com...”, “De vez em quando”, “Para efeito de...”, etc. Visto que na língua portuguesa não temos sujeitos preposicionados.

4- <Preposição> representa todas as palavras dessa classe.

5- <Pronome> representa todos os pronomes relativos.

6- <Ignorado> representa todas as palavras que serão excluídas segundo nossa heurística.

7- <Pontuação> representa os sinais de pontuação que separam sentenças.

IV.4 Análise de Complexidade Temporal do Método

Analisando o método poderíamos construir um *parser* que analisaria o texto da esquerda para direita, possivelmente recursivo, percorrer o arquivo texto uma única vez para processar os símbolos dando uma complexidade de O(n), onde n é o número de palavras do texto, e fazer a contagem das ocorrências com complexidade O(n log n), utilizando uma tabela *hash*. Por último restaria percorrer o texto novamente e selecionar as sentenças, que utilizando o algoritmo de Floyd de complexidade linear, para dectectar sub-sequências, nos daria uma complexidade $O(n^2)$. Concluindo teríamos um algoritmo com complexidade $O(n^2)$, pois em cada sentença deve se buscar uma subcadeia de caracteres igual a um dos símbolos relevantes.

IV.5 – Observações Finais

Repare que não tivemos a pretensão de apontarmos uma estratégia determinística que, diante de um documento de requisitos, dê todos os atores e casos de uso. Nossa estratégia daria algumas sugestões de possíveis atores e sentenças que, representando requisitos funcionais de um sistema, podem configurar casos de uso, levando-se em consideração uma relevância do número de ocorrências de termos em um documento de requisitos.

Tal proposta já seria de grande utilidade para o processo de desenvolvimento de software. Qualquer que seja o percentual de aceitação das sugestões pelos desenvolvedores, o trabalho mecânico de se buscar os atores mais prováveis seria feito automaticamente. Vale lembrar que a decisão de aceitá-los ou não caberá sempre ao desenvolvedor. Além disso, estaremos dando uma apoio que as ferramentas atuais de modelagem não dão (Exemplos: Rational Rose, Argo UML).

V – Extração de Indicadores de Qualidade

Seria muito interessante se pudéssemos avaliar, através de métricas, a qualidade de um documento de especificação de requisitos antes mesmo do processo de inspeção ser realizado.

Em seu artigo, Wilson *et al.* (1997) definem uma série de atributos de qualidade para as especificações de requisitos. Estas deveriam ser completas, consistentes, corretas, modificáveis, classificáveis, rastreáveis, não ambíguas e verificáveis.

Num documento de especificação de requisitos é raro encontrarmos atributos de qualidade subjetivos. Entretanto, existem aspectos deste documento que podem ser medidos e que poderiam fornecer indícios de sua qualidade.

O tamanho de um documento, por exemplo, é um atributo objetivo. Ele poderia ser medido através do número de palavras ou do número de especificações individuais que o

documento contem. A profundidade hierárquica das especificações contidas no documento pode dar indícios do seu nível de detalhamento. Também seria possível verificar a ocorrência de termos ou frases específicas que sinalizassem a fraqueza ou importância de uma especificação.

Parece então ser clara a necessidade de se categorizar estes atributos. Teríamos atributos relacionados às especificações individuais bem como atributos que se relacionassem ao documento como um todo. Nos itens que se seguem, será feita uma tentativa de adaptar para a língua portuguesa estes atributos identificados no trabalho de Wilson *et al.* (1997) para a língua inglesa.

V.1 Atributos Específicos:

(a) Imperativos: são os termos que exprimem uma ordem, demandando impreterivelmente um complemento. Abaixo são listados os termos, em ordem decrescente de força, encontrados na base histórica da NASA.

- "*Deve*": usado normalmente para declarar uma capacidade funcional desejada.
- "*Necessário*": mais usado para declarar requisitos não-funcionais (performance ou restrições).
- "*Aplicável à*": utilizado para fazer referência a um outro documento ou padrão.
- "*Responsável por*": freqüentemente utilizado em requisitos para sistemas com arquiteturas pré-definidas.
- "*Irá*": geralmente utilizado para citar recursos a serem fornecidos pelo ambiente operacional.

(b) Sucessivos: termos que acompanham um imperativo, detalhando a especificação sendo feita. Acredita-se que estes termos representariam uma indicação da organização e estruturação dos requisitos, contribuindo para a manutenção do documento. O uso contínuo destes termos indicariam também requisitos complexos e muito detalhados. Entre estes termos teríamos:

- "*Abaixo*";
- "*Seguintes*";
- "*Listados*", "*Listadas*";
- "*Em particular*".

