PARADIGMA: Uma Ferramenta de Apoio à Elicitação e Modelagem de Requisitos Baseada em Processamento de Linguagem Natural

Wilson Carlos da Silva[1]
Luiz Eduardo Galvão Martins[2]
[1]*UNASP – Centro Universitário Adventista de São Paulo*
[2]*UNIMEP- Universidade Metodista de Piracicaba*
wilson.carlos@unasp.edu.br
lgmartin@unimep.br

Abstract

The requirements engineers use several techniques to assist them on the elicitation, analysis, specification, validation and requirements management process, but the number of those tools in this segment is still reduced. The requirements are the base for the software development, which, most of time is described in natural language. In this paper a tool called PARADIGMA is presented, it was developed to assist the requirements engineers to identify in a easier way classes, attributes, operations and relationships from the described requirements in natural language, and using the Natural Language Processing (more specifically the morphosyntatics taggers), generating a conceptual classes model of UML (Unified Modeling Language). Besides the mentioned characteristics above, the tool uses the concept of linguistic standards that facilitate the creation of diagrams closer to those created by human. The experimentation of the tool was made by professionals and professors who act in the area of Requirements Engineering. On the basis of the results, we come to the conclusion of the relevance of the tool PARADIGMA in the context of the Requirements Engineering.

1. Introdução

A base para a construção de qualquer software é organizada pela Engenharia de Requisitos, a qual oferece suporte às fases iniciais do ciclo de vida do software. De acordo com Kotonya e Sommerville [5], a Engenharia de Requisitos engloba todas as atividades envolvidas na descoberta, documentação e manutenção de um conjunto de requisitos para um sistema computacional.

A especificação de requisitos é um documento, ou artefato, utilizado para a comunicação entre os clientes, usuários, engenheiros de requisitos e desenvolvedores de software. A maioria das especificações de requisitos é escrita em linguagem natural, sendo complementada, às vezes, por outros tipos de notações, tais como diagramas, equações, modelos formais etc. A especificação de requisitos em linguagem natural facilita a comunicação entre os atores envolvidos, pois não há necessidade de treinamento específico para sua compreensão.

De acordo com uma pesquisa *on-line* conduzida pela Universidade de Trento [9], a maioria dos documentos disponíveis para análise dos requisitos está em linguagem natural, ou é constituída de informações obtidas por intermédio de entrevistas. Esta pesquisa revelou os seguintes números:

- 79% dos documentos de especificação de requisitos são escritos em linguagem natural comum;
- 16% dos documentos são escritos em linguagem natural estruturada; e
- Somente 5% dos documentos são escritos em linguagem formal.

De acordo com os dados apresentados, fica evidente a grande quantidade de documentos em linguagem natural no processo de Engenharia de Requisitos. Existem várias técnicas que auxiliam os engenheiros de requisitos na elicitação dos requisitos, para que posteriormente os requisitos possam ser especificados da melhor maneira possível. Segundo Davis e Hickey [3], essas técnicas são utilizadas pelos engenheiros de requisitos para determinar as necessidades dos clientes e usuários do software. Essas necessidades, sendo identificadas claramente, poderão levar à construção de

um sistema com grande chance de atender às expectativas e às reais necessidades dos usuários.

O objetivo deste artigo é apresentar uma ferramenta de auxílio à elicitação e modelagem de requisitos, esta ferramenta se apóia nos avanços da área de Processamento de Linguagem Natural (PLN). A ferramenta, chamada PARADIGMA, consegue interpretar textos em linguagem natural (escritos em Língua Portuguesa) e gerar automaticamente modelos conceituais de classes (de acordo com a UML).

O restante deste artigo está organizado da seguinte forma: a seção 2 discute como está o cenário de utilização de PLN na Engenharia de Requisitos; a seção 3 apresenta uma visão geral da ferramenta PARADIGMA; a seção 4 relata o processo de experimentação da ferramenta e o cenário de utilização da mesma; na seção 5 são apresentados os resultados obtidos no processo de experimentação da ferramenta; na seção 6 é feita uma análise e discussão dos resultados da experimentação; na seção 7 são apresentados alguns trabalhos correlatos e um comparativo entre eles; a seção 8 conclui o trabalho e traz indicativos de trabalhos futuros.

2. PLN na Engenharia de Requisitos

A motivação para estudos relacionados ao PLN é revolucionar a forma como o computador é utilizado. A maioria do conhecimento humano é registrada na forma de linguagem natural; computadores que puderem entender a linguagem natural poderão utilizar essa informação de forma rápida e para as mais variadas aplicações [2].

A linguagem é um dos aspectos fundamentais do comportamento humano e é um componente crucial em nossas vidas. Na forma escrita, ela serve de base de conhecimento que passa de uma geração para outra. Na forma falada, serve como principal meio de comunicação no dia-a-dia entre as pessoas [2].

Com o avanço da tecnologia e dos computadores nos últimos anos, as informações estão disponíveis em grande quantidade no formato eletrônico. Essas informações estão dispostas nas mais variadas estruturas lingüísticas. A grande quantidade de textos, principalmente no formato eletrônico, fez com que o interesse por métodos empíricos de análise da Língua ressurgisse no início da década de 1990, fazendo com que os trabalhos na área de lingüística de *corpus* aumentassem significativamente [1].

2.1. Etiquetadores de Texto

Os etiquetadores (*taggers*) são as ferramentas utilizadas na etiquetagem automática de *corpus*. A etiquetagem automática tem sido bastante explorada em PLN. As etiquetas ou marcas, chamadas em inglês de *POS tags (Part-Of-Speech tags),* são, principalmente, classes gramaticais (morfossintáticas) das palavras do *corpus*.

Para realizar a etiquetagem de um texto é necessária a definição de um conjunto finito de etiquetas (*tags*) e essas etiquetas devem ter um significado lingüístico associado. Geralmente o significado dessa etiqueta refere-se à categoria morfossintática ou gramatical de uma determinada palavra, dentro de uma Língua e contexto.

O processo de etiquetagem consiste em realizar a associação das palavras que aparecem em um dado texto a uma determinada etiqueta do conjunto finito de etiquetas. Essa associação é feita de acordo com o algoritmo do etiquetador utilizado.

O processo de etiquetagem é executado com o auxílio de três componentes: escrutinador léxico, responsável por identificar os símbolos do texto (pontuação, palavras da língua, palavras estrangeiras, símbolos); classificador gramatical, responsável por atribuir classes gramaticais às palavras; e desambigüizador, responsável por resolver o problema da ambigüidade léxica, para situações onde existem palavras com significados diferentes, mas com mesma grafia. Geralmente essa ambigüidade ocorre quando a palavra é tratada isoladamente, sendo necessária a análise do contexto para resolver o problema. O desambigüizador realiza essa tarefa de analisar e atribuir somente uma classe gramatical para cada palavra do *corpus*.