(c) Diretrizes: categorias de palavras utilizadas para ilustrar alguma informação, auxiliando, portanto, na compreensão destas informações. A ocorrência destas palavras estaria relacionada ao nível de precisão com que os requisitos são especificados. Estes termos seriam:

- "*Figura*";
- "*Tabela*";

- "*Exemplo*";
- "*Nota*".

(d) Opcionais: categorias de palavra que dão imprecisão às especificações, deixando-as vagas.

- "*Pode*";
- "*Opcionalmente*".

(e) Frases Fracas: termos que trariam incerteza às especificações, dando margem a que múltiplas interpretações sejam feitas.

- "*Mínimo*";
- "*Ser capaz de*":
- "*Não limitado a*";
- "*Aplicável*";
- "*Normal*";
- "*Capaz de*".

V.2 Atributos Gerais:

(a) Tamanho: poderia ser expresso pelo número de linhas de texto ou pelo número de parágrafos.

(b) Estrutura do Texto/Profundidade de Especificação: número de identificadores encontrados em cada nível hierárquico do documento. Dariam uma indicação da organização, consistência e nível de detalhe do documento.

(c) Clareza: seriam utilizados índices para identificar a clareza de uma especificação. Por exemplo, o índice de Coleman-Liau utilizaria informações como o número de letras de uma palavra e o número de palavras numa oração para determinar este nível de clareza. Um estudo mais detalhado destes índices deveria ser feito para garantir as suas adequações para a língua portuguesa.

V.3 Trabalho Original *versus* Nossa Proposta:

Wilson *et al.* (1997) buscavam avaliar a qualidade de um documento de requisitos. Para este fim, relacionaram indicadores de qualidade aos atributos de qualidade desejáveis para um documento de especificação de requisitos.

Entretanto, a nossa idéia é utilizar estes indicadores de qualidade para fornecer sugestões aos inspetores. A partir do momento que tenhamos uma base histórica de documentos inspecionados, poderíamos comparar os defeitos (e seus tipos) encontrados e os indicadores de qualidade do documento. Assim, teríamos uma relação entre estes indicadores e os tipos de defeito. Espera-se, por exemplo, que o número de frases fracas esteja intimamente relacionado ao número de ambigüidades encontradas após o processo de inspeção.

Assim, de posse destas referências, poderíamos sugerir ao inspetor que busque mais por defeitos de um determinado tipo. Por fim, vale ressaltar que uma espécie de normalização deveria ser feita para permitir o tratamento de documentos de diferentes tamanhos.

VI – Conclusões e Perspectivas Futuras

Estamos muito otimistas com os possíveis resultados e conclusões que teremos com a implementação desses dois modelos computacionais. Acreditamos que, com a aquisição de uma base estatisticamente consolidada de conhecimento e através de experimentação, poderemos coletar informações que nos farão capazes de questionar, aperfeiçoar ou adequar ambos os métodos também para problemas e aplicações além daqueles para os quais foram inicialmente propostos, como por exemplo extração de possíveis casos de teste (outra atividade integrante do processo de inspeção com PBR) .

Referências

BOEHM, B. & BASILI, V., 2001, “Software Defect Reduction Top 10 List”, Janeiro, IEEE Software, pp. 135-137.

CHRISTIENSEN, M. & THAYER, R., 2001, “The Project Manager’s Guide to Software Engineering’s Best Practices”, IEEE Computer Society Press.

COLE, R. (Editor in Chief),1995, “Survey of the State of the Art in Human Language Technology”, National Science Foundation – Oregon Graduate Institute

GLASS, R., 1999, “Inspections – Some Surprising Findings”, Communications of the ACM, April, pp.17-19.

GRADY, R. & VAN SLACK, T., 1994, “Key Lessons in Achieving WideSpread Inspection Use”, July, IEEE Software, pp. 46-57.

IEEE Std 830-1998, 1998, IEEE Recommended Practice for Software Requirements Specifications.

MARTINS, C.B.; Pardo, T.A.S.; ESPINA, A.P.; RINO, L.H.M. (2001). “Introdução à Sumarização Automática”. Relatório Técnico RT-DC 002/2001, Departamento de Computação, Universidade Federal de São Carlos.

SHULL, F., RUS, I., BASILI, V., 2000, “How Perspective-Based Reading Can Improve Requirements Inspections”, July, IEEE Software, pp. 73-79.

WILSON, W., ROSENBERG, L., HYATT, L., 1997, International Conference on Software Engineering (ISCE '97), Boston, MA.