2.2. Contribuições de Ferramentas que Utilizam PLN na Engenharia de Requisitos

De acordo com Mich et al. [9], ferramentas que utilizam PLN na Engenharia de Requisitos podem trazer uma série de contribuições, dentre as quais destacam-se:

- Auxiliar no gerenciamento de Requisitos;
- Permitir que os engenheiros de requisitos concentrem-se no problema, ao invés de se concentrarem na modelagem;
- Auxiliar na localização de ambigüidades e contradições nos documentos de especificação de requisitos;

- Extrair diretamente dos textos, em linguagem natural, elementos para gerar modelos conceituais;
- Auxiliar na documentação dos requisitos;
- Rastreabilidade entre os textos em linguagem natural e os documentos produzidos (artefatos);
- Tradução dos documentos em várias línguas.

3. Ferramenta PARADIGMA: Visão Geral

A maioria das especificações de requisitos de software é escrita em linguagem natural, e a partir destes documentos de especificação, são gerados os demais documentos necessários para o desenvolvimento do software. A ferramenta PARADIGMA utiliza recursos do PLN em conjunto com padrões lingüísticos identificados nos textos de especificação de requisitos, permitindo a geração automatizada de modelos conceituais de classes utilizando a notação UML.

A ferramenta PARADIGMA permite obter um modelo conceitual de classes através de processo "manual", automatizado ou híbrido. No processo manual, o engenheiro de requisitos deverá desenhar o modelo de classes da forma tradicional, identificando as classes, atributos, operações e associações através do método heurístico.

No processo automatizado, a ferramenta recebe como entrada um texto em linguagem natural. Esse texto pode ser um conjunto de requisitos especificados, cenários de casos de uso ou documentos resultantes das técnicas utilizadas no processo de elicitação de requisitos. O texto de entrada é formatado no padrão exigido pelo etiquetador de texto MXPOST [11]. O etiquetador recebe o texto formatado como entrada e gera um arquivo contendo as marcações das palavras. Essas marcações identificam as palavras em categorias morfossintáticas.

São aplicados os padrões lingüísticos (apresentados na subseção 3.2) no arquivo contendo as marcações, através desses padrões cada palavra é classificada nos tipos: classe, atributo e operação. As associações entre as classes e respectivas multiplicidades são identificadas de acordo com a relação existente entre as palavras candidatas a classe no texto de entrada. Após a identificação de todos os elementos que compõem o modelo de classes, este é gerado de forma automatizada. O engenheiro de requisitos deverá interagir no processo automatizado para vincular os atributos e operações às respectivas classes.

No processo híbrido, ocorre a geração automatizada do modelo de classes, conforme descrito acima, e o engenheiro de requisitos interage com o modelo para realizar os devidos ajustes.

Independentemente do processo utilizado para chegar ao modelo conceitual de classes, o engenheiro de requisitos poderá interagir com o modelo. As funcionalidades implementadas na ferramenta incluem:

- Adicionar marcações no texto de entrada em linguagem natural;
- Analisar e aplicar os padrões lingüísticos definidos previamente, e classificar as palavras nos seguintes tipos: classes, atributos, operações e associações;
- Exibir o texto de entrada, no momento de captura, para facilitar o rastreamento pelo engenheiro de requisitos, para confirmar se a ferramenta classificou as palavras nos tipos corretos;
- Alterar e excluir palavras que estão fora do domínio da aplicação;
- Vincular os atributos e operações às suas classes respectivas;
- Gerar o modelo conceitual de classes automaticamente, com base nos dados capturados pela ferramenta;
- Alterar e/ou adicionar classes, atributos e operações no modelo de classes gerado;
- Adicionar associações entre as classes;
- Excluir classes, atributos, operações e associações do modelo de classes;
- Adicionar o relacionamento de generalização, caso necessário. Vale ressaltar, que a ferramenta não identifica automaticamente esse tipo de relacionamento, mas dá suporte para a modelagem manual;
- Alterar as multiplicidades das associações;
- Salvar/Abrir o modelo de classe conceitual gerado pela ferramenta;
- Organizar o modelo de classes graficamente, através do recurso de arrastar e soltar.

3.1. Arquitetura da Ferramenta

A arquitetura da ferramenta PARADIGMA é apresentada na Figura 1. Nessa visão de alto nível são apresentados os módulos principais da ferramenta, que englobam os textos em linguagem natural, a etiquetagem desse texto, a aplicação dos padrões lingüísticos no arquivo contendo as palavras com as marcações, a interação do engenheiro de requisitos no ambiente visual e a geração do modelo conceitual de classes como artefato de saída.

Os textos de entrada devem estar no padrão ASCII, para que o etiquetador de texto possa gerar um arquivo

similar com as marcações necessárias nas palavras. Os textos em linguagem natural devem ser ajustados no formato exigido pelo etiquetador. Esse formato requer que cada palavra, símbolo e pontuação do texto estejam separados por um espaço em branco.

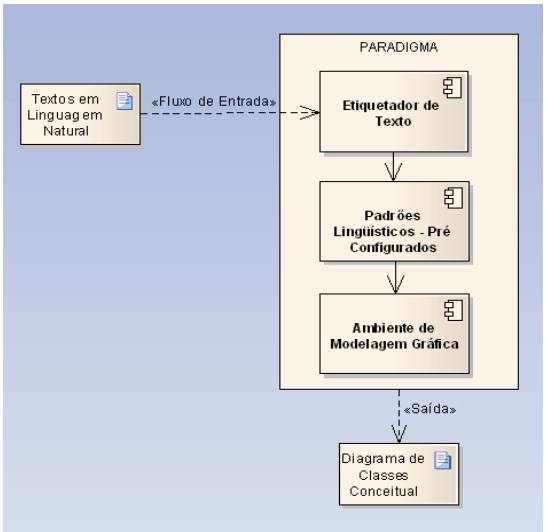

Figura 1 – Arquitetura da ferramenta PARADIGMA

O etiquetador de texto realiza a etiquetagem morfossintática no texto de entrada já formatado. Essa etiquetagem é feita com base na definição de um conjunto finito de etiquetas (*tags*) e essas etiquetas devem ter significados lingüísticos associados a elas. O etiquetador utilizado na ferramenta é o MXPOST e o artefato de saída do etiquetador é um arquivo no padrão ASCII contendo o texto de entrada devidamente etiquetado.

Os padrões lingüísticos são o diferencial da ferramenta PARADIGMA, através deles é possível aproximar os modelos de classes gerados de maneira automatizada dos modelos criados por modelador humano.

O ambiente de modelagem gráfica, ou *interface* gráfica, permite a interação do engenheiro de requisitos de maneira amigável e intuitiva com o modelo de classes gerado pela ferramenta, sendo possíveis os ajustes necessários ao modelo. Como artefato de saída temos o modelo conceitual de classes gerado a partir dos textos em linguagem natural e ajustados pelo engenheiro de requisitos.

3.2. Padrões Linguísticos Pré-definidos

A ferramenta PARADIGMA não é capaz de realizar a análise semântica das frases, para suprir essa tarefa foram definidos padrões lingüísticos. Os padrões lingüísticos nada mais são do que os mapeamentos dos elementos, ou seja, palavras dentro de uma frase. Esse mapeamento tomará como base a classificação gramatical (classificação sintática) de cada palavra e seu posicionamento dentro da frase. De acordo com os padrões lingüísticos, define-se que tratamento será dado àquela palavra. Se ela é candidata a ser uma classe, atributo, operação ou se deveremos simplesmente desprezá-la.

A proposta original era permitir que o engenheiro de requisitos pudesse configurar seus próprios padrões lingüísticos, permitindo que a ferramenta se adequasse à forma de escrita de cada indivíduo. Com as definições de alguns padrões lingüísticos básicos, percebeu-se que uma série de fatores influencia no resultado final, tais como:

- Prioridade de aplicação dos padrões lingüísticos;
- Ambigüidade produzida por padrões lingüísticos definidos de forma equivocada;
- Nível de conhecimento do engenheiro de requisitos em relação à classificação gramatical das palavras na Língua Portuguesa.

Com base nessas constatações, alterou-se a proposta, de forma que continuasse utilizando os padrões lingüísticos, em um formato diferente, onde o engenheiro de requisitos não teria a liberdade de configurá-los, mas a ferramenta traria um conjunto pré-definido de padrões.

Tabela 1 – Definições de padrões lingüísticos adotadas na ferramenta PARADIGMA

Regra	Padrões Lingüísticos	Candidato à	Prioridade
R-01	[Conjunção OU Dois Pontos OU Vírgula] E [Substantivo E [Preposição+ Artigo OU Preposição] E Substantivo] E [Conjunção OU Vírgula OU Ponto OU Preposição + Artigo]	**Atributo**	**01**
R-02	[Conjunção OU Dois Pontos OU Vírgula] E [Nome Próprio OU Substantivo] E [Conjunção OU Vírgula OU Ponto OU Preposição + Artigo]	**Atributo**	**02**
R-03	Verbo E Artigo E [Substantivo OU Nome Próprio]	**Operação**	**03**
	Substantivo E	**Classe com nome**	

R-04	Adjetivo	**composto**	**04**
R-05	Substantivo	**Classe**	**05**
R-06	Palavras: um, uma (artigo indefinido).	**Multiplicidade 1**	**06**
R-07	Palavras: muitos, muitas, vários, várias, bastante, pelos, pelas.	**Multiplicidade ***	**07**
R-08	Classes que estão dentro da frase terminada por [Ponto] ou [Ponto e Vírgula]	**Associação**	**08**

A definição desses padrões não é uma tarefa trivial, foram analisados vários textos em linguagem natural, para que fossem identificados alguns padrões lingüísticos. Algumas palavras atendem mais que uma regra, sendo necessária a definição de prioridades na aplicação dos padrões. A Tabela 1 apresenta a lista de padrões lingüísticos pré-definidos na sua ordem de prioridade.

A simples aplicação dos padrões listados na Tabela 1 não foi suficiente para identificar os elementos do modelo de classes, pois se perceberam os seguintes problemas:

- Os textos de entrada em linguagem natural não poderiam conter acentuações, pois a utilização do modelo conceitual para uma futura implementação apresentaria problemas;
- Identificação da mesma classe, atributo e/ou operação com nomes distintos quando se tratava de palavras no plural e singular;
- O vínculo automático das classes com seus respectivos atributos e operações funcionou de maneira ineficiente.

Os problemas identificados no processo de utilização dos padrões lingüísticos foram solucionados da seguinte forma:

- Eliminação da acentuação em todas as palavras do texto em linguagem natural;
- No processo de classificação das palavras em classe, atributo e/ou operação, deve ser validado se essa palavra não está classificada para um dos tipos (classe, atributo, operação) no singular ou plural;
- Não vinculamos os atributos e operações de forma automática no modelo, permitindo que o engenheiro de requisitos realize essa tarefa através do processo manual de arrastar e soltar. Os atributos e operações são identificados e ficam à disposição do engenheiro de requisitos para uso.

4. Experimentação da Ferramenta

A experimentação da ferramenta PARADIGMA demonstrou-se uma atividade de suma importância. Através dessa atividade, procuramos identificar se a ferramenta atingiu o objetivo proposto de auxiliar o engenheiro de requisitos, através da geração do modelo conceitual de classes a partir dos requisitos em linguagem natural. Foi possível mensurar, de forma preliminar, qual a efetiva contribuição da ferramenta no contexto da Engenharia de Requisitos.

4.1. Participantes

Participaram do experimento dois engenheiros de requisitos com experiência acadêmica e dois engenheiros de requisitos que atuam no mercado de trabalho há mais de 3 anos. Os participantes deveriam seguir um cenário de utilização, onde constava a descrição funcional do *software*, as atividades a serem realizadas durante a experimentação, o questionário para avaliar a usabilidade da ferramenta e um espaço para sugestões de melhorias da ferramenta.

4.2. Cenário de Utilização

Os participantes receberam um roteiro para realização do cenário contendo todos os passos necessários para o processo de experimentação. A descrição funcional em linguagem natural, utilizada no cenário de utilização, foi referente a um software para locação e venda de CDs em uma *Mega Store*. Utilizando essa descrição funcional, os participantes deveriam elaborar o modelo conceitual de classes da forma convencional, e desenhar esse modelo em uma ferramenta CASE de sua preferência (as ferramentas utilizadas foram *JUDE e Enterprise Architect*).

O procedimento de modelagem conceitual de classes, de maneira manual, foi cronometrado. A orientação é que essa atividade não sofresse interrupção, caso houvesse interrupção, os participantes deveriam anotar todos os tempos envolvidos na modelagem para que pudéssemos chegar a um tempo total final.

O próximo passo da experimentação foi gerar o modelo conceitual de classes com o auxílio da ferramenta PARADIGMA. Caso o modelo gerado de forma automática pela ferramenta apresentasse alguma divergência do modelo desenvolvido pelo participante, este deveria realizar os ajustes necessários para chegar ao modelo que ele considerasse "correto". Essa atividade também foi cronometrada.

Depois da realização das atividades de modelagem conceitual de classes manualmente e com a ajuda da ferramenta (automatizada), os participantes

responderam um questionário referente à utilização da ferramenta e caso desejassem, indicando (em alguns casos) melhorias necessárias na ferramenta PARADIGMA através do item sugestões.

Utilizou-se os resultados obtidos através da aplicação deste cenário de utilização para a realização das seguintes avaliações:

- Comparar o modelo conceitual de classes gerado pelo engenheiro de requisitos com o modelo conceitual de classes gerado automaticamente pela ferramenta PARADIGMA;
- Avaliar a relevância da ferramenta no contexto da Engenharia de Requisitos.

5. Resultados da Experimentação

Todos os participantes utilizaram a mesma descrição funcional de software para realizar a atividade de modelagem conceitual de classes. Esses modelos de classes foram criados de forma convencional, sem o auxílio da ferramenta PARADIGMA.

Durante a exposição e análise dos resultados os participantes são identificados como Participante 1 (P1), Participante 2 (P2), Participante 3 (P3) e Participante 4 (P4).

5.1. Comparativo entre os modelos dos participantes e da ferramenta

Os modelos de classes gerados pelos participantes apresentam diferenças significativas entre si, o que nos levou a realizar um comparativo para cada participante. Compararam-se os modelos gerados pelos participantes, com o modelo gerado automaticamente pela ferramenta PARADIGMA. Nesse comparativo, consideraram-se os seguintes elementos do modelo: classes, atributos, operações e relacionamentos. Foram totalizados os elementos identificados somente pela ferramenta, somente pelo participante e os identificados tanto pelo participante quanto pela ferramenta.

Na Figura 2, visualizamos o comparativo entre o modelo gerado pelo P1 com o modelo gerado pela ferramenta. Foram identificadas 33 classes, das quais 16 foram identificadas somente pela ferramenta, 8 foram identificadas somente pelo P1 e 9 classes foram identificadas por ambos, ou seja, 53% das classes do modelo gerado pelo P1 foram contempladas no modelo gerado pela ferramenta. Os atributos e operações não foram considerados pelo P1, não existindo base para a comparação entre os modelos, pois somente a ferramenta identificou atributos e operações. Quanto aos relacionamentos, nenhum dos identificados por P1 coincidiram com os identificados pela ferramenta.

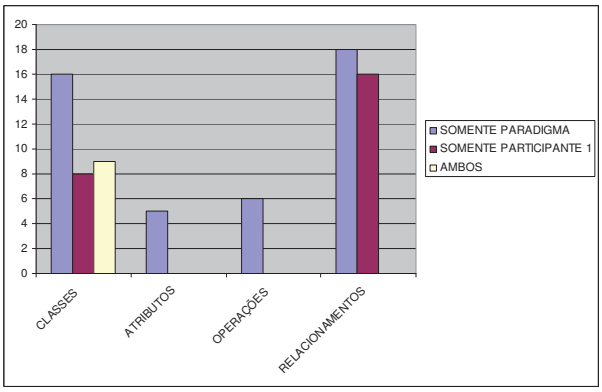

Figura 2 – Gráfico comparativo entre o modelo do P1 e o da ferramenta

Na Figura 3, é apresentado o comparativo entre o modelo gerado por P2 com o modelo gerado pela ferramenta. Foram identificadas 25 classes, das quais 19 classes foram identificadas somente pela ferramenta, e 6 classes foram identificadas por ambos. Nesse caso, 100% das classes do modelo gerado por P2 foram contempladas pelo modelo gerado automaticamente. Foram identificados 20 atributos, nos quais 4 atributos foram identificados somente pela ferramenta; 15 atributos foram identificados somente por P2 e 1 atributo foi identificado por ambos. Foram identificados 6 operações pela ferramenta e 1 operação por P2, não havendo operações identificadas por ambos. Quanto aos relacionamentos, somente 1 foi identificado por ambos, 17 somente pela ferramenta e 6 somente por P2.

Na Figura 4, é apresentado o comparativo entre o modelo gerado por P3 com o modelo gerado pela ferramenta. Foram identificadas 30 classes, das quais 16 classes foram identificadas somente pela ferramenta; 5 classes foram identificadas somente por P3 e 9 classes foram identificadas por ambos. Nesse comparativo, 64% das classes identificadas por P3 foram contempladas pelo modelo gerado pela ferramenta. Foram identificados 57 atributos, dos quais 5 atributos foram identificados somente pela ferramenta; 52 atributos foram identificados somente por P3, e não houve atributos identificados por ambos. O P3 não identificou as operações, somente a ferramenta identificou 6 operações. Quanto aos relacionamentos, 4 foram identificados por ambos, 14 somente pela ferramenta e 13 somente por P3.

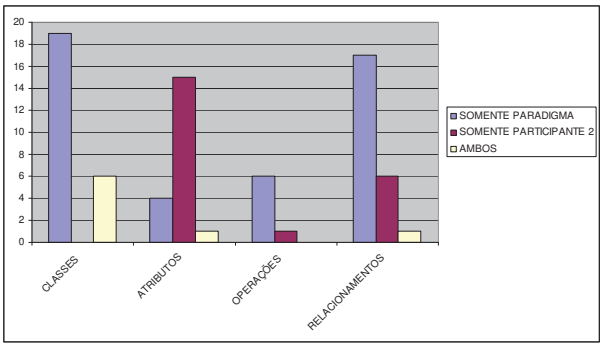

Figura 3 – Gráfico comparativo entre o modelo do P2 e o da ferramenta

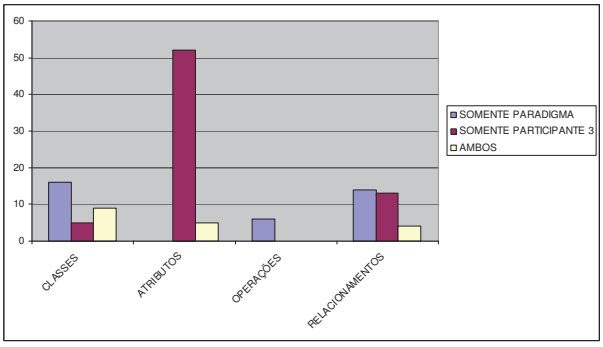

Figura 4 – Gráfico comparativo entre o modelo do P3 e o da ferramenta

Na Figura 5, é apresentado o comparativo entre o modelo gerado por P4 com o modelo gerado pela ferramenta. Foram identificadas 25 classes, das quais 21 foram identificadas somente pela ferramenta, 4 classes foram identificadas por ambos. Nesse caso, 100% das classes identificadas por P4 foram contempladas pelo modelo gerado pela ferramenta PARADIGMA. Foram identificados 13 atributos, onde 8 atributos foram identificados somente por P4 e 5 foram identificados por ambos. Foram identificados 6 operações pela ferramenta e 1 operação por P4, não havendo operações identificadas por ambos. Quanto aos relacionamentos, 18 deles foram identificados somente pela ferramenta e 4 somente por P4, não havendo relacionamentos identificados por ambos.

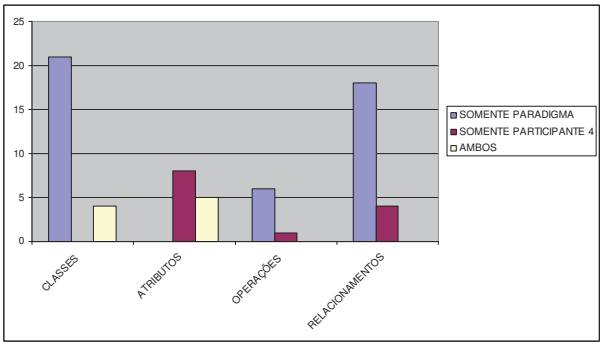

Figura 5 – Gráfico comparativo entre o modelo do P4 e o da ferramenta

5.2. Definição do Modelo Conceitual de Classes Padrão

Os modelos conceituais de classes gerados pelos participantes apresentaram diferenças significativas entre si. Alguns adicionaram detalhes a mais, outros geraram modelos mais gerais, subtraindo detalhes importantes. A descrição funcional pode ter influenciado, pois não trazia muitas informações referentes aos atributos e operações, fazendo com que os participantes focassem mais na identificação das classes e dos relacionamentos.

Para a realização de um comparativo com o modelo conceitual de classes gerado pela ferramenta, foi criado um modelo PADRÃO, construído a partir dos elementos que foram identificados em pelo menos 75% dos modelos dos participantes. Como a ênfase maior dos participantes foi para a identificação das classes, foram levados em consideração as classes e os relacionamentos.

Foram identificadas 27 classes, considerando todos os modelos gerados manualmente pelos participantes, e dessas apenas duas apresentaram ambigüidade em sua identificação (essas classes foram COMPRA e VENDA). 50% dos participantes identificaram a classe COMPRA referindo-se à visão do cliente realizando o ato de comprar CDs na loja, e os outros 50% dos participantes identificaram a classe VENDA referindo-se à visão da loja realizando o ato de vender CDs. Para a definição do modelo conceitual de classes padrão, consideramos a classe VENDA/COMPRA como única (definida como VENDA), totalizando então 26 classes identificadas por todos os participantes.

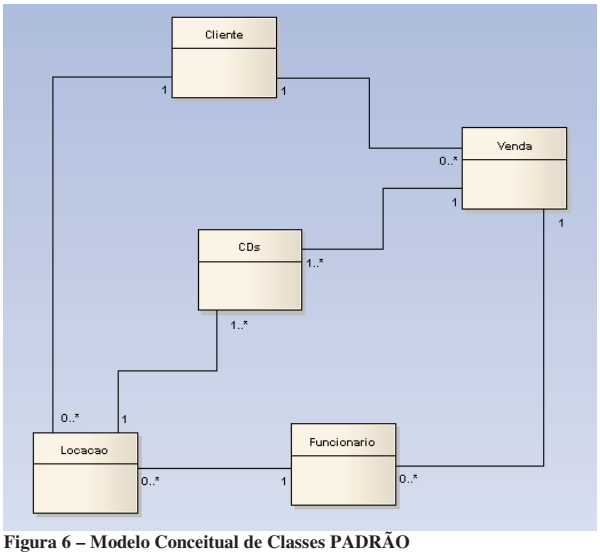

Figura 6 – Modelo Conceitual de Classes PADRÃO

O modelo conceitual de classes PADRÃO é apresentado na figura 6. A partir do modelo de classes padrão, realizou-se o comparativo com o modelo conceitual de classes gerado pela ferramenta PARADIGMA, da mesma forma que foram comparados os modelos dos participantes individualmente. A Figura 7 apresenta o diagrama de classes gerado automaticamente pela ferramenta e a Figura 8 traz o resultado da comparação, onde foram identificadas 25 classes, considerando os dois modelos de classes, o padrão e o gerado pela ferramenta PARADIGMA.

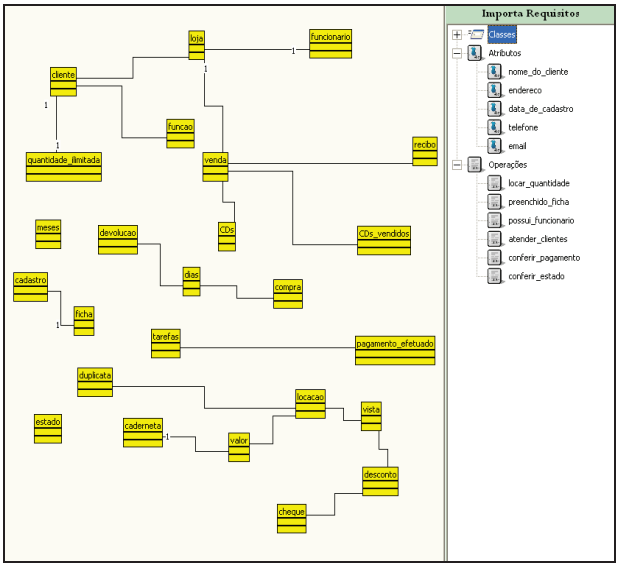

Figura 7 – Modelo Conceitual de Classes gerado pela farramenta

Constatou-se que 100% das classes do modelo conceitual de classes PADRÃO foram identificadas no modelo conceitual de classes gerado pela ferramenta. A ferramenta identificou 20 classes a mais do que as apresentadas no modelo PADRÃO. Quanto aos relacionamentos, foram identificados 23, dos quais 17 somente pela ferramenta PARADIGMA, 5 somente no modelo PADRÃO e 1 relacionamento nos dois modelos.

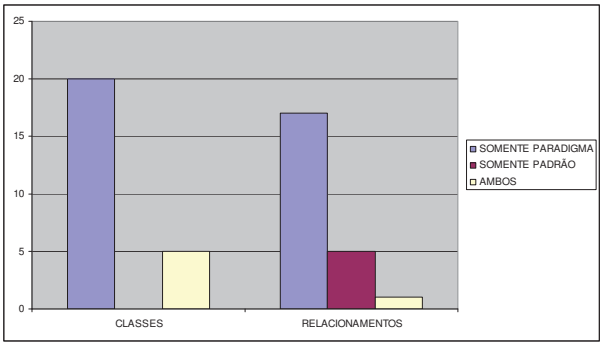

Figura 8 – Gráfico comparativo entre o modelo PADRÃO e o gerado pela ferramenta PARADIGMA.

5.3. Avaliação da Usabilidade da Ferramenta

Através do cenário de utilização, os participantes puderam avaliar algumas características da ferramenta PARADIGMA para indicar o grau de concordância quanto à sua facilidade de uso. Essa avaliação foi feita através de um questionário que utilizou a escala de *Likert* de 5 pontos. Para a análise dos resultados foi utilizada uma abordagem quantitativa para mensurar o grau de concordância dos participantes referente à pesquisa realizada. Utilizamos os seguintes pesos para cada ponto da escala: Concordo Completamente (5), Concordo (4), Indiferente (3), Discordo (2), Discordo Completamente (1).

Tabela 2 – Questionário de avaliação de usabilidade da ferramenta

	Concordo Completamente	Concordo	Indiferente	Discordo	Discordo Completamente	Média Ponderada
Pesos →	**5**	**4**	**3**	**2**	**1**	
1) A ferramenta permite que o engenheiro de requisitos realize as tarefas de forma intuitiva.	1	3	0	0	0	4,25
2) Para utilizar a ferramenta é dispensável a utilização do arquivo de ajuda (AJUDA.PDF).	1	1	0	2	0	3,25
3) O modelo conceitual de classes gerado automaticamente pela ferramenta auxilia o engenheiro de requisitos de forma satisfatória.	0	3	0	1	0	3,50
4) A ferramenta provê recursos necessários para representar satisfatoriamente um modelo conceitual de classes usando a linguagem UML.	3	1	0	0	0	4,75
5) O ajuste do modelo de classes gerado pela ferramenta, para o modelo de classes que o engenheiro de requisitos indica como correto, demanda pouco esforço.	1	1	0	1	1	3,00
Média Geral						**3,75**

A Tabela 2 apresenta os resultados do questionário e as respectivas médias ponderadas por questão e a média geral. A média ponderada foi feita utilizando a seguinte fórmula: MP = [(Peso x Quantidade de Resposta)/Total de Participantes]. Os resultados maiores que 3 foram considerados concordantes, os menores que 3 foram considerados discordantes e os iguais a 3 foram considerados indiferentes.

6. Análise e Discussão dos Resultados

No cenário de utilização proposto para a experimentação da ferramenta, buscou-se fazer uma avaliação da ferramenta PARADIGMA para uso no contexto da Engenharia de Requisitos. Os resultados da experimentação foram de grande valia para a descoberta de alguns desafios pertinentes à utilização do processamento de linguagem natural para a Língua Portuguesa, algumas propostas de trabalhos futuros e algumas melhorias na ferramenta PARADIGMA.

Percebeu-se que os padrões lingüísticos auxiliaram de forma efetiva na descoberta dos elementos do modelo conceitual de classes a partir de textos em linguagem natural, tornando-se um diferencial frente às ferramentas que se propõem a realizar atividades similares às realizadas pela ferramenta PARADIGMA. Constatou-se, após uma bateria de testes, que a identificação dos elementos do modelo conceitual de classes com a aplicação dos padrões lingüísticos ocorreu com um índice maior de acertos quando o texto em linguagem natural está escrito na voz ativa, com redução da utilização de sujeito oculto nas orações.

Percebeu-se que algumas palavras que não fazem parte do domínio da aplicação são identificadas como classes, atributos ou operações, demandando esforço por parte do engenheiro de requisitos para remoção desses elementos. Nos trabalhos futuros pode-se avaliar a possibilidade da ferramenta trabalhar em conjunto com uma base de conhecimento, onde pudessem ser catalogadas palavras do domínio da aplicação e/ou algumas palavras padrões que devam ser desconsideradas, mesmo havendo ocorrência no texto em linguagem natural. Outro item importante é a possibilidade de desprezar parte do texto no momento de realizar as identificações dos elementos.

Outro aspecto importante percebido nos resultados da experimentação foi sobre a questão da ambigüidade. A ambigüidade pode ocorrer em vários momentos e por várias razões. Ficou evidente a existência da ambigüidade na criação do modelo conceitual de classes dos participantes. Quais elementos devem ser modelados em um modelo conceitual de classes ? Depende do que se quer mostrar, neste experimento cada participante teve uma interpretação diferente de quais elementos deveriam adicionar nesse modelo. Alguns desconsideraram os atributos e operações, outros adicionaram atributos e operações que não apareciam na descrição funcional. Alguns identificaram os atributos e operações, mas deixaram de considerar alguns atributos que estavam evidentes na descrição funcional. Quando simplesmente dizemos modelo conceitual de classes, abrimos margem para algo muito abrangente, no qual cada engenheiro de requisitos pode modelar com mais ou menos detalhes, e foi justamente o que ocorreu no processo de experimentação da ferramenta. Mesmo com um número pequeno de participantes conseguimos modelos bem diferentes.

Essa diferença nos modelos criados pelos participantes nos mostra o quão desafiador é desenvolver uma ferramenta que gere um modelo conceitual de classes automaticamente e atenda às expectativas de cada engenheiro de requisitos. Ficou evidente que a ferramenta conseguiu identificar 100% das classes que compunham o modelo PADRÃO. Nas comparações individuais, obtivemos um bom índice de identificação de classes pela ferramenta, iguais as identificadas pelos participantes. O modelo que apresentou o pior índice foi de 53%, sendo que 50% dos participantes tiveram 100% das suas classes identificadas automaticamente pela ferramenta. O que ocorreu em todos os casos é que a ferramenta identificou elementos adicionais, os quais tiveram de ser excluídos pelo participante no ajuste do modelo de classes definido como "correto" para ele.

Os atributos e operações atingiram um índice baixo de aproveitamento devido aos resultados apresentados pelos participantes. Em alguns momentos foram apresentados atributos e operações que não constavam na descrição funcional, caso impossível da ferramenta identificar. Em outros momentos, atributos apresentados na descrição funcional foram simplesmente ignorados, ou seja, a ferramenta identificou mas o participante não.

Quanto aos relacionamentos, são necessárias melhorias na especificação dos padrões lingüísticos para identificá-los de forma mais eficiente. Atualmente, o escopo da definição de um relacionamento está restrito a uma frase terminada por ponto ou ponto e vírgula.

Os resultados referentes ao tempo gasto para ajustar o modelo gerado pela ferramenta e o questionário para avaliar a facilidade de uso da ferramenta, foram obtidos através da utilização da ferramenta pelos participantes, sem nenhuma forma de treinamento. Os participantes baixaram a ferramenta em um *link* na internet e realizaram as atividades de acordo com roteiro de utilização e o arquivo de ajuda.

Após a criação do modelo conceitual de classes de forma convencional e o ajuste do modelo gerado automaticamente pela ferramenta, o tempo utilizado para cada uma dessas atividades foi anotado. Percebeu-se que somente um dos participantes demorou mais tempo no processo de ajuste do modelo gerado pela ferramenta do que na atividade de criar o modelo da forma convencional.

Quanto ao questionário, o posicionamento, ou o grau de concordância dos participantes em relação às afirmativas apresentadas foi de 3,75. O significado disto é que, em regra geral, os participantes concordaram com as afirmativas que avaliaram a intuitividade da ferramenta, facilidade de uso sem o auxílio da ajuda, auxílio satisfatório ao engenheiro de requisitos, recursos suficientes para gerar o modelo conceitual de classes utilizando a linguagem UML.

As observações dos participantes foram sempre em relação à grande quantidade de elementos identificados

pela ferramenta, principalmente os elementos que estão fora do domínio da aplicação. Devemos ponderar o que é menos trabalhoso no momento de gerar um modelo conceitual de classes. Identificar e adicionar uma classe manualmente ou eliminar as classes que não fazem parte do contexto? Como no cenário de utilização não se previu uma atividade para realizar essa ponderação, não temos base para afirmar qual é menos onerosa.

7. Trabalhos Correlatos

Vários estudos têm proposto recentemente a utilização de ferramentas lingüísticas para suporte à Engenharia de Requisitos [4][6][7][8][10]. Existem duas razões principais para que isso ocorra. Primeiro, o progresso feito na área de PLN. Segundo, a necessidade de prover aos engenheiros de requisitos ferramentas que dêem suporte aos estágios iniciais do ciclo de vida do software [9].

O estudo comparativo entre as ferramentas NL-OOPS, DIPETT-HAIKU, CM-BUILDER, LIDA RCR e GOOAL apresentado nas Tabelas 3, 4 e 5 foi realizado por [7]. O código-fonte ou versões funcionais das ferramentas não foram analisadas diretamente, por razões diversas. O quadro comparativo obtido foi produzido com base nos artigos escritos e publicados pelos autores [7].

A partir do estudo comparativo realizado por [7], adicionou-se a ferramenta PARADIGMA. Esse estudo abrangeu ferramentas desenvolvidas em projetos que propõem a utilização do PLN, em que as ferramentas recebem como entrada textos em linguagem natural, e geram diagramas que dão suporte ao paradigma OO. Na Tabela 3 são apresentados os tipos de diagramas que cada ferramenta gera como saída.

Tabela 3 – Modelagem OO automática utilizando PLN

LISTA DE FERRAMENTAS
1. PARADIGMA
2. NL-OOPS
3. DIPETT-HAIKU
4. CM-BUILDER
5. LIDA
6. RECORD
7. GOOAL

Ferramentas / Tipo de Diagrama	1	2	3	4	5	6	7
Casos de Uso	Não	Não	Não	Não	Não	Sim	Não
Classes	Sim	Não	Não	Sim	Sim	Não	Sim
Objetos	Não	Sim	Sim	Não	Sim	Sim	Não
Seqüência	Não	Não	Não	Não	Não	Não	Sim
Atividade	Não	Não	Não	Não	Não	Não	Não

Os diagramas que a maioria das ferramentas geram são os diagramas de classes e objetos. O diagrama de casos de uso é provido somente pela ferramenta RECORD, e GOOAL é a única que provê o diagrama de sequência. Fica evidente a necessidade de pesquisas para atender aos aspectos dinâmicos dos sistemas em desenvolvimento, a maioria das ferramentas se especializou-se em atender a visão estática.

Na Tabela 4 são analisadas as ferramentas que geram o diagrama de classes como saída e os repectivos métodos utilizados para chegar a esse diagrama.

A ferramenta LIDA exige uma grande interação do engenheiro de requisitos em todo o processo de definição do diagrama de classes. As demais ferramentas realizam boa parte do processo de forma automatizada, permitindo a interação do engenheiro de requisitos após a geração do diagrama de classes inicial.

A ferramenta PARADIGMA permite a geração automatizada do diagrama de classes, aplicando as regras identificadas em estruturas comuns de escrita dos requisitos em linguagem natural, para a Língua Portuguesa, os quais chamamos de padrões lingüísticos. Foram identificados oito regras para identificação de classes, atributos, operações, relacionamentos e multiplicidades (conforme apresentado na Subseção 3.2). Após essa identificação inicial, o engenheiro de requisitos poderá interagir com o diagrama realizando os ajustes necessários, utilizando a edição gráfica da própria ferramenta PARADIGMA.

Tabela 4 – Identificando classes utilizando PLN

	PARADIGMA	CM-BUILDER	LIDA	GOOAL
Candidato à Classe	R-04/R-05 [Padrões Lingüísticos]	Substantivo	Engenheiro de Requisitos	Substantivo
Identificando Classes	Palavras que atendem a R-04/R-05 na ordem de prioridade	Freqüência do substantivo no texto	Engenheiro de Requisitos	Técnica de análise linguistica
Identificando Associações	R-08 [Padrões Lingüísticos]	Sujeito/ Verbo/ Objeto	Engenheiro de Requisitos	Linguagem Natural semi-estruturada
Multiplicidade	R-06/R-07 [Padrões Lingüísticos]	Utiliza os determinantes indefinidos, artigos indefinidos ou artigos definidos com substantivo no singular	Engenheiro de Requisitos	Não
Identificando Atributos	R-01/R-02 [Padrões Lingüísticos]	Sim	Engenheiro de Requisitos	Sim
Identificando Operações	R-03 [Padrões Lingüísticos]	Não	Engenheiro de Requisitos	Sim

A ferramenta CM-BUILDER basea-se na freqüência com que cada substantivo aparece no texto para selecionar as palavras candidatas às classes. Resumem em 10 passos, o processo necessário, para partir dos textos já etiquetados até o diagrama de classes (geração

automatizada). Esse diagrama inicial é gerado em um formato que permite a importação para uma ferramenta CASE, de tal forma que o engenheiro de requisitos possa interagir com o diagrama gerado [4].

A ferramenta GOOAL utiliza uma linguagem natural semi-estruturada. As descrições dos requisitos em linguagem são traduzidas para a linguagem semi-estruturada 4W, a partir dessa linguagem, a ferramenta identifica os elementos do diagrama. O engenheiro de requisitos pode interagir nos modelos gerados para realizar as correções necessárias.

As ferramentas capazes de identificar os relacionamentos entre as classes de forma automática são: PARADIGMA, CM-BUILDER e a GOOAL. Na Tabela 5, são apresentados os tipos de relacionamentos que cada ferramenta é capaz de identificar automaticamente.

A associação e suas multiplicidades são identificadas por todas as ferramentas. A agregação é identificada somente pela ferramenta CM-BUILDER. A composição, generalização e dependência não são identificadas automaticamente por nenhuma das ferramentas.

Os relacionamentos estão diretamente ligados ao estilo de escrita em linguagem natural. Para a correta identificação dos relacionamentos de agregação, composição, generalização e dependência, deveriam ser seguidos alguns padrões estruturados de escrita.

Alguns padrões de escrita que permitem facilmente a identificação de alguns relacionamentos são:

- **Agregação:** "... pertence a..."
- **Generalização: "**... é um tipo de...**"**
- **Composição:** "... é parte de..."
- **Dependência:** "... utiliza..." ou "... depende de..."

Como os textos de entrada em linguagem natural não são descritos seguindo essas regras de padrões estruturados, a identificação desses tipos de relacionamentos acaba sendo realizada pela intervenção do engenheiro de requisitos. A ferramenta PARADIGMA permite representar no modelo conceitual de classes, na forma gráfica, somente os seguintes tipos de relacionamentos: associação e generalização.

Tabela 5 – Identificando relacionamentos entre classes utilizando PLN

	PARADIGMA	CM-BUILDER	GOOAL
Associação	Sim	Sim	Sim
Multiplicidade na Associação	Simples/ Múltipla	Simples/ Múltipla	Simples/ específica
Agregação	Não	Sim	Não
Composição	Não	Não	Não
Generalização	Não	Não	Não
Dependência	Não	Não	Não

8. Conclusão

A ferramenta apresentada neste artigo auxilia o engenheiro de requisitos nas atividades de elicitação e modelagem de requisitos, utilizando os recursos de PLN, mais especificamente os etiquetadores morfossintáticos. A ferramenta PARADIGMA utiliza as marcações adicionadas nos textos em linguagem natural pelos etiquetadores, em conjunto com os padrões lingüísticos pré-definidos, para identificar as classes, atributos, operações, associações e multiplicidades, gerando automaticamente um modelo conceitual de classes (atendendo ao padrão UML).

A experimentação da ferramenta PARADIGMA foi realizada por professores e profissionais da área de Engenharia de Requisitos, os resultados trouxeram informações relevantes sobre a ferramenta, das quais podemos destacar:

- A ferramenta apresentou um bom desempenho na identificação das classes. Os percentuais de classes identificadas pela ferramenta que coincidiram com as classes criadas pelos participantes variaram de 53% a 100%. O modelo conceitual de classes padrão, que foi criado a partir dos modelos gerados pelos participantes, teve 100% de suas classes atendidas pela ferramenta PARADIGMA.
- O método adotado para identificar as associações se mostrou pouco eficiente, demandando melhorias futuras.

Os resultados mostraram que a ferramenta PARADIGMA é intuitiva e cumpriu eficazmente seu papel no processo de identificação de elementos conceituais do domínio da aplicação, auxiliando o engenheiro de requisitos durante a elicitação e modelagem de requisitos, que são atividades fundamentais do ciclo de vida do software. A ferramenta libera o engenheiro de requisitos de boa parte da atividade de modelagem, possibilitando que este dedique mais tempo e esforço nas atividades de elicitação e análise dos requisitos. Os primeiros modelos conceituais de classes, tipicamente obtidos a partir dos textos resultantes de entrevistas e outras técnicas de elicitação, podem ser gerados automaticamente pela ferramenta, precisando o engenheiro de requisitos fazer apenas alguns ajustes no modelo gerado.

Como trabalhos futuros identificou-se a necessidade de explorar os diagramas dinâmicos da UML, pois a maioria das ferramentas de geração automática de modelos está focada nos diagramas que representam a estrutura estática do sistema. Também se faz necessária

a utilização de uma base de conhecimentos em conjunto com a ferramenta, para facilitar a identificação das palavras que fazem parte do domínio da aplicação, bem como a implementação de uma *stop list*, ou seja, termos do domínio da aplicação ou da Língua em geral, que não devem ser considerados no modelo. Outros aspectos que devem ser trabalhados no futuro envolvem a busca de soluções para aumento de eficiência na identificação dos relacionamentos, permitindo assim a identificação dos tipos generalização, agregação e composição; e a comunicação com outras ferramentas CASE que atendem às demais fases do ciclo de vida do *software*.

Referências

[1] AIRES, Rachel V.X. "Implementação, adaptação, combinação e avaliação de etiquetadores para o português do Brasil**"**, 2000. 154 f. Dissertação de mestrado – Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, São Carlos, 2000.

[2] ALLEN, James, "Natural Language Understanding". The Benjamin/Cummings Publishing Company Inc, 1995.

[3] DAVIS, A.M., HICKEY, A.M. "Elicitation Technique Selection: How Do Experts Do It", Proceedings of the 11th International Requirements Engineering Conference, IEEE Computer Society Press, 2003.

[4] HARMAIN, H.M., GAIZAUSKAS R. "CM-Builder: An Automated NL-based CASE Tool", In Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE'2000), 2000, p. 45-53.

[5] KOTONYA, G.; SOMMERVILLE, I. "Requirements Engineering: process and techniques", New York: John Wiley & Sons Ltda, 1998.

[6] LI, K., DEWAR, R.G., POOLEY, R.J. "Requirements capture in natural language problem statements", Technical Report HW-MACS-TR-0023, Heriot-Watt University, Edinburgh, Scotland, UK, 2004.

[7] LI, K., DEWAR, R.G., POOLEY, R.J. "Object-oriented Analysis Using Natural Language Processing", Technical Report HW-MACS-TR-0033, Heriot-Watt University Edinburgh, Scotland, UK, 2005.

[8] LIU, D., SUBRAMANIAM, K., EBERLEIN, A., FAR, B.H. "Natural Language Requirements Analysis and Class Model Generation Using UCDA", Springer, 2004.

[9] MICH, L., FRANCH, M. and NOVI INVERARDI, P. "Requirements Analysis using Linguistic Tools: Results of an On-line Survey". Technical Report 66, Department of Computer and Management Sciences, 2003.

[10] OVERMYER, S.; LAVOIE, B.; and RAMBOW, O. "Conceptual Modeling through Linguistic Analysis Using LIDA". In Proceedings of 23rd International Conference on Software Engineering (ICSE 2001), Toronto, Canada, 2001.

[11] RATNAPARKHI, A., "A Maximum Entropy Part-Of-Speech Tagger" Proceedings of the Empirical Methods in Natural Language Processing Conference, University of Pennsylvania, 1996.